Package ‘gratia’

June 25, 2024

Version 0.9.2
Date 2024-06-25
Title Graceful 'ggplot'-Based Graphics and Other Functions for GAMs Fitted Using 'mgcv'
Maintainer Gavin L. Simpson <ucfagls@gmail.com>
Depends R (>= 4.1.0)
Imports mgcv (>= 1.9-0), ggplot2 (>= 3.5.0), tibble (>= 3.0.0), dplyr (>= 1.1.0), tidyr, rlang, patchwork (>= 1.2.0), vctrs, grid, mvnfast, purrr, stats, tools, grDevices, stringr, tidyselect (>= 1.2.0), lifecycle, pillar, cli, nlme, ggokabeito, withr
Suggests gamm4, lme4, testthat, vdiffr, MASS, scam, datasets, knitr, markdown, forcats, GJRM, readr, glmmTMB, gdist, distributional, hexbin, gamair, sf (>= 0.7-3), svglite (>= 2.0.0), curl
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 7.3.1
Encoding UTF-8
VignetteBuilder knitr
Config/testthat/edition 3
Config/Needs/website rmarkdown, gdist
NeedsCompilation no
Author Gavin L. Simpson [aut, cre, cph]
               (https://orcid.org/0000-0002-9084-8413), Henrik Singmann [ctb] (https://orcid.org/0000-0002-4842-3657)
Contents

Repository CRAN
Date/Publication 2024-06-25 17:00:02 UTC

Contents

add_confint ................................................. 4
add_constant ................................................. 5
add_fitted .................................................. 6
add_fitted.gam .............................................. 7
add_fitted_samples ......................................... 8
add_partial_residuals ....................................... 9
add_residuals ............................................... 10
add_residuals.gam .......................................... 10
add_sizer .................................................... 11
appraise ..................................................... 12
basis ......................................................... 14
basis_size ................................................... 16
bird_move .................................................... 17
boundary ...................................................... 18
check_user_select_smooths ................................ 18
coef.scam ..................................................... 19
compare_smooths ............................................. 20
confint.fderiv .............................................. 21
confint.gam .................................................. 23
data_combos ............................................... 25
data_sim ...................................................... 26
data_slice ................................................... 28
derivatives ................................................... 29
derivative_samples ........................................... 32
difference_smooths .......................................... 34
draw .......................................................... 35
draw.basis .................................................... 36
draw.compare_smooths ....................................... 38
draw.derivatives ............................................ 38
draw.difference_smooth ...................................... 40
draw.eevaluated_parametric_term ......................... 42
draw.gam ...................................................... 43
draw.gamlss .................................................. 47
draw.mgcv_smooth ........................................... 48
draw.pairwise_concurvity .................................. 50
draw.parametric_effects .................................... 51
draw.penalty_df ............................................ 52
draw.rootogram ............................................. 54
draw.smooth_estimates ..................................... 55
draw.smooth_samples ....................................... 58
def ........................................................... 60
evaluate_parametric_term ................................... 62
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>evaluate_smooth</td>
<td>63</td>
</tr>
<tr>
<td>eval_smooth</td>
<td>63</td>
</tr>
<tr>
<td>evenly</td>
<td>67</td>
</tr>
<tr>
<td>factor_combos</td>
<td>68</td>
</tr>
<tr>
<td>family.gam</td>
<td>68</td>
</tr>
<tr>
<td>family_name</td>
<td>69</td>
</tr>
<tr>
<td>family_type</td>
<td>69</td>
</tr>
<tr>
<td>fitted_samples</td>
<td>70</td>
</tr>
<tr>
<td>fitted_values</td>
<td>72</td>
</tr>
<tr>
<td>fixef</td>
<td>74</td>
</tr>
<tr>
<td>fixef.gam</td>
<td>74</td>
</tr>
<tr>
<td>fix_offset</td>
<td>75</td>
</tr>
<tr>
<td>gaussian_draws</td>
<td>76</td>
</tr>
<tr>
<td>get_by_smooth</td>
<td>77</td>
</tr>
<tr>
<td>get_smooth</td>
<td>78</td>
</tr>
<tr>
<td>get_smooths_by_id</td>
<td>78</td>
</tr>
<tr>
<td>gss_vocab</td>
<td>79</td>
</tr>
<tr>
<td>gw_f0</td>
<td>79</td>
</tr>
<tr>
<td>has_theta</td>
<td>80</td>
</tr>
<tr>
<td>is_by_smooth</td>
<td>81</td>
</tr>
<tr>
<td>is_factor_term</td>
<td>81</td>
</tr>
<tr>
<td>is_mgcv_smooth</td>
<td>82</td>
</tr>
<tr>
<td>is_offset</td>
<td>83</td>
</tr>
<tr>
<td>link</td>
<td>84</td>
</tr>
<tr>
<td>load_mgcv</td>
<td>86</td>
</tr>
<tr>
<td>lp_matrix</td>
<td>86</td>
</tr>
<tr>
<td>mh_draws</td>
<td>87</td>
</tr>
<tr>
<td>model_concurvity</td>
<td>88</td>
</tr>
<tr>
<td>model_constant</td>
<td>90</td>
</tr>
<tr>
<td>model_vars</td>
<td>90</td>
</tr>
<tr>
<td>nb_theta</td>
<td>91</td>
</tr>
<tr>
<td>null_deviance</td>
<td>92</td>
</tr>
<tr>
<td>n_smooths</td>
<td>93</td>
</tr>
<tr>
<td>observed_fitted_plot</td>
<td>93</td>
</tr>
<tr>
<td>overview</td>
<td>94</td>
</tr>
<tr>
<td>parametric_effects</td>
<td>95</td>
</tr>
<tr>
<td>parametric_terms</td>
<td>96</td>
</tr>
<tr>
<td>partial_derivatives</td>
<td>97</td>
</tr>
<tr>
<td>partial_residuals</td>
<td>100</td>
</tr>
<tr>
<td>penalty</td>
<td>101</td>
</tr>
<tr>
<td>posterior_samples</td>
<td>103</td>
</tr>
<tr>
<td>post_draws</td>
<td>105</td>
</tr>
<tr>
<td>predicted_samples</td>
<td>108</td>
</tr>
<tr>
<td>qq_plot</td>
<td>109</td>
</tr>
<tr>
<td>ref_level</td>
<td>112</td>
</tr>
<tr>
<td>ref_sims</td>
<td>112</td>
</tr>
<tr>
<td>rep_first_factor_value</td>
<td>113</td>
</tr>
<tr>
<td>residuals_hist_plot</td>
<td>113</td>
</tr>
</tbody>
</table>
### add_confint

Add a confidence interval to an existing object

---

**Description**

Add a confidence interval to an existing object
add_constant

Usage

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

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

## S3 method for class 'parametric_effects'
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.

Description

Add a constant to estimated values

Usage

add_constant(object, constant = NULL, ...)

## S3 method for class 'smooth_estimates'
add_constant(object, constant = NULL, ...)

## S3 method for class 'smooth_samples'
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

```
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.
Add fitted values from a GAM to a data frame

Description

Add fitted values from a GAM to a data frame

Usage

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

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.
- `...`: additional arguments passed to `mgcv::predict.gam()`.

Value

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

Examples

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

# add fitted values to our data
add_fitted(df, m)

# with type = "terms" or "iterms"
add_fitted(df, m, type = "terms")
```
add_fitted_samples  Add posterior draws from a model to a data object

Description

Adds draws from the posterior distribution of model to the data object using one of fitted_samples(), predicted_samples(), or posterior_samples().

Usage

add_fitted_samples(object, model, n = 1, seed = NULL, ...)
add_predicted_samples(object, model, n = 1, seed = NULL, ...)
add_posterior_samples(object, model, n = 1, seed = NULL, ...)
add_smooth_samples(object, model, n = 1, seed = NULL, select = NULL, ...)

Arguments

object  a data frame or tibble to which the posterior draws will be added.
model  a fitted GAM (or GAM-like) object for which a posterior draw method exists.
n  integer; the number of posterior draws to add.
seed  numeric; a value to seed the random number generator.
...  arguments are passed to the posterior draw function, currently one of fitted_samples(), predicted_samples(), or posterior_samples(). n and seed are already specified here as arguments and are also passed on to the posterior sampling function.
select  character; select which smooth’s posterior to draw from. The default, NULL, means the posteriors of all smooths in model will be sampled from individually. If supplied, a character vector of requested smooth terms.

Examples

load_mgcv()

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

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

# add fitted samples (posterior draws of the expected value of the response)
# note that there are 800 rows in the output: 400 data by `n = 2` samples.
df |> add_fitted_samples(m, n = 2, seed = 84)

# add posterior draws from smooth s(x2)
df |>
add_partial_residuals

add_smooth_samples(m, n = 2, seed = 2, select = "s(x2)"

Description
Add partial residuals

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

Arguments
- data: a data frame containing values for the variables used to fit the model. Passed to 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
load_mgcv()

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

## add partial residuals
add_partial_residuals(df, m)

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

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 
\texttt{stats::residuals()} as \texttt{newdata}.
model a fitted model for which a \texttt{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 
\texttt{stats::predict()} as \texttt{newdata}.
model a fitted model for which a \texttt{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 \texttt{mgcv::residuals.gam()} for options.
... additional arguments passed to \texttt{mgcv::residuals.gam()}.
Value

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

Examples

```r
load_mgcv()

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

##
add_residuals(df, m)
```

---

**add_sizer**  
Add indicators of significant change after SiZeR

Description

Add indicators of significant change after SiZeR

Usage

```r
add_sizer(object, type = c("change", "sizer"), ...)

## S3 method for class 'derivatives'
add_sizer(object, type = c("change", "sizer"), ...)

## S3 method for class 'smooth_estimates'
add_sizer(object, type = c("change", "sizer"), derivatives = NULL, ...)
```

Arguments

- **object**: an R object. Currently supported methods are for classes "derivatives".
- **type**: character; "change" adds a single variable to object indicating where the credible interval on the derivative excludes 0. "sizer" adds two variables indicating whether the derivative is positive or negative.
- **...**: arguments passed to other methods
- **derivatives**: an object of class "derivatives", resulting from a call to `derivatives()`.
Examples

```r
load_mgcv()

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

## first derivatives of all smooths using central finite differences
d <- derivatives(m, type = "central") |> add_sizer()

# default adds a .change column
names(d)
```

---

appraise  Model diagnostic plots

---

Description

Model diagnostic plots

Usage

```
appraise(model, ...)
```

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

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

model  a fitted model. Currently only class "gam".
...  arguments passed to patchwork::wrap_plots().
method  character; method used to generate theoretical quantiles. Note that method = "direct" is deprecated in favour of method = "uniform".
use_worm  logical; should a worm plot be drawn in place of the QQ plot?
n_uniform  numeric; number of times to randomize uniform quantiles in the direct computation method (method = "direct") for QQ plots.
n_simulate  numeric; number of data sets to simulate from the estimated model when using the simulation method (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 patchwork::plot_layout().
level  numeric; the coverage level for QQ plot reference intervals. Must be strictly $0 < \text{level} < 1$. Only used with method = "simulate".

ci_alpha, ci_col  colour and transparency used to draw the QQ plot reference interval when method = "simulate".
point_col, point_alpha  colour and transparency 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.
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 mgcv::qq.gam() uses direct in reference to the simulated residuals method (method = "simulated"). To avoid confusion, method = "direct" is deprecated in favour of method = "uniform".

See Also

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

Examples

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

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

basis

### 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()
- mgcv::t2()

or from a fitted GAM(M).

**Usage**

```r
basis(object, ...)
```

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

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

... )

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

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

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

Arguments

object a smooth specification, the result of a call to one of mgcv::s(), mgcv::te(),
mgcv::ti(), or mgcv::t2(), or a fitted GAM(M) model.

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

select character; select smooths in a fitted model
term [Deprecated] This argument has been renamed select
data a data frame containing the variables used in smooth.
n numeric; the number of points over the range of the covariate at which to evaluate the smooth.
n_2d numeric; the number of new observations for each dimension of a bivariate smooth. Not currently used; n is used for both dimensions.
basis_size

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 ($k \geq 4$). For example, if the smooth is a 4D smooth, each of dimensions 3 and 4 will get n_4d new observations.

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.

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

constraints logical; should identifiability constraints be applied to the smooth basis. See argument absorb.cons in \texttt{mgcv::smoothCon()}.

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

Value

A tibble.

Author(s)

Gavin L. Simpson

Examples

load_mgcv()

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

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

| basis_size          | Extract basis dimension of a smooth |

Description

Extract basis dimension of a smooth

Usage

basis_size(object, ...)

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

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

### Arguments

- **object**: A fitted GAM(M). Currently `mgcv::gam()` (and anything that inherits from the "gam" class, e.g. `mgcv::bam()`) and `mgcv::gamm()` are supported.
- **...**: Arguments passed to other methods.

### Examples

```r
load_mgcv()

df <- data_sim("eg1", n = 200, seed = 1)
m <- bam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = df)
basis_size(m)
```

---

### bird_move

**Simulated bird migration data**

#### Description

Data generated from a hypothetical study of bird movement along a migration corridor, sampled throughout the year. This dataset consists of simulated sample records of numbers of observed locations of 100 tagged individuals each from six species of bird, at ten locations along a latitudinal gradient, with one observation taken every four weeks. Counts were simulated randomly for each species in each location and week by creating a species-specific migration curve that gave the probability of finding an individual of a given species in a given location, then simulated the distribution of individuals across sites using a multinomial distribution, and subsampling that using a binomial distribution to 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

boundary

Extract the boundary of a soap film smooth

Description

[Experimental]

Usage

boundary(x, ...)

## S3 method for class 'soap.film'
boundary(x, ...)

## S3 method for class 'gam'
boundary(x, select, ...)

Arguments

x an R object. Currently only objects that inherit from classes "soap.film" and "gam".
...
arguments passed to other methods.
select character; the label of the soap film smooth from which to extract the boundary.

Value

A list of lists or data frames specifying the loops that define the boundary of the soap film smooth.

See Also

mgcv::soap

check_user_select_smooths

Select smooths based on user’s choices

Description

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

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

Arguments

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

---

ccoef.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

Description

Compare smooths across models

Usage

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

Arguments

- **model**
  Primary model for comparison.
- **...**
  Additional models to compare smooths against those of `model`.
- **select**
  character; select which smooths to compare. The default (NULL) means all smooths in `model` will be compared. Numeric select indexes the smooths in the order they are specified in the formula and stored in `model`. 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.
- **smooths**
  [Deprecated] Use `select` instead.
- **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?
- **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()
dat <- data_sim("eg1", seed = 2)

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

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

confint.fderiv  
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

## 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.
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

```r
load_mgcv()

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

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

## first derivatives of all smooths...
fd <- fderiv(mod, newdata = new_data)

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

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

Description

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

Usage

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

## 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 frame; new values of the covariates used in the model fit. The selected smooth(s) will be evaluated at the supplied values.

n numeric; the number of points to evaluate smooths at.

type character; the type of interval to compute. One of "confidence" for point-wise intervals, or "simultaneous" for simultaneous intervals.

nsim integer; the number of simulations used in computing the simultaneous intervals.

shift logical; should the constant term be add to the smooth?

transform logical; should the smooth be evaluated on a transformed scale? For generalised models, this involves applying the inverse of the link function used to fit the model. Alternatively, the name of, or an actual, function can be supplied to transform the smooth and its 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 \texttt{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 \texttt{parm} use a partial match or an exact match? Can only be used if \texttt{length(parm)} is 1.

... additional arguments for methods

newdata DEPRECATED! data frame; containing new values of the covariates used in the model fit. The selected smooth(s) will be evaluated at the supplied values.

Value

a tibble with components:

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

Author(s)

Gavin L. Simpson
Examples

load_mgcv()

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

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

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

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,
  envir = environment(formula(object)),
  data = NULL,
  ...
)

Arguments

object a fitted model object.
...
vars terms to include or exclude from the returned object. Uses tidyselect principles.
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 = NULL,
  theta = 3,
  power = 1.5,
  dist = c("normal", "poisson", "binary", "negbin", "tweedie", "gamma", "ocat", "ordered categorical"),
  n_cat = 4,
  cuts = c(-1, 0, 5),
  seed = NULL,
  gfam_families = c("binary", "tweedie", "normal")
)
```

Arguments

- `model` character; either "egX" where X is an integer 1:7, or the name of a model. See Details for possible options.
- `n` numeric; the number of observations to simulate.
- `scale` numeric; the level of noise to use.
- `theta` numeric; the dispersion parameter $\theta$ to use. The default is entirely arbitrary, chosen only to provide simulated data that exhibits extra dispersion beyond that assumed by under a Poisson.
- `power` numeric; the Tweedie power parameter.
- `dist` character; a sampling distribution for the response variable. "ordered categorical" is a synonym of "ocat".
- `n_cat` integer; the number of categories for categorical response. Currently only used for distr %in% c("ocat", "ordered categorical").
data_sim

cuts numeric; vector of cut points on the latent variable, excluding the end points -Inf and Inf. Must be one fewer than the number of categories: length(cuts) == n_cat - 1.

seed numeric; the seed for the random number generator. Passed to base::set.seed().
gfam_families character; a vector of distributions to use in generating data with grouped families for use with family = gfam(). The allowed distributions as per dist.

Details

data_sim() can simulate data from several underlying models of known true functions. The available options currently are:

- "eg1": a four term additive true model. This is the classic Gu & Wahba four univariate term. See gw_functions for more details of the underlying four functions.
- "eg2": a bivariate smooth true model.
- "eg3": an example containing a continuous by smooth (varying coefficient) true model. The model is \( \hat{y}_i = f_2(x_{1i})x_{2i} \) where the function \( f_2() \) is \( f_2(x) = 0.2 * x^{11} * (10 * (1 - x))^6 + 10 * (10 + x)^3 * (1 - x)^{10} \).
- "eg4": a factor by smooth true model. The true model contains a factor with 3 levels, where the response for the nth level follows the nth Gu & Wabha function (for \( n \in 1, 2, 3 \)).
- "eg5": an additive plus factor true model. The response is a linear combination of the Gu & Wabha functions 2, 3, 4 (the latter is a null function) plus a factor term with four levels.
- "eg6": an additive plus random effect term true model.
- "eg7": a version of the model in "eg1", but where the covariates are correlated.
- "gwf2": a model where the response is Gu & Wabha’s \( f_2(x_i) \) plus noise.
- "lwf6": a model where the response is Luo & Wabha’s "example 6" function \( \sin(2(4x - 2)) + 2\exp(-256(x - 0.5)^2) \) plus noise.
- "gfam": simulates data for use with GAMs with family = gfam(families). See example in mgcv::gfam(). If this model is specified then dist is ignored and gfam_families is used to specify which distributions are included in the simulated data. Can be a vector of any of the families allowed by dist. For "ocat" %in% gfam_families (or "ordered categorical"), 4 classes are assumed, which can't be changed. Link functions used are "identity" for "normal", "logit" for "binary", "ocat", and "ordered categorical", and "exp" elsewhere.

The random component providing noise or sampling variation can follow one of the distributions, specified via argument dist

- "normal": Gaussian,
- "poisson": Poisson,
- "binary": Bernoulli,
- "negbin": Negative binomial,
- "tweedie": Tweedie,
- "gamma": gamma, and
- "ordered categorical": ordered categorical

Other arguments provide the parameters for the distribution.
References


Examples

```r
data_sim("eg1", n = 100, seed = 1)
# an ordered categorical response
data_sim("eg1", n = 100, dist = "ocat", n_cat = 4, cuts = c(-1, 0, 5))
```

---

data_slice Prepare a data slice through model covariates

Description

Prepare a data slice through model covariates

Usage

```r
data_slice(object, ...)
```

## Default S3 method:

```r
data_slice(object, ...)
```

## S3 method for class 'data.frame'

```r
data_slice(object, ...)
```

## S3 method for class 'gam'

```r
data_slice(object, ..., data = NULL, envir = NULL)
```

## S3 method for class 'gamm'

```r
data_slice(object, ...)
```

## S3 method for class 'list'

```r
data_slice(object, ...)
```

## S3 method for class 'scam'

```r
data_slice(object, ...)
```

Arguments

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

envir the environment within which to recreate the data used to fit object.

Examples

load_mgcv()

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

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

# Want to predict over f(x2) while holding `x1` at some value.
# Default will use the observation closest to the median for unspecified # variables.
ds <- data_slice(m, x2 = evenly(x2, n = 50))
ds

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

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

derivatives  Derivatives of estimated smooths via finite differences

Description

Derivatives of estimated smooths via finite differences

Usage

derivatives(object, ...)

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

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

## S3 method for class 'gam'
derivatives(
object,
select = NULL,
term = deprecated(),
data = newdata,
order = 1L,
type = c("forward", "backward", "central"),
n = 100,
eps = 1e-07,
interval = c("confidence", "simultaneous"),
n_sim = 10000,
level = 0.95,
unconditional = FALSE,
frequentist = FALSE,
offset = NULL,
ncores = 1,
partial_match = FALSE,
...
newdata = NULL
)

Arguments

object an R object to compute derivatives for.
...
arguments passed to other methods.
select 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. Can be a partial match to a smooth term; see argument partial_match below.
term [Deprecated] Use select instead.
data a data frame containing the values of the model covariates at which to evaluate the first derivatives of the smooths.
order numeric; the order of derivative.
type character; the type of finite difference used. One of "forward", "backward", or "central".
n numeric; the number of points to evaluate the derivative at.
eps numeric; the finite difference.
interval character; the type of interval to compute. One of "confidence" for point-wise intervals, or "simultaneous" for simultaneous intervals.
n_sim integer; the number of simulations used in computing the simultaneous intervals.
level numeric; 0 < level < 1; the confidence level of the point-wise or simultaneous interval. The default is 0.95 for a 95% interval.
unconditional logical; use smoothness selection-corrected Bayesian covariance matrix?
frequentist logical; use the frequentist covariance matrix?
offset numeric; a value to use for any offset term
derivatives

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 smooths be selected by partial matches with term? If TRUE, term can only be a single string to match against.

newdata Deprecated: use `data` instead.

Value

A tibble, currently with the following variables:

- `smooth`: the smooth each row refers to,
- `var`: the name of the variable involved in the smooth,
- `data`: values of `var` at which the derivative was evaluated,
- `derivative`: the estimated derivative,
- `se`: the standard error of the estimated derivative,
- `crit`: the critical value such that `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.

Note

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

Author(s)

Gavin L. Simpson

Examples

```r
load_mgcv()

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

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

## derivatives for a selected smooth
derivatives(mod, type = "central", select = "s(x1)"

## or via a partial match
derivatives(mod, type = "central", select = "x1", partial_match = TRUE)
```
**Description**

Posterior expectations of derivatives from an estimated model

**Usage**

```r
derivative_samples(object, ...) # Default S3 method:
derivative_samples(object, ...)

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

## S3 method for class 'gam'
derivative_samples(
  object,
  focal = NULL,
  data = NULL,
  order = 1L,
  type = c("forward", "backward", "central"),
  scale = c("response", "linear_predictor"),
  method = c("gaussian", "mh", "inla", "user"),
  n = 100,
  eps = 1e-07,
  n_sim = 10000,
  level = 0.95,
  seed = NULL,
  envir = environment(formula(object)),
  draws = NULL,
  mvn_method = c("mvnfast", "mgcv"),
  ...
)
```

**Arguments**

- `object`: an R object to compute derivatives for
- `focal`: character; name of the focal variable. The response derivative of the response with respect to this variable will be returned. All other variables involved in the model will be held at constant values. This can be missing if supplying `data`, in which case, the focal variable will be identified as the one variable that is not constant.
- `...`: arguments passed to other methods and on to `fitted_samples()`
data a data frame containing the values of the model covariates at which to evaluate the first derivatives of the smooths. If supplied, all but one variable must be held at a constant value.

order numeric; the order of derivative.

type character; the type of finite difference used. One of "forward", "backward", or "central".

scale character; should the derivative be estimated on the response or the linear predictor (link) scale? One of "response" (the default), or "linear predictor".

method character; which method should be used to draw samples from the posterior distribution. "gaussian" uses a Gaussian (Laplace) approximation to the posterior. "mh" uses a Metropolis Hastings sample that alternates t proposals with proposals based on a shrunken version of the posterior covariance matrix. "inla" uses a variant of Integrated Nested Laplace Approximation due to Wood (2019), (currently not implemented). "user" allows for user-supplied posterior draws (currently not implemented).

n numeric; the number of points to evaluate the derivative at (if data is not supplied).

eps numeric; the finite difference.

n_sim integer; the number of simulations used in computing the simultaneous intervals.

level numeric; 0 < level < 1; the coverage level of the credible interval. The default is 0.95 for a 95% interval.

seed numeric; a random seed for the simulations.

envir the environment within which to recreate the data used to fit object.

draws matrix; user supplied posterior draws to be used when method = "user".

mvn_method character; one of "mvnfast" or "mgcv". The default is uses mvnfast::rmvn(), which can be considerably faster at generate large numbers of MVN random values than mgcv::rmvn(), but which might not work for some marginal fits, such as those where the covariance matrix is close to singular.

Value

A tibble, currently with the following variables:

- .derivative: the estimated partial derivative,
- additional columns containing the covariate values at which the derivative was evaluated.

Author(s)

Gavin L. Simpson

Examples

load_mgcv()

df <- data_sim("eg1", dist = "negbin", scale = 0.25, seed = 42)

# fit the GAM (note: for execution time reasons using bam())
m <- bam(y ~ s(x0) + s(x1) + s(x2) + s(x3),
  data = df, family = nb(), method = "fREML")

# data slice through data along x2 - all other covariates will be set to
# typical values (value closest to median)
ds <- data_slice(m, x2 = evenly(x2, n = 200))

# samples from posterior of derivatives
fd_samp <- derivative_samples(m,
  data = ds, type = "central",
  focal = "x2", eps = 0.01, seed = 21, n_sim = 100)

# plot the first 20 posterior draws
if (requireNamespace("ggplot2") && requireNamespace("dplyr")) {
  library("ggplot2")
  fd_samp |>
    dplyr::filter(.draw <= 20) |>
    ggplot(aes(x = x2, y = .derivative, group = .draw)) +
    geom_line(alpha = 0.5)
}

difference_smooths

Differences of factor smooth interactions

Description

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

Usage

difference_smooths(model, ...)

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

model  A fitted model.
...
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.

smooth  [Deprecated] Use select instead.

n  numeric; the number of points at which to evaluate the difference between pairs of smooths.

ci_level  numeric between 0 and 1; the coverage of credible interval.

data  data frame of locations at which to evaluate the difference between smooths.

group_means  logical; should the group means be included in the difference?

partial_match  logical; should smooth match partially against smooths? If partial_match = TRUE, smooth must only be a single string, a character vector of length 1. Unlike similar functions, the default here is TRUE because the intention is that users will be matching against factor-by smooth labels.

unconditional  logical; account for smoothness selection in the model?

frequentist  logical; use the frequentist covariance matrix?

Examples

load_mgcv()

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

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

draw(sm_dif)

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

---

draw  Generic plotting via ggplot2

Description

Generic plotting via ggplot2
draw.basis

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

Description

Plots basis functions using ggplot2

Usage

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

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

Value

A patchwork object.

Author(s)

Gavin L. Simpson

Examples

```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")

bf <- basis(m)
draw(bf)

bf <- basis(m, "s(x2)")
draw(bf)
```
**draw.compare_smooths**  
*Plot comparisons of smooths*

**Description**
Plot comparisons of smooths

**Usage**
```r
## 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, 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**
```r
## S3 method for class 'derivatives'
draw(
  object,
  select = NULL,
  scales = c("free", "fixed"),
  add_change = FALSE,
  change_type = c("change", "sizer"),
  alpha = 0.2,
  change_col = "black",
  decrease_col = "#56B4E9",
  increase_col = "#E69F00",
  lwd_change = 1.5,
  ncol = NULL,
  nrow = NULL,
  guides = "keep",
  angle = NULL,
)```
...)

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

Arguments

object a fitted GAM, the result of a call to 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.

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

add_change logical; should the periods of significant change be highlighted on the plot?

change_type character; the type of change to indicate. If "change", no differentiation is made between periods of significant increase or decrease. If "sizer", the periods of increase and decrease are differentiated in the resulting plot.

alpha numeric; alpha transparency for confidence or simultaneous interval.

change_col, decrease_col, increase_col colour specifications to use for indicating periods of change. col_change is used when change_type = "change", while col_decrease and col_increase are used when 'change_type = "sizer"'.

lwd_change numeric; the linewidth to use for the change indicators.

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()

angle numeric; the angle at which the x axis tick labels are to be drawn passed to the angle argument of ggplot2::guide_axis().

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

```r
load_mgcv()

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

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

## fixed axis scales
draw(df, scales = "fixed")
```

---

**draw.difference_smooth**

*Plot differences of smooths*

**Description**

Plot differences of smooths

**Usage**

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

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

ci_alpha     numeric; alpha transparency for confidence or simultaneous interval.


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, select = "s(x2)"
draw(diffs)
```
draw.evaluated_parametric_term

Plot estimated parametric effects

Description

[Deprecated]
Plots estimated univariate and bivariate smooths using ggplot2.

Usage

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

Arguments

object an object, the result of a call to evaluate_parametric_term().

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()`.

**rug**
For `evaluate_parametric_terms()`, a logical to indicate if a rug plot should be drawn.

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

**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.

**Value**
A `ggplot2::ggplot()` object.

**Author(s)**
Gavin L. Simpson

---

**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,
grouped_by = FALSE,
c1_alpha = 0.2,
c1_col = "black",
smooth_col = "black",
resid_col = "steelblue3",
contour_col = "black",
n_contour = NULL,
partial_match = FALSE,
discrete_colour = NULL,
discrete_fill = NULL,
continuous_colour = NULL,
continuous_fill = NULL,
position = "identity",
angle = NULL,
ncol = NULL,
nrow = NULL,
guides = "keep",
widths = NULL,
heights = NULL,
crs = NULL,
default_crs = NULL,
lims_method = "cross",
wrap = TRUE,
envir = environment(formula(object)),
...
)

Arguments

object a fitted GAM, the result of a call to mgcv::gam().
data a optional data frame that may or may not be used? FIXME!
select character, logical, or numeric; which smooths to plot. If NULL, the default, then all model smooths are drawn. Numeric select indexes the smooths in the order they are specified in the formula and stored in object. Character select matches the labels for smooths as shown for example in the output from summary(object). Logical select operates as per numeric select in the order that smooths are stored.
parametric logical; plot parametric terms also? Note that select is used for selecting which smooths to plot. The terms argument is used to select which parametric effects are plotted. The default, as with mgcv::plot.gam(), is to not draw parametric effects.
terms character; which model parametric terms should be drawn? The Default of NULL will plot all parametric terms that can be drawn.
residuals logical; should partial residuals for a smooth be drawn? Ignored for anything but a simple univariate smooth.
<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scales</td>
<td>character; should all univariate smooths be plotted with the same y-axis scale? If scales = &quot;free&quot;, the default, each univariate smooth has its own y-axis scale. If scales = &quot;fixed&quot;, a common y axis scale is used for all univariate smooths. Currently does not affect the y-axis scale of plots of the parametric terms.</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>
<tr>
<td>dist</td>
<td>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.</td>
</tr>
<tr>
<td>rug</td>
<td>logical; draw a rug plot at the bottom of each plot for 1-D smooths or plot locations of data for higher dimensions.</td>
</tr>
<tr>
<td>contour</td>
<td>logical; should contours be draw on the plot using ggplot2::geom_contour().</td>
</tr>
<tr>
<td>grouped_by</td>
<td>logical; should factor by smooths be drawn as one panel per level of the factor (FALSE, the default), or should the individual smooths be combined into a single panel containing all levels (TRUE)?</td>
</tr>
<tr>
<td>ci_alpha</td>
<td>numeric; alpha transparency for confidence or simultaneous interval.</td>
</tr>
<tr>
<td>ci_col</td>
<td>colour specification for the confidence/credible intervals band. Affects the fill of the interval.</td>
</tr>
<tr>
<td>smooth_col</td>
<td>colour specification for the smooth line.</td>
</tr>
<tr>
<td>resid_col</td>
<td>colour specification for the partial residuals.</td>
</tr>
<tr>
<td>contour_col</td>
<td>colour specification for contour lines.</td>
</tr>
<tr>
<td>n_contour</td>
<td>numeric; the number of contour bins. Will result in n_contour - 1 contour lines being drawn. See ggplot2::geom_contour().</td>
</tr>
<tr>
<td>partial_match</td>
<td>logical; should smooths be selected by partial matches with select? If TRUE, select can only be a single string to match against.</td>
</tr>
</tbody>
</table>
discrete_colour
   a suitable colour scale to be used when plotting discrete variables.
discrete_fill
   a suitable fill scale to be used when plotting discrete variables.
continuous_colour
   a suitable colour scale to be used when plotting continuous variables.
continuous_fill
   a suitable fill scale to be used when plotting continuous variables.
position
   Position adjustment, either as a string, or the result of a call to a position adjustment function.
angle
   numeric; the angle at which the x axis tick labels are to be drawn passed to the angle argument of ggplot2::guide_axis().
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()
widths, heights
   The relative widths and heights of each column and row in the grid. Will get repeated to match the dimensions of the grid. If there is more than 1 plot and widths = NULL, the value of widths will be set internally to widths = 1 to accommodate plots of smooths that use a fixed aspect ratio.
crs
   the coordinate reference system (CRS) to use for the plot. All data will be projected into this CRS. See ggplot2::coord_sf() for details.
default_crs
   the coordinate reference system (CRS) to use for the non-sf layers in the plot. If left at the default NULL, the CRS used is 4326 (WGS84), which is appropriate for spline-on-the-sphere smooths, which are parameterized in terms of latitude and longitude as coordinates. See ggplot2::coord_sf() for more details.
lims_method
   character; affects how the axis limits are determined. See ggplot2::coord_sf(). Be careful; in testing of some examples, changing this to "orthogonal" for example with the chlorophyll-a example from Simon Wood’s GAM book quickly used up all the RAM in my test system and the OS killed R. This could be incorrect usage on my part; right now the grid of points at which SOS smooths are evaluated (if not supplied by the user) can produce invalid coordinates for the corners of tiles as the grid is generated for tile centres without respect to the spacing of those tiles.
wrap
   logical; wrap plots as a patchwork? If FALSE, a list of ggplot objects is returned, 1 per term plotted.
envir
   an environment to look up the data within.
...
   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

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("eg2", 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)

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

---

draw.gamlss

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

Description

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

Usage

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

Arguments

- **object**: a model, fitted by `GJRM::gamlss()`
- **scales**: character; should all univariate smooths be plotted with the same y-axis scale? If `scales = "free"`, the default, each univariate smooth has its own y-axis scale. If `scales = "fixed"`, a common y axis scale is used for all univariate smooths. Currently does not affect the y-axis scale of plots of the parametric terms.
- **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()`
- **widths, heights**: The relative widths and heights of each column and row in the grid. Will get repeated to match the dimensions of the grid. If there is more than 1 plot and `widths = NULL`, the value of `widths` will be set internally to `widths = 1` to accommodate plots of smooths that use a fixed aspect ratio.
- **...**: arguments passed to `draw.gam()`

Note

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

Examples

```r
if (require("GJRM", quietly = TRUE)) {
  # follow example from ?GJRM::gamlss
  load_mgcv()
  suppressPackageStartupMessages(library("GJRM"))
  set.seed(0)
  n <- 100
  x1 <- round(runif(n))
  x2 <- runif(n)
  x3 <- runif(n)
  f1 <- function(x) cos(pi * 2 * x) + sin(pi * x)
  y1 <- -1.55 + 2 * x1 + f1(x2) + rnorm(n)
  dataSim <- data.frame(y1, x1, x2, x3)
  eq_mu <- y1 ~ x1 + s(x2)
  eq_s <- ~ s(x3, k = 6)
  fl <- list(eq_mu, eq_s)
  m <- gamlss(fl, data = dataSim)
  draw(m)
}
```

---

**draw.mgcv_smooth**

*Plot basis functions*

Description

Plots basis functions using ggplot2
Usage

```r
## S3 method for class 'mgcv_smooth'
draw(
  object,
  legend = FALSE,
  use_facets = TRUE,
  labeller = NULL,
  xlab,
  ylab,
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  angle = NULL,
  ...
)
```

Arguments

- **object**: an object, the result of a call to `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()`.
- **angle**: numeric; the angle at which the x axis tick labels are to be drawn passed to the angle argument of `ggplot2::guide_axis()`.
- **...**: 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 <- data_sim("eg4", n = 400, seed = 42)

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

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

draw.pairwise_concurvity

Plot concurvity measures

Description

Plot concurvity measures

Usage

```r
## S3 method for class 'pairwise_concurvity'
draw(
  object,
  title = "Smooth-wise concurvity",
  subtitle = NULL,
  caption = NULL,
  x_lab = "Term",
  y_lab = "With",
  fill_lab = "Concurvity",
  continuous_colour = NULL,
  ...
)

## S3 method for class 'overall_concurvity'
draw(
  object,
  title = "Overall concurvity",
  subtitle = NULL,
  caption = NULL,
  y_lab = "Concurvity",
  x_lab = NULL,
  bar_col = "steelblue",
  bar_fill = "steelblue",
  ...
)
```
Arguments

- **object**: An object inheriting from class "concurvity", usually the result of a call to `model_concurvity()` or its abbreviated form `concrvity()`.
- **title**: character; the plot title.
- **subtitle**: character; the plot subtitle.
- **caption**: character; the plot caption
- **x_lab**: character; the label for the x axis.
- **y_lab**: character; the label for the y axis.
- **fill_lab**: character; the label to use for the fill guide.
- **continuous_colours**: function; continuous colour (fill) scale to use.
- **...**: arguments passed to other methods.
- **bar_col**: colour specification for the bar colour.
- **bar_fill**: colour specification for the bar fill

---

**Description**

Plot estimated effects for model parametric terms

**Usage**

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

```
Arguments

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

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

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 for 1-D smooths or plot locations of data for higher dimensions.

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

angle
- numeric; the angle at which the x axis tick labels are to be drawn passed to the angle argument of `ggplot2::guide_axis()`.

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

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()`

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,
  as_matrix = TRUE,
  continuous_fill = NULL,
)```
Arguments

object a fitted GAM, the result of a call to \texttt{mgcv::gam}.
normalize logical; normalize the penalty to the range -1, 1?
as_matrix logical; how should the plotted penalty matrix be oriented? If TRUE row 1, column 1 of the penalty matrix is drawn in the upper left, whereas, if FALSE it is drawn in the lower left of the plot.
continuous_fill a suitable fill scale to be used when plotting continuous variables.
xlab character or expression; the label for the x axis. If not supplied, no axis label will be drawn. May be a vector, one per penalty.
ylab character or expression; the label for the y axis. If not supplied, no axis label will be drawn. May be a vector, one per penalty.
title character or expression; the title for the plot. See \texttt{ggplot2::labs}. May be a vector, one per penalty.
subtitle character or expression; the subtitle for the plot. See \texttt{ggplot2::labs}. May be a vector, one per penalty.
caption character or expression; the plot caption. See \texttt{ggplot2::labs}. May be a vector, one per penalty.
ncol, nrow numeric; the numbers of rows and columns over which to spread the plots.
guides character; one of "keep" (the default), "collect", or "auto". Passed to \texttt{patchwork::plot_layout}.
... additional arguments passed to \texttt{patchwork::wrap_plots}.

Examples

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

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

# for a specific smooth
draw(penalty(m, select = "s(x2):fac1"))
\end{verbatim}
Description

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

Usage

```r
## S3 method for class 'rootogram'
draw(
  object,
  type = c("hanging", "standing", "suspended"),
  sqrt = TRUE,
  ref_line = TRUE,
  warn_limits = TRUE,
  fitted_colour = "steelblue",
  bar_colour = NA,
  bar_fill = "grey",
  ref_line_colour = "black",
  warn_line_colour = "black",
  ylab = NULL,
  xlab = NULL,
  ...
)
```

Arguments

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

Value

A `ggplot` object.

References


See Also

`rootogram()` to compute the data for the rootogram.

Examples

```r
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 = poissson(), data = df, method = "REML")
rg <- rootogram(m)

# plot the rootogram
draw(rg)

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

---

draw.smooth_estimates  Plot the result of a call to smooth_estimates()

Description

Plot the result of a call to smooth_estimates()

Usage

```r
## S3 method for class 'smooth_estimates'
draw(
  object,
  constant = NULL,
  fun = NULL,
  contour = TRUE,
  grouped_by = FALSE,
  contour_col = "black",
  n_contour = NULL,
  ci_alpha = 0.2,
  ci_col = "black",
```
smooth_col = "black",
resid_col = "steelblue3",
decrease_col = "#56B4E9",
increase_col = "#E69F00",
change_lwd = 1.75,
partial_match = FALSE,
discrete_colour = NULL,
discrete_fill = NULL,
continuous_colour = NULL,
continuous_fill = NULL,
angle = NULL,
ylim = NULL,
crs = NULL,
default_crs = NULL,
lims_method = "cross",
...
)

Arguments

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

constant
numeric; a constant to add to the estimated values of the smooth. constant, if supplied, will be added to the estimated value before the confidence band is computed.

fun
function; a function that will be applied to the estimated values and confidence interval before plotting. Can be a function or the name of a function. Function fun will be applied after adding any constant, if provided.

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

grouped_by
logical; should factor by smooths be drawn as one panel per level of the factor (FALSE, the default), or should the individual smooths be combined into a single panel containing all levels (TRUE)?

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 smooth line.

resid_col
colour specification for the partial residuals.

decrease_col, increase_col
colour specifications to use for indicating periods of change. col_change is used when change_type = "change", while col_decrease and col_increase are used when 'change_type = "sizer"'.

change_lwd
numeric; the value to set the linewidth to in `ggplot2::geom_line()`, used to represent the periods of change.
draw.smooth_estimates

- 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: a suitable colour scale to be used when plotting discrete variables.
- discrete_fill: a suitable fill scale to be used when plotting discrete variables.
- continuous_colour: a suitable colour scale to be used when plotting continuous variables.
- continuous_fill: a suitable fill scale to be used when plotting continuous variables.
- angle: numeric; the angle at which the x axis tick labels are to be drawn passed to the angle argument of `ggplot2::guide_axis()`.
- ylim: numeric; vector of y axis limits to use all all panels drawn.
- crs: the coordinate reference system (CRS) to use for the plot. All data will be projected into this CRS. See `ggplot2::coord_sf()` for details.
- default_crs: the coordinate reference system (CRS) to use for the non-sf layers in the plot. If left at the default NULL, the CRS used is 4326 (WGS84), which is appropriate for spline-on-the-sphere smooths, which are parameterized in terms of latitude and longitude as coordinates. See `ggplot2::coord_sf()` for more details.
- lims_method: character; affects how the axis limits are determined. See `ggplot2::coord_sf()`.

Examples

```r
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, select = "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])

# Adding change indicators to smooths based on derivatives of the smooth
d <- derivatives(m, n = 100) # n to match smooth_estimates()

smooth_estimates(m) |>
    add_sizer(derivatives = d, type = "sizer") |>
    draw()

draw.smooth_samples  Plot posterior smooths

Description

Plot posterior smooths

Usage

## S3 method for class 'smooth_samples'

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

Arguments

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

- **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? If scales = "free", the default, each univariate smooth has its own y-axis scale. If scales = "fixed", a common y axis scale is used for all univariate smooths. Currently does not affect the y-axis scale of plots of the parametric terms.

- **rug** logical; draw a rug plot at the bottom of each plot for 1-D smooths or plot locations of data for higher dimensions.

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

- **angle** numeric; the angle at which the x axis tick labels are to be drawn passed to the angle argument of `ggplot2::guide_axis()`.

- **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()` 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
sm1 <- smooth_samples(m1, n = 15, seed = 23478)
## plot
draw(sm1, alpha = 0.7)
## plot only 5 randomly sampled draws
draw(sm1, n_samples = 5, alpha = 0.7)

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

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

---

### edf

**Effective degrees of freedom for smooths and GAMs**

#### Description

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

#### Usage

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

## S3 method for class 'gam'
edf(
  object,
  select = NULL,
  smooth = deprecated(),
```

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

Arguments

- **object**: a fitted model from which to extract smooth-specific EDFs.
- **...**: arguments passed to 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.
- **smooth** [Deprecated] Use select instead. extracted. If NULL, the default, EDFs for all smooths will be returned.
- **type**: character: which type of EDF to return. "default" returns the standard EDF; "unconditional" selects the EDF corrected for smoothness parameter selection, if available; "alternative" returns the alternative formulation for EDF from Wood (2017, pp. 252)
- **partial_match**: logical; should smooths be selected by partial matches with select? If TRUE, select can only be a single string to match against.

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

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

where 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 relevent elements of 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
```
evaluate_parametric_term

Evaluate parametric model terms

Description

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

Usage

evaluate_parametric_term(object, ...)

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

Arguments

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

Evaluate a smooth

Description

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

Usage

evaluate_smooth(object, ...)

Arguments

object an object of class "gam" or "gamm".

... arguments passed to other methods.

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

Value

A data frame, which is of class "evaluated_1d_smooth" or evaluated_2d_smooth, which inherit from classes "evaluated_smooth" and "data.frame".

eval_smooth S3 methods to evaluate individual smooths

Description

S3 methods to evaluate individual smooths
Usage

```r
eval_smooth(smooth, ...)
```

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

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

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

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

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

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

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

Arguments

smooth currently an object that inherits from class 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, 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.
`evenly`  

Create a sequence of evenly-spaced values

Description

For a continuous vector \( x \), `evenly` and `seq_min_max()` create a sequence of \( n \) evenly-spaced values over the range lower – upper. By default, lower is defined as \( \min(x) \) and upper as \( \max(x) \), excluding NAs. For a factor \( x \), the function returns `levels(x)`.

Usage

```r
# Create a sequence of 100 evenly-spaced values over the range of the vector x
evenly(x, n = 100, by = NULL, lower = NULL, upper = NULL)

# Create a sequence of 10 evenly-spaced values over the range of the vector x
evenly(x, n = 10, by = 0.2)

# And so on...
```

Arguments

- `x`: numeric; vector over which evenly-spaced values are returned
- `n`: numeric; the number of evenly-spaced values to return. A default of 100 is used for convenience as that what is typically used when evaluating a smooth.
- `by`: numeric; the increment of the sequence. If specified, argument \( n \) is ignored and the sequence returned will be from \( \min(x) \) to \( \max(x) \) in increments of \( \text{by} \).
- `lower`: numeric; the lower bound of the interval.
- `upper`: numeric; the upper bound of the interval.

Value

A numeric vector of length \( n \).

See Also

See `base::seq()` for details of the behaviour of `evenly()` when using `by`.

Examples

```r
x <- rnorm(10)
n <- 10L

# 10 values evenly over the range of `x`
evenly(x, n = n)

# evenly spaced values, incrementing by 0.2
evenly(x, by = 0.2)

# evenly spaced values, incrementing by 0.2, starting at -2
evenly(x, by = 0.2, lower = -2)
```
factor_combos

All combinations of factor levels

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, ...)
family_name

Arguments

object  a fitted model. Models fitted by \texttt{mgcv::gam()}, \texttt{mgcv::bam()}, \texttt{mgcv::gamm()}, and \texttt{gamm4::gamm4()} are currently supported.

... arguments passed to other methods.

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 \texttt{family()} objects and anything with a \texttt{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 or a simple Metropolis Hastings sampler.

**Usage**

```r
defitted_samples(model, ...)
```

```r
## S3 method for class 'gam'
defitted_samples(model,
    n = 1,
    data = newdata,
    seed = NULL,
    scale = c("response", "linear_predictor"),
    method = c("gaussian", "mh", "inla", "user"),
    n_cores = 1,
    burnin = 1000,
    thin = 1,
    t_df = 40,
    rw_scale = 0.25,
    freq = FALSE,
    unconditional = FALSE,
    draws = NULL,
    mvn_method = c("mvnfast", "mgcv"),
    
    newdata = NULL,
    ncores = NULL
)
```

**Arguments**

- `model` - a fitted model of the supported types
- `...` - arguments passed to other methods. For `fitted_samples()`, these are passed on to `mgcv::predict.gam()`. For `posterior_samples()` these are passed on to `fitted_samples()`. For `predicted_samples()` these are passed on to the relevant `simulate()` method.
- `n` - numeric; the number of posterior samples to return.
- `data` - data frame; new observations at which the posterior draws from the model should be evaluated. If not supplied, the data used to fit the model will be used for `data`, if available in `model`.
- `seed` - numeric; a random seed for the simulations.
scale character; what scale should the fitted values be returned on? "linear predictor" is a synonym for "link" if you prefer that terminology.

method character; which method should be used to draw samples from the posterior distribution. "gaussian" uses a Gaussian (Laplace) approximation to the posterior. "mh" uses a Metropolis Hastings sampler that alternates t proposals with proposals based on a shrunken version of the posterior covariance matrix. "inla" uses a variant of Integrated Nested Laplace Approximation due to Wood (2019), (currently not implemented). "user" allows for user-supplied posterior draws (currently not implemented).

n_cores 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).

burnin numeric; number of samples to discard as the burnin draws. Only used with method = "mh".

thin numeric; the number of samples to skip when taking n draws. Results in thin * n draws from the posterior being taken. Only used with method = "mh".

t_df numeric; degrees of freedom for t distribution proposals. Only used with method = "mh".

rw_scale numeric; Factor by which to scale posterior covariance matrix when generating random walk proposals. Negative or non finite to skip the random walk step. Only used with method = "mh".

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.

draws matrix; user supplied posterior draws to be used when method = "user".

mvn_method character; one of "mvnfast" or "mgcv". The default is uses `mvnfast::rmvn()`, which can be considerably faster at generate large numbers of MVN random values than `mgcv::rmvn()`, but which might not work for some marginal fits, such as those where the covariance matrix is close to singular.

newdata Deprecated: use data instead.

ncores Deprecated; use n_cores instead. The 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 `data` that each posterior draw relates to,
- `draw` (integer) an index, in range 1:n, indicating which draw each row relates to,
- `response` (numeric) the predicted response for the indicated row of `data`. 
Note

Models with offset terms supplied via the offset argument to \texttt{mgcv::gam()} etc. are ignored by \texttt{mgcv::predict.gam()}. As such, this kind of offset term is also ignored by \texttt{posterior_samples()}. Offset terms that are included in the model formula supplied to \texttt{mgcv::gam()} etc are not ignored and the posterior samples produced will reflect those offset term values. This has the side effect of requiring any new data values provided to \texttt{posterior_samples()} via the \texttt{data} argument must include the offset variable.

Author(s)

Gavin L. Simpson

References


Examples

```r
load_mgcv()

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

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

fs
```

```r
# can generate own set of draws and use them
drws <- generate_draws(m1, n = 2, seed = 24)
fs2 <- fitted_samples(m1, method = "user", draws = drws)

fs2
```

---

**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,
  ...
)

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

## S3 method for class 'scam'
fitted_values(object, ...)

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.

Models fitted with certain families will include additional variables

- `mgcv::ocat()` models: when `scale = "response"`, the returned object will contain a row column and a category column, which indicate to which row of the data each row of the returned object belongs. Additionally, there will be `nrow(data) * n_categories` rows in the returned object; each row is the predicted probability for a single category of the response.
Note

For most families, 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. An exception is the mgcv::ocat() family, for which the se is on the response scale if scale = "response".

Examples

```r
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
```

**Note**

For most families, 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. An exception is the mgcv::ocat() family, for which the se is on the response scale if scale = "response".

**Examples**

```r
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
```

**Description**

Extract fixed effects estimates

**Arguments**

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

**Usage**

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

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

## S3 method for class 'lm'
fixef(object, ...)
```
## S3 method for class 'glm'
fixef(object, ...)

fixed_effects(object, ...)

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

### Arguments

- **object**: a fitted GAM
- **...**: arguments passed to other methods

### Examples

```r
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

```r
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

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

---

### Description

Posterior samples using a simple Metropolis Hastings sampler

### Usage

```r
gaussian_draws(model, ...)
```

## S3 method for class 'gam'
```r
gaussian_draws(
  model,
  n,
  n_cores = 1L,
  index = NULL,
  frequentist = FALSE,
  unconditional = FALSE,
  mvn_method = "mvnfast",
  ...
)
```

## S3 method for class 'scam'
```r
gaussian_draws(
  model,
  n,
  n_cores = 1L,
  index = NULL,
  frequentist = FALSE,
  parametrized = TRUE,
  mvn_method = "mvnfast",
  ...
)
```
Arguments

model  a fitted R model. Currently only models fitted by mgcv::gam() or mgcv::bam(), or return an object that `inherits` from such objects are supported. Here, "inherits" is used in a loose fashion; models fitted by scam::scam() are support even though those models don't strictly inherit from class "gam" as far as `inherits()` is concerned.

...  arguments passed to methods.

n_cores  integer; number of CPU cores to use when generating multivariate normal distributed random values. Only used if `mvn_method = "mvnfast"` and `method = "gaussian"`.

index  numeric; vector of indices of coefficients to use. Can be used to subset the mean vector and covariance matrix extracted from model.

frequentist  logical; if TRUE, the frequentist covariance matrix of the parameter estimates is used. If FALSE, the Bayesian posterior covariance matrix of the parameters is used. See mgcv::vcov.gam().

unconditional  logical; if TRUE the Bayesian smoothing parameter uncertainty corrected covariance matrix is used, if available for model. See mgcv::vcov.gam().

mvn_method  character; one of "mvnfast" or "mgcv". The default is uses `mvnfast::rmvn()`, which can be considerably faster at generate large numbers of MVN random values than `mgcv::rmvn()`, but which might not work for some marginal fits, such as those where the covariance matrix is close to singular.

parametrized  logical; use parametrized coefficients and covariance matrix, which respect the linear inequality constraints of the model. Only for scam::scam() model fits.

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**

```r
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)
```

```r
## S3 method for class 'gam'
get_smooths_by_id(object, id)
```

```r
## S3 method for class 'scam'
get_smooths_by_id(object, id)
```

```r
## S3 method for class 'gamm'
get_smooths_by_id(object, id)
```

```r
## S3 method for class 'gamm4'
get_smooths_by_id(object, id)
```

```r
## S3 method for class 'list'
get_smooths_by_id(object, id)
```
Arguments

- **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+.

**gw_f0**  
*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)
- **...**: arguments passed to other methods, ignored.
Examples

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

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.

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)
```
is_by_smooth  

Tests for by variablesmooths

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

Usage
is_by_smooth(smooth)
is_factor_by_smooth(smooth)
is_continuous_by_smooth(smooth)
by_variable(smooth)
by_level(smooth)

Arguments
smooth an object of class "mgcv.smooth"

Value
A logical vector.

Author(s)
Gavin L. Simpson

is_factor_term  

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.
is_mgcv_smooth

Usage

is_factor_term(object, term, ...)

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

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

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

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

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

Arguments

object an R object on which method dispatch is performed
term character; the name of a model term, in the sense of attr(terms(object),
"term.labels"). Currently not checked to see if the term exists in the model.
... arguments passed to other methods.

Value

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

is_mgcv_smooth

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)

stop_if_not_mgcv_smooth(smooth)

check_is_mgcv_smooth(smooth)

is_mrf_smooth(smooth)
Arguments

smooth an R object, typically a list

Details

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

Description

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

Usage

is_offset(terms)

Arguments

terms character vector of model terms.

Value

A logical vector of the same length as terms.

Author(s)

Gavin L. Simpson

Examples

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

```r
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, ...)
```

```r
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 'glm'
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 "phi" for the scale parameter of `mgcv::gammals()` & `mgcv::twlss()`.
- `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.

---

**lp_matrix**  
*Return the linear prediction matrix of a fitted GAM*

**Description**

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

**Usage**

```r
lp_matrix(model, ...)
```

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

**Arguments**

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

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

Value

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

Examples

```r
load_mgcv()

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

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

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

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

Description

Posterior samples using a Gaussian approximation to the posterior distribution
Usage

 mh_draws(model, ...)

## S3 method for class 'gam'
 mh_draws(
   model,
   n,
   burnin = 1000,
   thin = 1,
   t_df = 40,
   rw_scale = 0.25,
   index = NULL,
   ...
)

Arguments

model a fitted R model. Currently only models fitted by mgcv::gam() or mgcv::bam(), or return an object that inherits from such objects are supported. Here, "inherits" is used in a loose fashion; models fitted by scam::scam() are support even though those models don't strictly inherit from class "gam" as far as inherits() is concerned.

... arguments passed to methods.
n numeric; the number of posterior draws to take.
burnin numeric; the length of any initial burn in period to discard. See mgcv::gam.mh().
thin numeric; retain only thin samples. See mgcv::gam.mh().
t_df numeric; degrees of freedom for static multivariate t proposal. See mgcv::gam.mh().
rw_scale numeric; factor by which to scale posterior covariance matrix when generating random walk proposals. See mgcv::gam.mh().
index numeric; vector of indices of coefficients to use. Can be used to subset the mean vector and covariance matrix extracted from model.

---

model_concurvity Concurvity of an estimated GAM

Description

Concurvity of an estimated GAM

Usage

 model_concurvity(model, ...)

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

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

**Arguments**

- **model**: a fitted GAM. Currently only objects of class "gam" are supported
- **terms**: currently ignored
- **type**: character;
- **pairwise**: logical; extract pairwise concurvity of model terms?

**Examples**

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

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

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

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

Extract the model constant term

Description

Extracts the model constant term, the model intercept, from a fitted model object.

Usage

model_constant(model)

Arguments

model

a fitted model for which a coef() method exists

Examples

load_mgcv()

# simulate a small example
df <- data_sim("eg1", seed = 42)

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

# extract the estimate of the constant term
model_constant(m)
# same as coef(m)[1L]
coef(m)[1L]

model_vars

List the variables involved in a model fitted with a formula

Description

List the variables involved in a model fitted with a formula

Usage

model_vars(model, ...)

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

## Default S3 method:

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

# Default S3 method:
Arguments

model  
a fitted model object with a $pred.formula, $terms component or a "terms" attribute

...  
Arguments passed to other methods. Currently ignored.

Examples

load_mgcv()

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

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

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

---

nb_theta  

*Negative binomial parameter theta*

Description

Negative binomial parameter theta

Usage

nb_theta(model)

## S3 method for class 'gam'

nb_theta(model)
null_deviance

Arguments

model a fitted model.

Value

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

Methods (by class)

- nb_theta(gam): Method for class "gam"

Examples

load_mgcv()

# Generate example data
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")

# Extract null deviance
nb_theta(m)

null_deviance

Extract the null deviance of a fitted model

Description

Extract the null deviance of a fitted model

Usage

null_deviance(model, ...)

Arguments

model a fitted model

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

How many smooths in a fitted model

**Usage**

\[ n\_smooths(object) \]

### Default S3 method:

\[ n\_smooths(object) \]

### S3 method for class 'gam'

\[ n\_smooths(object) \]

### S3 method for class 'gamm'

\[ n\_smooths(object) \]

### S3 method for class 'bam'

\[ n\_smooths(object) \]

**Arguments**

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

**Description**

Plot of fitted against observed response values

**Usage**

\[ observed\_fitted\_plot( 
  model, 
  ylab = NULL, 
  xlab = NULL, 
  title = NULL, 
  subtitle = NULL, 
  caption = NULL, 
  point_col = "black", 
  point_alpha = 1 
) \]
Overview

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

Description

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

Usage

overview(model, ...)

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

Arguments

model a fitted model object to overview.
...
parametric logical; include the model parametric terms in the overview?
parametric_effects

random_effects  tests of fully penalized smooth terms (those with a zero-dimensional null space, e.g., random effects) are computationally expensive and for large data sets producing these p values can take a very long time. If random_effects = FALSE, the tests of the expensive terms will be skipped.

dispersion  numeric; a known value for the dispersion parameter. The default NULL implies that the estimated value or the default value (1 for the Poisson distribution for example) where this is specified is used instead.

frequentist  logical; by default the Bayesian estimated covariance matrix of the parameter estimates is used to calculate p values for parametric terms. If frequentist = FALSE, the frequentist covariance matrix of the parameter estimates is used.

accuracy  numeric; accuracy with which to report p values, with p values below this value displayed as "< accuracy".

stars  logical; should significance stars be added to the output?

Examples

load_mgcv()

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

parametric_effects  Estimated values for parametric model terms

Description

Estimated values for parametric model terms

Usage

parametric_effects(object, ...)

## S3 method for class 'gam'
parametric_effects(
   object,
   terms = NULL,
   data = NULL,
   unconditional = FALSE,
   unnest = TRUE,
   ci_level = 0.95,
   envir = environment(formula(object)),
   transform = FALSE,
   ...
)
)
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.
- **data**: a optional data frame that may or may not be used? FIXME!
- **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 parametric effect objects?
- **ci_level**: numeric; the coverage required for the confidence interval. Currently ignored.
- **envir**: an environment to look up the data within.
- **transform**: logical; if TRUE, the parametric effect will be plotted on its transformed scale which will result in the effect being a straight line. If FALSE, the effect will be plotted against the raw data (i.e. for log10(x), or poly(z), the x-axis of the plot will be x or z respectively.)

Description

Names of any parametric terms in a GAM

Usage

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

Arguments

- **model**: a fitted model.
- **...**: arguments passed to other methods.
partial_derivatives

Partial derivatives of estimated multivariate smooths via finite differences

Description

Partial derivatives of estimated multivariate smooths via finite differences

Usage

partial_derivatives(object, ...)

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

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

## S3 method for class 'gam'
partial_derivatives(
  object, 
  select = NULL, 
  term = deprecated(), 
  focal = NULL, 
  data = newdata, 
  order = 1L, 
  type = c("forward", "backward", "central"), 
  n = 100, 
  eps = 1e-07, 
  interval = c("confidence", "simultaneous"), 
  n_sim = 10000, 
  level = 0.95, 
  unconditional = FALSE, 
  frequentist = FALSE, 
  offset = NULL, 
  ncores = 1, 
  partial_match = FALSE, 
  seed = NULL, 
  ..., 
  newdata = NULL
)

Arguments

object an R object to compute derivatives for.
...
arguments passed to other methods.
partial_derivatives

select 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.

term [Deprecated] Use select instead.

term character; name of the focal variable. The partial derivative of the estimated smooth with respect to this variable will be returned. All other variables involved in the smooth will be held at constant. This can be missing if supplying data, in which case, the focal variable will be identified as the one variable that is not constant.
data a data frame containing the values of the model covariates at which to evaluate the first derivatives of the smooths. If supplied, all but one variable must be held at a constant value.

order numeric; the order of derivative.
type character; the type of finite difference used. One of "forward", "backward", or "central".
n numeric; the number of points to evaluate the derivative at.
eps numeric; the finite difference.
interval character; the type of interval to compute. One of "confidence" for point-wise intervals, or "simultaneous" for simultaneous intervals.
n_sim integer; the number of simulations used in computing the simultaneous intervals.
level numeric; \(0 < \text{level} < 1\); the confidence level of the point-wise or simultaneous interval. The default is \(0.95\) for a 95\% interval.

unconditional logical; use smoothness selection-corrected Bayesian covariance matrix?
frequentist logical; use the frequentist covariance matrix?
offset numeric; a value to use for any offset term

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

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

seed numeric; RNG seed to use.

newdata Deprecated: use data instead.

Value

A tibble, currently with the following variables:

- `smooth`: the smooth each row refers to,
- `partial_deriv`: the estimated partial derivative,
- `se`: the standard error of the estimated partial derivative,
- `crit`: the critical value such that `derivative \pm (crit \times se)` gives the upper and lower bounds of the requested confidence or simultaneous interval (given level),
- `lower_ci`: the lower bound of the confidence or simultaneous interval,
- `upper_ci`: the upper bound of the confidence or simultaneous interval.
partial_derivatives

Note

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

Author(s)

Gavin L. Simpson

Examples

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

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

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

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

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

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

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

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

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

# draw all three panels
partial_residuals

p1 + p2 + p3 + plot_layout(ncol = 3)

partial_residuals  Partial residuals

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
partial_residuals(m, select = "s(x2)")

## or with partial matching
penalty

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

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

- **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.
data: data frame; a data frame of values for terms mentioned in the smooth specification.

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().

select: character, logical, or numeric; which smooths to extract penalties for. If NULL, the default, then penalties for 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.

smooth: [Deprecated] Use select instead.

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

margins: logical; extract the penalty matrices for the tensor product or the marginal smooths of the tensor product?

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.

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

Author(s):

Gavin L. Simpson
Examples

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

# penalties for all smooths
penalty(m)

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

---

### posterior_samples

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

#### Description

Draw samples from the posterior distribution of an estimated model

#### Usage

```
posterior_samples(model, ...)
```

## S3 method for class 'gam'

```
posterior_samples(
  model,
  n = 1,
  data = newdata,
  seed = NULL,
  method = c("gaussian", "mh", "inla", "user"),
  n_cores = 1,
  burnin = 1000,
  thin = 1,
  t_df = 40,
  rw_scale = 0.25,
  freq = FALSE,
  unconditional = FALSE,
  weights = NULL,
  draws = NULL,
  mvn_method = c("mvnfast", "mgcv"),
  ...
)
```

```
newdata = NULL,
ncores = NULL
```
Arguments

model  a fitted model of the supported types
...  arguments passed to other methods. For fitted_samples(), these are passed on to mgcv::predict.gam(). For posterior_samples() these are passed on to fitted_samples(). For predicted_samples() these are passed on to the relevant simulate() method.

n  numeric; the number of posterior samples to return.

data  data frame; new observations at which the posterior draws from the model should be evaluated. If not supplied, the data used to fit the model will be used for data, if available in model.

seed  numeric; a random seed for the simulations.

method  character; which method should be used to draw samples from the posterior distribution. "gaussian" uses a Gaussian (Laplace) approximation to the posterior. "mh" uses a Metropolis Hastings sampler that alternates t proposals with proposals based on a shrunken version of the posterior covariance matrix. "inla" uses a variant of Integrated Nested Laplace Approximation due to Wood (2019), (currently not implemented). "user" allows for user-supplied posterior draws (currently not implemented).

n_cores  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).

burnin  numeric; number of samples to discard as the burnin draws. Only used with method = "mh".

thin  numeric; the number of samples to skip when taking n draws. Results in thin * n draws from the posterior being taken. Only used with method = "mh".

t_df  numeric; degrees of freedom for t distribution proposals. Only used with method = "mh".

rw_scale  numeric; Factor by which to scale posterior covariance matrix when generating random walk proposals. Negative or non finite to skip the random walk step. Only used with method = "mh".

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 data is null then defaults to object[["prior.weights"]], otherwise a vector of ones.

draws  matrix; user supplied posterior draws to be used when method = "user".

mvn_method  character; one of "mvnfast" or "mgcv". The default is uses mvnfast::rmvn(), which can be considerably faster at generate large numbers of MVN random values than mgcv::rmvn(), but which might not work for some marginal fits, such as those where the covariance matrix is close to singular.

newdata  Deprecated: use data instead.
ncores  Deprecated; use n_cores instead. The 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 data that each posterior draw relates to,
- **draw** (integer) an index, in range 1:n, indicating which draw each row relates to,
- **response** (numeric) the predicted response for the indicated row of data.

Note

Models with offset terms supplied via the offset argument to `mgcv::gam()` etc. are ignored by `mgcv::predict.gam()` As such, this kind of offset term is also ignored by `posterior_samples()`. Offset terms that are included in the model formula supplied to `mgcv::gam()` etc are not ignored and the posterior samples produced will reflect those offset term values. This has the side effect of requiring any new data values provided to `posterior_samples()` via the data argument must include the offset variable.

Author(s)

Gavin L. Simpson

References


post_draws  Low-level Functions to generate draws from the posterior distribution of model coefficients

Description

Low-level Functions to generate draws from the posterior distribution of model coefficients

Generate posterior draws from a fitted model
Usage

post_draws(model, ...)  # Default S3 method:
post_draws(  
  model,  
  n,  
  method = c("gaussian", "mh", "inla", "user"),  
  mu = NULL,  
  sigma = NULL,  
  n_cores = 1L,  
  burnin = 1000,  
  thin = 1,  
  t_df = 40,  
  rw_scale = 0.25,  
  index = NULL,  
  frequentist = FALSE,  
  unconditional = FALSE,  
  parametrized = TRUE,  
  mvn_method = c("mvnfast", "mgcv"),  
  draws = NULL,  
  seed = NULL,  
  ...  
)

generate_draws(model, ...)  # S3 method for class 'gam'
generate_draws(  
  model,  
  n,  
  method = c("gaussian", "mh", "inla"),  
  mu = NULL,  
  sigma = NULL,  
  n_cores = 1L,  
  burnin = 1000,  
  thin = 1,  
  t_df = 40,  
  rw_scale = 0.25,  
  index = NULL,  
  frequentist = FALSE,  
  unconditional = FALSE,  
  mvn_method = c("mvnfast", "mgcv"),  
  seed = NULL,  
  ...  
)

Arguments

model a fitted R model. Currently only models fitted by mgcv::gam() or mgcv::bam(), or return an object that inherits from such objects are supported. Here, "inherits" is used in a loose fashion; models fitted by scam::scam() are support even though those models don’t strictly inherit from class “gam” as far as inherits() is concerned.

... arguments passed to methods.

n numeric; the number of posterior draws to take.

method character; which algorithm to use to sample from the posterior. Currently implemented methods are: “gaussian” and "mh". "gaussian" calls gaussian_draws() which uses a Gaussian approximation to the posterior distribution. "mh" uses a simple Metropolis Hasting sampler which alternates static proposals based on a Gaussian approximation to the posterior, with random walk proposals. Note, setting t_df to a low value will result in heavier-tailed statistic proposals. See mgcv::gam.mh() for more details.

mu numeric; user-supplied mean vector (vector of model coefficients). Currently ignored.

sigma matrix; user-supplied covariance matrix for mu. Currently ignored.

n_cores integer; number of CPU cores to use when generating multivariate normal distributed random values. Only used if mvn_method = "mvnfast" and method = "gaussian".

burnin numeric; the length of any initial burn in period to discard. See mgcv::gam.mh().

thin numeric; retain only thin samples. See mgcv::gam.mh().

t_df numeric; degrees of freedom for static multivariate t proposal. See mgcv::gam.mh().

rw_scale numeric; factor by which to scale posterior covariance matrix when generating random walk proposals. See mgcv::gam.mh().

index numeric; vector of indices of coefficients to use. Can be used to subset the mean vector and covariance matrix extracted from model.

frequentist logical; if TRUE, the frequentist covariance matrix of the parameter estimates is used. If FALSE, the Bayesian posterior covariance matrix of the parameters is used. See mgcv::vcov.gam().

unconditional logical; if TRUE the Bayesian smoothing parameter uncertainty corrected covariance matrix is used, if available for model. See mgcv::vcov.gam().

parametrized logical; use parametrized coefficients and covariance matrix, which respect the linear inequality constraints of the model. Only for scam::scam() model fits.

mvn_method character; one of "mvnfast" or "mgcv". The default is uses mvnfast::rmvn(), which can be considerably faster at generate large numbers of MVN random values than mgcv::rmvn(), but which might not work for some marginal fits, such as those where the covariance matrix is close to singular.

draws matrix; user supplied posterior draws to be used when method = "user".

seed numeric; the random seed to use. If NULL, a random seed will be generated without affecting the current state of R’s RNG.
predicted_samples   Draw new response values from the conditional distribution of the response

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,
  data = newdata,
  seed = NULL,
  weights = NULL,
  ...,  
  newdata = NULL
)
```

Arguments

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

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

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

Author(s)

Gavin L. Simpson

Examples

```r
load_mgcv()

dat <- data_sim("eg1", n = 1000, dist = "normal", scale = 2, seed = 2)
m <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = dat, method = "REML")
predicted_samples(m, n = 5, seed = 42)

## Can pass arguments to predict.gam()

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

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

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

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

Description

Quantile-quantile plot of model residuals
Usage

qq_plot(model, ...)

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

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

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

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

Arguments

model a fitted model. Currently only class “gam”.
...
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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ylab</td>
<td>character or expression; the label for the y axis. If not supplied, a suitable label will be generated.</td>
</tr>
<tr>
<td>xlab</td>
<td>character or expression; the label for the y axis. If not supplied, a suitable label will be generated.</td>
</tr>
<tr>
<td>title</td>
<td>character or expression; the title for the plot. See <code>ggplot2::labs()</code>. May be a vector, one per penalty.</td>
</tr>
<tr>
<td>subtitle</td>
<td>character or expression; the subtitle for the plot. See <code>ggplot2::labs()</code>. May be a vector, one per penalty.</td>
</tr>
<tr>
<td>caption</td>
<td>character or expression; the plot caption. See <code>ggplot2::labs()</code>. May be a vector, one per penalty.</td>
</tr>
<tr>
<td>ci_col</td>
<td>fill colour for the reference interval when <code>method = &quot;simulate&quot;</code>.</td>
</tr>
<tr>
<td>ci_alpha</td>
<td>alpha transparency for the reference interval when <code>method = &quot;simulate&quot;</code>.</td>
</tr>
<tr>
<td>point_col</td>
<td>colour of points on the QQ plot.</td>
</tr>
<tr>
<td>point_alpha</td>
<td>alpha transparency of points on the QQ plot.</td>
</tr>
<tr>
<td>line_col</td>
<td>colour used to draw the reference line.</td>
</tr>
</tbody>
</table>

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

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

**Usage**
```
ref_level(fct)
level(fct, level)
```

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

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

**Examples**
```
f <- factor(sample(letters[1:5], 100, replace = TRUE))

# the reference level
ref_level(f)

# a specific level
level(f, level = "b")

# note that the levels will always match the input factor
identical(levels(f), levels(ref_level(f)))
identical(levels(f), levels(level(f, "c")))
```

**ref_sims**  
*Reference simulation data*

**Description**
A set of reference objects for testing `data_sim()`.

**Format**
A named list of simulated data sets created by `data_sim()`.
rep_first_factor_value

Repeat the first level of a factor n times

Description

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

Usage

rep_first_factor_value(f, n)

Arguments

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

Value

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

residuals_hist_plot

Histogram of model residuals

Description

Histogram of model residuals

Usage

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

Plot of residuals versus linear predictor values

Description
Plot of residuals versus linear predictor values

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

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

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 \texttt{ggplot2::labs()}.
subtitle character or expression; the subtitle for the plot. See \texttt{ggplot2::labs()}.
caption character or expression; the plot caption. See \texttt{ggplot2::labs()}.
point_col colour 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.
point_alpha numeric; alpha transparency for points in plots.
line_col colour specification for 1:1 line.

---

response_derivatives \textit{Derivatives on the response scale from an estimated GAM}

\textbf{Description}

Derivatives on the response scale from an estimated GAM

\textbf{Usage}

\begin{verbatim}
response_derivatives(object, ...)  
## Default S3 method:  
response_derivatives(object, ...)  
## S3 method for class \textquote{gamm}  
response_derivatives(object, ...)  
## S3 method for class \textquote{gam}  
response_derivatives(  
    object,  
    focal = NULL,  
    data = NULL,  
    order = 1L,  
    type = c("forward", "backward", "central"),  
    scale = c("response", "linear_predictor"),  
    method = c("gaussian", "mh", "inla", "user"),  
    n = 100,  
    eps = 1e-07,  
    n_sim = 10000,  
    level = 0.95,  
    seed = NULL,  
    mvn_method = c("mvnfast", "mgcv"),  
    ...  
)
\end{verbatim}
Arguments

object an R object to compute derivatives for.
... arguments passed to other methods and on to fitted_samples()
focal character; name of the focal variable. The response derivative of the response with respect to this variable will be returned. All other variables involved in the model will be held at constant values. This can be missing if supplying data, in which case, the focal variable will be identified as the one variable that is not constant.
data a data frame containing the values of the model covariates at which to evaluate the first derivatives of the smooths. If supplied, all but one variable must be held at a constant value.
order numeric; the order of derivative.
type character; the type of finite difference used. One of "forward", "backward", or "central".
scale character; should the derivative be estimated on the response or the linear predictor (link) scale? One of "response" (the default), or "linear predictor".
method character; which method should be used to draw samples from the posterior distribution. "gaussian" uses a Gaussian (Laplace) approximation to the posterior. "mh" uses a Metropolis Hastings sample that alternates t proposals with proposals based on a shrunken version of the posterior covariance matrix. "inla" uses a variant of Integrated Nested Laplace Approximation due to Wood (2019), (currently not implemented). "user" allows for user-supplied posterior draws (currently not implemented).
n numeric; the number of points to evaluate the derivative at (if data is not supplied).
eps numeric; the finite difference.
n_sim integer; the number of simulations used in computing the simultaneous intervals.
level numeric; \(0 < \text{level} < 1\); the coverage level of the credible interval. The default is 0.95 for a 95% interval.
seed numeric; a random seed for the simulations.
mvn_method character; one of "mvnfast" or "mgcv". The default is uses mvnfast::rmvn(), which can be considerably faster at generate large numbers of MVN random values than mgcv::rmvN(), but which might not work for some marginal fits, such as those where the covariance matrix is close to singular.

Value

A tibble, currently with the following variables:

- .row: integer, indexing the row of data each row in the output represents
- .focal: the name of the variable for which the partial derivative was evaluated,
- .derivative: the estimated partial derivative,
- .lower_ci: the lower bound of the confidence or simultaneous interval,
- .upper_ci: the upper bound of the confidence or simultaneous interval,
- additional columns containing the covariate values at which the derivative was evaluated.
response_derivatives

Author(s)

Gavin L. Simpson

Examples

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

df <- data_sim("eg1", dist = "negbin", scale = 0.25, seed = 42)

# fit the GAM (note: for execution time reasons using bam())
m <- bam(y ~ s(x0) + s(x1) + s(x2) + s(x3),
  data = df, family = nb(), method = "fREML"
)

# data slice through data along x2 - all other covariates will be set to
# typical values (value closest to median)
ds <- data_slice(m, x2 = evenly(x2, n = 100))

# fitted values along x2
fv <- fitted_values(m, data = ds)

# response derivatives - ideally n_sim = >10000
y_d <- response_derivatives(m,
  data = ds, type = "central", focal = "x2",
  eps = 0.01, seed = 21, n_sim = 1000
)

# draw fitted values along x2
p1 <- fv |>
  ggplot(aes(x = x2, y = .fitted)) +
  geom_ribbon(aes(ymin = .lower_ci, ymax = .upper_ci, y = NULL),
              alpha = 0.2
             ) +
  geom_line() +
  labs(
    title = "Estimated count as a function of x2",
    y = "Estimated count"
  )

# draw response derivatives
p2 <- y_d |>
  ggplot(aes(x = x2, y = .derivative)) +
  geom_ribbon(aes(ymin = .lower_ci, ymax = .upper_ci), alpha = 0.2) +
  geom_line() +
  labs(
    title = "Estimated 1st derivative of estimated count",
    y = "First derivative"
  )

# draw both panels
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

\texttt{rootogram(object, \ldots)}

\texttt{## S3 method for class 'gam'}
\texttt{rootogram(object, max_count = NULL, breaks = "Sturges", \ldots)}

Arguments

\begin{itemize}
  \item \texttt{object} \hspace{1cm} an R object
  \item \texttt{\ldots} \hspace{1cm} arguments passed to other methods
  \item \texttt{max_count} \hspace{1cm} integer; the largest count to consider
  \item \texttt{breaks} \hspace{1cm} for continuous responses, how to group the response. Can be anything that is acceptable as the breaks argument of \texttt{graphics::hist.default()}
\end{itemize}

References


Examples

\texttt{load_mgcv()}  
\texttt{df <- data_sim("eg1", n = 1000, dist = "poisson", scale = 0.1, seed = 6)}  
\texttt{# A poisson example}  
\texttt{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")}  
\texttt{rg <- rootogram(m)}  
\texttt{rg}  
\texttt{draw(rg) \# plot the rootogram}
# A Gaussian example

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

## seq_min_max_eps

Create a sequence of evenly-spaced values adjusted to accommodate a small adjustment

### Description

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

### Usage

\[
\text{seq_min_max_eps}(x, n, \text{order}, \text{type} = \text{c("forward", "backward", "central")}, \text{eps})
\]

### 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**

Shift numeric values in a data frame by an amount \( \varepsilon \).

**Usage**

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

**Arguments**

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

**Description**

Simulate from the posterior distribution of a GAM

**Usage**

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

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

---

**shift_values**  
*Shift numeric values in a data frame by an amount \( \varepsilon \)*

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

object a fitted GAM, typically the result of a call to \texttt{mgcv::gam()} or \texttt{mgcv::gamm()}.

nsim numeric; the number of posterior simulations to return.

seed numeric; a random seed for the simulations.

data data frame; new observations at which the posterior draws from the model should be evaluated. If not supplied, the data used to fit the model will be used for \texttt{newdata}, if available in \texttt{object}.

weights numeric; a vector of prior weights. If \texttt{newdata} is null then defaults to \texttt{object["prior.weights"]}, otherwise a vector of ones.

... arguments passed to methods. \texttt{simulate.gam()} and \texttt{simulate.scam()} pass ... on to \texttt{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.

newdata Deprecated. Use \texttt{data} instead.

Details

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

Value

(Currently) A matrix with \texttt{nsim} columns.
Author(s)

Gavin L. Simpson

Examples

```r
load_mgcv()

dat <- data_sim("eg1", n = 400, dist = "normal", scale = 2, seed = 2)

m1 <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = dat, method = "REML")

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

head(sims)
```

---

**smallAges**  

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

Description

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

Format

A data frame with 12 rows and 7 variables.

Details

The variables are as follows:

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

Source

smooths

_Names of smooths in a GAM_

**Description**

Names of smooths in a GAM

**Usage**

```r
smooths(object)
```

S3 method for class 'gamm'

```r
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

_Coefficients for a particular smooth_

**Description**

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

**Usage**

```r
smooth_coefs(object, ...)
```

S3 method for class 'gam'

```r
smooth_coefs(object, select, term = deprecated(), ...)
```

S3 method for class 'bam'

```r
smooth_coefs(object, select, term = deprecated(), ...)
```

S3 method for class 'gamm'

```r
smooth_coefs(object, select, term = deprecated(), ...)
```

S3 method for class 'gamm4'

```r
smooth_coefs(object, select, term = deprecated(), ...)
```
smooth_coefs

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

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

## S3 method for class 'scam'
smooth_coefs(object, select, term = deprecated(), ...)

### Arguments

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

### Value

A numeric vector of model coefficients.

### Author(s)

Gavin L. Simpson

### See Also

smooth_coef_indices() for extracting the indices of the coefficients for a particular smooth.

### Examples

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

## IGNORE_RDIFF_BEGIN
smooth_coefs(m, select = "s(x2)")
## IGNORE_RDIFF_END
```
**smooth_coef_indices**

Indices of the parametric terms for a particular smooth

**Description**

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

**Usage**

```r
smooth_coef_indices(smooth)
```

**Arguments**

- `smooth`  
an object that inherits from class `mgcv.smooth`

**Value**

A numeric vector of indices.

**Author(s)**

Gavin L. Simpson

**See Also**

- `smooth_coefs()` for extracting the coefficients for a particular smooth.

---

**smooth_data**

Generate regular data over the covariates of a smooth

**Description**

Generate regular data over the covariates of a smooth

**Usage**

```r
smooth_data(
  model,
  id,
  n = 100,
  n_2d = NULL,
  n_3d = NULL,
  n_4d = NULL,
  offset = NULL,
  include_all = FALSE,
  var_order = NULL
)
```
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_2d**: numeric; the number of new observations to generate for the second dimension of a 2D smooth. *Currently ignored.*
- **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 \((k \geq 4)\). 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 smooth. This can be useful if you want to pass the returned data frame on to `mgcv::PredictMat()`.
- **var_order**: character; the order in which the terms in the smooth should be processed. Only useful for tensor products with at least one 2d marginal smooth.

Examples

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

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

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

Description

Extracts the dimension of an estimated smooth.
smooth_estimates

Usage

smooth_dim(object)

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

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

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

Arguments

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

Details

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

Value

A numeric vector of dimensions for each smooth.

Author(s)

Gavin L. Simpson

smooth_estimates Evaluate smooths at covariate values

Description

Evaluate a smooth at a grid of evenly spaced value over the range of the covariate associated with the smooth. Alternatively, a set of points at which the smooth should be evaluated can be supplied. smooth_estimates() is a new implementation of evaluate_smooth(), and replaces that function, which has been removed from the package.

Usage

smooth_estimates(object, ...)

## S3 method for class 'gam'
smooth_estimates(
  object,
  select = NULL,
  smooth = deprecated(),
)
n = 100,
n_3d = 16,
n_4d = 4,
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.
select 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.
smooth [Deprecated] Use select instead.
n numeric; the number of points over the range of the covariate at which to evaluate the smooth.
n_3d, n_4d numeric; the number of points over the range of last covariate in a 3D or 4D smooth. The default is NULL which achieves the standard behaviour of using n points over the range of all covariates, 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 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

```r
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, select = c("s(x0)", "s(x1)"))
```

smooth_label

Extract the label for a smooth used by 'mgcv'

Description

The label 'mgcv' uses for smooths is useful in many contexts, including selecting smooths or labelling plots. `smooth_label()` extracts this label from an 'mgcv' smooth object, i.e. an object that inherits from class "mgcv.smooth". These would typically be found in the $smooth component of a GAM fitted by `mgcv::gam()` or `mgcv::bam()`, or related functions.

Usage

```r
smooth_label(object, ...)

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

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

Arguments

- `object` an R object. Currently, methods for class "gam" and for mgcv smooth objects inheriting from class "mgcv.smooth" are supported.
- `id` numeric; the indices of the smooths whose labels are to be extracted. If missing, labels for all smooths in the model are returned.
- `...` arguments passed to other methods.

Value

A character vector.
Examples

```r
load_mgcv()
df <- data_sim("gwf2", n = 100)
m <- gam(y ~ s(x), data = df, method = "REML")

# extract the smooth
sm <- get_smooths_by_id(m, id = 1)[[1]]

# extract the label
smooth_label(sm)

# or directly on the fitted GAM
smooth_label(m$smooth[[1]])

# or extract labels by idex/position
smooth_label(m, id = 1)
```

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.

Usage

```r
smooth_samples(model, ...)
```

## S3 method for class 'gam'
smooth_samples(
  model,
  select = NULL,
  term = deprecated(),
  n = 1,
  data = newdata,
  method = c("gaussian", "mh", "inla", "user"),
  seed = NULL,
  freq = FALSE,
  unconditional = FALSE,
  n_cores = 1L,
  n_vals = 200,
  burnin = 1000,
  thin = 1,
  t_df = 40,
  rw_scale = 0.25,
  rng_per_smooth = FALSE,
  draws = NULL,
)
Arguments

- **model**: a fitted model of the supported types
- **...**: arguments passed to other methods. For fitted_samples(), these are passed on to mgcv::predict.gam(). For posterior_samples() these are passed on to fitted_samples(). For predicted_samples() these are passed on to the relevant simulate() method.
- **select**: 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.
- **data**: data frame; new observations at which the posterior draws from the model should be evaluated. If not supplied, the data used to fit the model will be used for data, if available in model.
- **method**: character; which method should be used to draw samples from the posterior distribution. "gaussian" uses a Gaussian (Laplace) approximation to the posterior. "mh" uses a Metropolis Hastings sampler that alternates t proposals with proposals based on a shrunken version of the posterior covariance matrix. "inla" uses a variant of Integrated Nested Laplace Approximation due to Wood (2019), (currently not implemented). "user" allows for user-supplied posterior draws (currently not implemented).
- **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.
- **n_cores**: 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 data not supplied
- **burnin**: numeric; number of samples to discard as the burnin draws. Only used with method = "mh".
- **thin**: numeric; the number of samples to skip when taking n draws. Results in thin * n draws from the posterior being taken. Only used with method = "mh".
- **t_df**: numeric; degrees of freedom for t distribution proposals. Only used with method = "mh".
smooth_samples

rw_scale numeric; Factor by which to scale posterior covariance matrix when generating random walk proposals. Negative or non finite to skip the random walk step. Only used with method = "mh".

rng_per_smooth logical; if TRUE, the behaviour of gratia version 0.8.1 or earlier is used, whereby a separate call the the random number generator (RNG) is performed for each smooth. If FALSE, a single call to the RNG is performed for all model parameters.

draws matrix; user supplied posterior draws to be used when method = "user".

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

mvn_method character; one of "mvnfast" or "mgcv". The default is uses mvnfast::rmvn(), which can be considerably faster at generate large numbers of MVN random values than mgcv::rmvn(), but which might not work for some marginal fits, such as those where the covariance matrix is close to singular.

newdata Deprecated: use data instead.

ncores Deprecated; use n_cores instead. The 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 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; character vector. If the smooth involves a by term, the by variable will be named here, NA_character_ otherwise.
- .row; integer. A vector of values seq_len(n_vals), repeated if n > 1L. Indexes the row in data for that particular draw.
- .draw; integer. A vector of integer values indexing the particular posterior draw that each row belongs to.
- .value; numeric. The value of smooth function for this posterior draw and covariate combination.
- xxx; numeric. A series of one or more columns containing data required for the smooth, named as per the variables involved in the respective smooth.
- Additional columns will be present in the case of factor by smooths, which will contain the level for the factor named in by_variable for that particular posterior draw.

Warning

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

Author(s)
Gavin L. Simpson

Examples

load_mgcv()

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

sms <- smooth_samples(m1, select = "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)

smooth_terms

List the variables involved in smooths

Description

[Experimental]

Usage

smooth_terms(object, ...)

Arguments

object an R object the result of a call to mgcv::gam(), mgcv::bam(), or mgcv::gamm(), or that inherits from classes "gam" or "mgcv.smooth", or "fs.interaction".
... arguments passed to other methods. Currently unused.
smooth_type

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

Description

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

Usage

smooth_type(smooth)

## Default S3 method:
smooth_type(smooth)

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

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

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

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

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

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

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

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

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

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

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

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

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

## S3 method for class 'sw'
smooth_type(smooth)

## S3 method for class 'sf'
smooth_type(smooth)

## S3 method for class 'soap.film'
smooth_type(smooth)

## S3 method for class 't2.smooth'
smooth_type(smooth)

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

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

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

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

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

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

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

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

## S3 method for class 'mdcx.smooth'
smooth_type(smooth)
spline_values

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

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

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

Arguments

- **smooth**: an object inheriting from class `mgcv.smooth`.

spline_values

Evaluate a spline at provided covariate values

Description

Evaluate a spline at provided covariate values

Usage

spline_values(
  smooth,
  data,
  model,
  unconditional,
  overall_uncertainty = TRUE,
  frequentist = FALSE
)

Arguments

- **smooth**: currently an object that inherits from class `mgcv.smooth`.
- **data**: a data frame of values to evaluate smooth at.
- **model**: a fitted model; currently only `mgcv::gam()` and `mgcv::bam()` models are supported.
- **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?
- **frequentist**: logical; use the frequentist covariance matrix?
**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**

```r
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 (inheriting from class "gam" or an *mgcv::smooth.construct* smooth object, inheriting from class "mgcv.smooth").
- **...** 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**

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

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

**Arguments**

- **object** a fitted GAM object (inheriting from class "gam" or an *mgcv::smooth.construct* smooth object, inheriting from class "mgcv.smooth").
- **term** a term label.
- **...** arguments passed to other methods. Not currently used.

**Value**

A vector of variable names involved in a specified model 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.

---

### Description

General extractor for additional parameters in mgcv models

### Usage

```r
theta(object, ...)  
```

## 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

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

Arguments

- **smooth**: a smooth object of or inheriting from class "mgcv.smooth". Typically, such objects are returned as part of a fitted GAM or GAMM in the $smooth component of the model object or the $gam$smooth component if the model was fitted by mgcv::gamm() or gamm4::gamm4().
- **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.
- **coefs**: numeric; an optional vector of coefficients for the smooth
- **p_ident**: logical vector; only used for handling scam::scam() smooths.

Value

A tibble.

Author(s)

Gavin L. Simpson
Examples

```r
load_mgcv()

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

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

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

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

### too_far

<table>
<thead>
<tr>
<th>too_far</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exclude values that lie too far from the support of data</td>
</tr>
</tbody>
</table>

#### 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

```r
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`.

#### Value

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

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

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

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

## S3 method for class 'smooth_samples'
transform_fun(object, fun = NULL, constant = NULL, ...)

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

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

## S3 method for class 'parametric_effects'
transform_fun(object, fun = NULL, constant = NULL, ...)

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

Arguments
- **object**: an object to apply the transform function to.
- **fun**: the function to apply.
- **...**: additional arguments passed to methods.
- **constant**: numeric; a constant to apply before transformation.
- **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(),
    envir = environment(formula(object)),
    data = NULL,
    ...
)
```

```r
# S3 method for class 'data.frame'
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.
- `envir` the environment within which to recreate the data used to fit `object`.
- `data` an optional data frame of data used to fit the model if reconstruction of the data from the model doesn’t work.

### user_draws

**Handle user-supplied posterior draws**

#### Description

Handle user-supplied posterior draws

#### Usage

```r
user_draws(model, draws, ...)
```
Arguments

- **model**: a fitted R model. Currently only models fitted by `mgcv::gam()` or `mgcv::bam()`, or return an object that inherits from such objects are supported. Here, “inherits” is used in a loose fashion; models fitted by `scam::scam()` are supported even though those models don’t strictly inherit from class “gam” as far as inherits() is concerned.

- **draws**: matrix; user supplied posterior draws to be used when method = "user".

- **...**: arguments passed to methods.

---

**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
variance_comp(object, ...) 
```

```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.

Details

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

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

Returns names of variables from a smooth label

Description

Returns names of variables from a smooth label

Usage

vars_from_label(label)

Arguments

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

Examples

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

which_smooths

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, ...)

## 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

worm_plot(model, ...)

## S3 method for class 'gam'

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

## S3 method for class 'glm'

worm_plot(model, ...)

## S3 method for class 'lm'

worm_plot(model, ...)
Arguments

- model: a fitted model. Currently only class "gam".
- 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 x axis. If not supplied, a suitable label will be generated.
- title: character or expression; the title for the plot. See ggplot2::labs(). May be a vector, one per penalty.
- subtitle: character or expression; the subtitle for the plot. See ggplot2::labs(). May be a vector, one per penalty.
- caption: character or expression; the plot caption. See ggplot2::labs(). May be a vector, one per penalty.
- ci_col: character or expression; the fill colour for the reference interval when method = "simulate".
- ci_alpha: character or expression; the alpha transparency for the reference interval when method = "simulate".
- point_col: character or expression; the colour of points on the QQ plot.
- point_alpha: character or expression; the alpha transparency of points on the QQ plot.
- line_col: character or expression; the colour used to draw the reference line.

Note

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

Examples

load_mgcv()
## simulate binomial data...
dat <- data_sim("eg1", n = 200, dist = "binary", scale = .33, seed = 0)
p <- binomial()$linkinv(dat$x) # 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),

```
zooplankton

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

```r
worm_plot(m)
```

## Alternatively use simulate new data from the model, which
## allows construction of reference intervals for the Q-Q plot

```r
worm_plot(m,
  method = "simulate", point_col = "steelblue",
  point_alpha = 0.4
)
```

## ... or use the usual normality assumption

```r
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, 17  
gss_vocab, 79  
ref_sims, 112  
smallAges, 122  
zooplankton, 148
* draw methods
  draw.rootogram, 54
* utility
  boundary, 18  
  check_is_mgcv_smooth (is_mgcv_smooth), 82  
  coef.scam, 19  
  compare_smooths, 20  
  compare_smooths(), 38  
  concrvity (model_concurvity), 88  
  concrvity(), 51  
  confint.fderiv, 21  
  confint.gam, 23  
  confint.gamm (confint.gam), 23  
  confint.list (confint.gam), 23  
  data_combos, 25  
  data_sim, 26  
  data_sim(), 112  
  data_slice, 28  
  derivative_samples, 32  
  derivatives, 29  
  derivatives(), 11  
  difference_smooths, 34  
  draw, 35  
  draw(), 47  
  draw.basis, 36  
  draw.compare_smooths, 38  
  draw.derivatives, 38  
  draw.difference_smooth, 40  
  draw.evaluated_parametric_term, 42  
  draw.gam, 43  
  draw.gam(), 48  
  draw.gamlss, 47  
  draw.mgcv_smooth, 48  
  draw.overall_concurvity (draw.pairwise_concurvity), 50  
  draw.pairwise_concurvity, 50  
  draw.parametric_effects, 51  
  draw.partial_derivatives (draw.derivatives), 38  
  draw.penalty_df, 52  
  draw.rootogram, 54  
  draw.smooth_estimates, 55  
  draw.smooth_samples, 58
edf, 60
eval_smooth, 63
evaluate_parametric_term, 62
evaluate_parametric_term(), 42
evaluate_smooth, 63
evenly, 67
extract_link(link), 84
factor_combos, 68
family(), 69, 80, 85
family bam (family.gam), 68
family.gam, 68
family.gam (family.gam), 68
family.list (family.gam), 68
family_name, 69
family_type, 69
fitted_samples, 70
fitted_samples(), 8
fitted_values, 72
fix_offset, 75
fixed_effects (fixef.gam), 74
fixef, 74
fixef.gam, 74
fixef.gamm (fixef.gam), 74
fixef.glm (fixef.gam), 74
fixef.lm (fixef.gam), 74

gam::gam(), 69, 85, 139
gaussian_draws, 76
generate_draws (post_draws), 105
get_by_smooth, 77
get_smooth, 78
get_smooths_by_id, 78
ggplot2::coord_sf(), 46, 57
ggplot2::geom_contour(), 37, 41, 45, 56, 59
ggplot2::geom_line(), 56, 59
ggplot2::ggplot(), 36, 43, 46, 49, 54, 118
ggplot2::guide_axis(), 37, 39, 41, 46, 49, 52, 57, 59
ggplot2::label_both(), 37, 49
ggplot2::labs(), 37, 42, 43, 49, 53, 59, 94, 111, 114, 115, 147
GJRM::gamlss(), 47, 48
graphics::hist.default(), 118
graphics::par(), 13, 94, 115
gss_vocab, 79
gw_f0(), 79
gw_f1(gw_f0), 79
gw_f2(gw_f0), 79
gw_f3(gw_f0), 79
gw_functions, 27
gw_functions (gw_f0), 79
has_theta, 80
inv_link (link), 84
is_by_smooth, 81
is_continuous_by_smooth (is_by_smooth), 81
is_factor_by_smooth (is_by_smooth), 81
is_factor_term, 81
is_mgcv_smooth, 82
is_mrf_smooth (is_mgcv_smooth), 82
is_offset, 83
level (ref_level), 112
link, 84
load_mgcv, 86
lp_matrix, 86

mgcv::bam(), 17, 66, 69, 73, 85, 93, 123, 129, 133, 136, 144
mgcv::exclude.too.far(), 45, 66, 128, 140
mgcv::fix.family.rd(), 121
mgcv::gam, 121
mgcv::gam(), 17, 39, 41, 44, 52, 53, 56, 58, 66, 69, 72, 73, 85, 93, 105, 123, 129, 133, 136, 144
mgcv::gam.vcomp(), 144
mgcv::gamm, 17, 69, 85, 93, 101, 121, 123, 133, 139
mgcv::gammals(), 85
mgcv::gamSim(), 26
mgcv::gaulss(), 85
mgcv::gevlss(), 85
mgcv::gfam(), 27
mgcv::multinom(), 85
mgcv::mvn(), 85
mgcv::plot.gam(), 44
mgcv::predict.bam(), 86
mgcv::predict.gam(), 7, 70, 72, 73, 86, 104, 105, 108, 131
mgcv::PredictMat(), 126
mgcv::qq.gam(), 13, 111, 147
mgcv::residuals.gam(), 10
mgcv::s(), 14, 15
mgcv::shash(), 85
INDEX

mgcv::smooth.construct, 137
mgcv::smoothCon(), 15, 16, 102
mgcv::soap, 18
mgcv::t2(), 14, 15
mgcv::te(), 14, 15
mgcv::ti(), 14, 15
mgcv::twlss(), 85
mgcv::zipss(), 85
mh_draws, 87
model_concurvity, 88
model_concurvity(), 51
model_constant, 90
model_edf (edf), 60
model_vars, 90
mvnfast::rmvn(), 22, 24, 31, 71, 98, 104, 105, 131, 132
n_smooths, 93
nb_theta, 91
null_deviance, 92
observed_fitted_plot, 93
observed_fitted_plot(), 13
overview, 94

parametric_effects, 95
parametric_effects(), 62
parametric_terms, 96
partial_derivatives, 97
partial_residuals, 100
patchwork::plot_layout(), 13, 37–39, 41, 46, 48, 52, 53, 59
patchwork::wrap_plots(), 13, 38, 39, 41, 46, 49, 52, 53, 57, 59
penalty, 101
post_draws, 105
posterior_samples, 103
posterior_samples(), 8
predicted_samples, 108
predicted_samples(), 8

qq_plot, 109
qq_plot(), 13

ref_level, 112
ref_sims, 112
rep_first_factor_value, 113
residuals_hist_plot, 113
residuals_hist_plot(), 13
residuals_linpred_plot, 114
residuals_linpred_plot(), 13
response_derivatives, 115
rootogram, 118
rootogram(), 55
scam::scam(), 139
seq_min_max (evenly), 67
seq_min_max_eps, 119
shift_values, 120
simulate.gam, 120
simulate.gam(), 108
simulate.gamm (simulate.gam), 120
simulate.scam (simulate.gam), 120
smallAges, 122
smooth_coef_indices, 125
smooth_coef_indices(), 124
smooth_coefs, 123
smooth_coefs(), 125
smooth_data, 125
smooth_dim, 126
smooth_estimates, 127
smooth_estimates(), 63
smooth_label, 129
smooth_samples, 130
smooth_terms, 133
smooth_type, 134
smooths, 123
spline_values, 136
stats::family(), 68
stats::glm(), 85
stats::predict(), 6, 7, 10
stats::residuals(), 9, 10
stop_if_not_mgcv_smooth
(is_mgcv_smooth), 82

term_names, 137
term_variables, 137
theta, 138
tidy_basis, 139
to_na, 141
too_far, 140
too_far_to_na, 141
transform_fun, 142
typical_values, 143
user_draws, 143

variance_comp, 144
vars_from_label, 145
which_smooths, 145
worm_plot, 146
zooplankton, 148