Package: gratia (via r-universe)

October 27, 2026

Version 0.11.2-73

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

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

Depends R (>= 4.4.0)

Imports Rcpp, parallel, KernSmooth, mgcv (>= 1.9-4), ggplot2 (>= 4.0.3), tibble (>= 3.0.0), dplyr (>= 1.2.0), tidyr, rlang, patchwork (>= 1.3.2), vctrs, grid, mvnfast, purrr (>= 1.1.0), stats, grDevices, stringr, tidyselect (>= 1.2.0), lifecycle, pillar, cli, withr, scales, utils

Suggests posterior, coda, bayesplot, loo, ggcube (>= 0.3.0), geometry, qgam, tweedie (>= 2.3.5), gamm4 (>= 0.2-7), lme4, nlme, testthat, R6, pkgload, vdiffr, MASS, scam (>= 1.2-21), datasets, knitr, rmarkdown, forcats, GJRM, readr, glmmTMB (>= 1.1.15.2), sdmTMB (>= 1.2.0), RTMB, TMB, Matrix, gamair, sf (>= 1.0-0), marginaleffects, mirai, carrier

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

LinkingTo Rcpp

Roxygen list(markdown = TRUE)

Encoding UTF-8

Language en-GB

VignetteBuilder knitr

Config/testthat/edition 3

Config/Needs/website rmarkdown, ggdist, MASS, nlme, lme4, gamm4, GJRM, glmmTMB, sdmTMB, sf, ggcube, distributional, hexbin

Config/roxygen2/version 8.1.1

Config/pak/sysreqs libicu-dev

Repository https://gavinsimpson.r-universe.dev

Date/Publication 2026-10-27 06:29:06 UTC

RemoteUrl https://github.com/gavinsimpson/gratia

RemoteRef HEAD

RemoteSha 0b3434e0de72e649ce51862f76766989f29e397d

Contents

add_confint . . . 6
add_constant . . . 6
add_fitted . . . 7
add_fitted.gam . . . 8
add_fitted_samples . . . 10
add_partial_residuals . . . 12
add_residuals . . . 13
add_residuals.gam . . . 14
add_sizer . . . 15
ale . . . 16
appraise . . . 19
as_mcmc_draws . . . 26
assemble . . . 27
assemble.gam . . . 28
basis . . . 37
basis_size . . . 40
bird_move . . . 41
boundary . . . 42
check_user_select_smooths . . . 42
coef.scam . . . 43
compare_smooths . . . 44
conditional_differences . . . 46
conditional_moments . . . 50
conditional_values . . . 52
confint.fderiv . . . 54
confint.gam . . . 56
data_combos . . . 59
data_sim . . . 60
data_sim_families . . . 63
data_sim_shapes . . . 66
data_slice . . . 67
derivative_samples . . . 70

derivatives . . . 75
dharma_residuals . . . 79
diagnostic_support . . . 80
difference_smooths . . . 87
dispersion . . . 92
draw . . . 93
draw.ale . . . 94
draw.basis . . . 95
draw.compare_smooths . . . 97
draw.conditional_differences . . . 99
draw.conditional_moments . . . 100
draw.conditional_values . . . 101
draw.derivatives . . . 103
draw.difference_smooth . . . 105
draw.evaluated_parametric_term . . . 107
draw.gam . . . 108
draw.gamlss . . . 118
draw.mgcv_smooth . . . 123
draw.pairwise_concurvity . . . 124
draw.parametric_effects . . . 125
draw.penalty_df . . . 129
draw.rootogram . . . 130
draw.sampling_diagnostics . . . 132
draw.smooth_estimates . . . 134
draw.smooth_samples . . . 139
draw.variance_comp . . . 141
draw_label_control . . . 143
edf . . . 145
eval_smooth . . . 146
evaluate_parametric_term . . . 151
evaluate_smooth . . . 152
evenly . . . 153
factor_combos . . . 154
family.gam . . . 154
family_diagnostics . . . 155
family_name . . . 157
family_parameters . . . 158
family_type . . . 159
fitted_moments . . . 159
fitted_samples . . . 162
fitted_values . . . 168
fix_offset . . . 173
fixed_effects . . . 174
gamm4_support . . . 175
gaussian_draws . . . 176
get_by_smooth . . . 179
get_smooth . . . 179
get_smooths_by_id . . . 180

get_vcov . . . 180
gratia_palette . . . 181
gratia_theme_get . . . 182
gss_vocab . . . 184
gw_f0 . . . 184
has_theta . . . 185
is_by_smooth . . . 186
is_factor_term . . . 187
is_mgcv_family . . . 188
is_mgcv_smooth . . . 188
is_multivariate_y . . . 189
is_offset . . . 189
link . . . 190
load_mgcv . . . 192
log_lik . . . 192
loo.gam . . . 195
lp_matrix . . . 196
mh_draws . . . 198
model_concurvity . . . 201
model_constant . . . 202
model_terms . . . 203
model_vars . . . 205
n_eta . . . 206
n_smooths . . . 207
nb_theta . . . 208
null_deviance . . . 209
observed_fitted_plot . . . 209
overview . . . 210
parametric_effects . . . 212
parametric_terms . . . 215
partial_derivatives . . . 216
partial_residuals . . . 220
penalty . . . 221
post_draws . . . 224
posterior_predictive_plot . . . 228
posterior_samples.lm . . . 231
pp_check . . . 236
predicted_samples . . . 239
qq_plot . . . 242
qq_plot.glmmTMB . . . 247
quantile_residuals . . . 250
random_effect_samples . . . 255
ref_level . . . 257
ref_sims . . . 257
rep_first_factor_value . . . 258
residual_correlation . . . 258
residuals_covariate_plot . . . 260
residuals_hist_plot . . . 264

residuals_linpred_plot 266
response_derivatives 267
rootogram 272
rtw 273
sampling_diagnostics 274
scalable_qq_plot 275
scale_fill_partial_effect 280
sdmTMB-methods 281
seq_min_max_eps 285
shape_confidence_set 285
shape_intervals 288
shift_values 289
simulate 290
simultaneous_bands 293
smallAges 295
smooth_coef_indices 296
smooth_coefs 297
smooth_data 298
smooth_dim 300
smooth_estimates 301
smooth_label 307
smooth_samples 308
smooth_terms 313
smooth_type 313
smooths 318
spline_values 319
surrogate_residuals 320
term_names 322
term_variables 323
theta 324
tidy_basis 325
to_na 326
too_far 327
too_far_to_na 327
transform_fun 328
typical_values 329
user_draws 329
variance_comp 331
vars_from_label 335
which_smooths 336
worm_plot 337
zooplankton 341

Index **343**

add_confint *Add a confidence interval to an existing object*

Description

Add a confidence interval to an existing object

Usage

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

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

## 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.

Details

For smooth estimates, this function computes Gaussian pointwise intervals from the saved estimates and standard errors. It replaces any existing limits and updates their interval metadata. To retain stored bootstrap, shape-enforced or simultaneous limits, draw the estimates directly with `draw()` without calling `add_confint()`.

add_constant *Add a constant to estimated values*

Description

Add a constant to estimated values

Usage

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

## S3 method for class '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 = ".fitted", ...)

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

## S3 method for class 'scam'
add_fitted(data, model, ...)
```

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, stored in ".fitted" by default.

See Also

gamm4_support

---

add_fitted.gam *Add fitted values from a GAM to a data frame*

Description

Add fitted values from a GAM to a data frame

Usage

```
## S3 method for class 'gam'
add_fitted(
  data,
  model,
  value = ".fitted",
  type = "response",
  ...,
  envir = NULL,
  marginalise = NULL,
  integration = c("quadrature", "monte_carlo"),
```

```
  n_quad = 40L,
  n_integrate = 5000L,
  seed = NULL
)
```

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().
envir	environment or NULL; functions and constants needed to evaluate model expressions in prediction data. With NULL, use the model's formula environment. Supply raw covariates in data.
marginalise	NULL retains the usual predictions. TRUE selects all supported re and fs terms; character vectors select exact smooth labels. Fitted predictions integrate response means over new zero-mean Gaussian effects. Response samplers draw new effects before sampling responses. Smoothing parameters and dispersion remain fixed. Only ordinary gam() and bam() fits with a scalar response mean are supported.
integration	Integration method when marginalise is supplied: "quadrature" (the default) or "monte_carlo". Both integrate over the Gaussian distribution of the selected effects at each row.
n_quad	Initial Gauss-Hermite point count, between 2 and 1000. Successive doubled orders, up to four times this count, and an interlaced rule with one fewer point must agree within 1e-8 + 1e-6 * abs(value) for means and derivatives. Where the random-effect variance exceeds one, an independent adaptive rule also checks agreement. Failure to agree reports an error. Identity, log and probit links use their closed-form Gaussian integrals; their metadata records zero quadrature points and zero integration error.
n_integrate	Number of Monte Carlo integration draws, at least two. These are shared across rows and posterior coefficient draws. This count is separate from n_draws. Ignored for quadrature.
seed	Optional integer seed. An explicit seed preserves the caller's random-number state; NULL advances the current state.

Value

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

Examples

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

# add fitted 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(
  data,
  model,
  ...,
  n_draws = NULL,
  seed = NULL,
  chains = NULL
)

add_predicted_samples(data, model, ..., n_draws = NULL, seed = NULL)

add_posterior_samples(
  data,
  model,
  ...,
  n_draws = NULL,
  seed = NULL,
  chains = NULL
)

add_smooth_samples(
  data,
  model,
  ...,
```

```
  n_draws = NULL,
  seed = NULL,
  select = NULL,
  chains = NULL
)
```

Arguments

data	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.
...	arguments are passed to the posterior draw function, currently one of fitted_samples(), predicted_samples(), or posterior_samples(). n_draws and seed are already specified here as arguments and are also passed on to the posterior sampling function.
n_draws	positive integer; total retained draws across all chains. Required when generating draws. With method = "user", the supplied matrix determines the count; an explicit count must agree with its rows.
seed	numeric; a value to seed the random number generator.
chains	positive integer or NULL; number of chains. NULL uses one chain, or the chain mapping in supplied draws. n_draws must be divisible by chains. Burn-in is applied separately to each MH chain.
select	character; select which smooth's posterior to draw from. The default, NULL, means the posteriors of all smooths in model will be returned using the same joint coefficient draws. If supplied, a vector of 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_draws = 2` samples.
df |>
  add_fitted_samples(m, n_draws = 2, seed = 84)

# add posterior draws from smooth s(x2)
df |>
  add_smooth_samples(m, n_draws = 2, seed = 2, select= "s(x2)")
```

`add_partial_residuals` *Add partial residuals*

Description

Add partial residuals

Usage

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

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

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

## S3 method for class 'scam'
add_partial_residuals(data, model, ...)
```

Arguments

`data`	a data frame containing values for the variables used to fit the model, in fitting order. Supply either the retained fitting rows or, for `na.exclude`, the original fitting rows after any `subset`. This does not compute residuals for new or re-ordered observations.
`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.

Details

Partial residuals require working residuals and working weights from the fitted model. Families that do not provide these, including the location-scale families in `mgcv`, are not supported, even for terms in the location predictor.

See Also

gamm4_support

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", ...)

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

## S3 method for class 'scam'
add_residuals(data, model, ...)
```

Arguments

`data`	a data frame containing values for the variables used to fit the model, in fitting order. Supply either the retained fitting rows or, for `na.exclude`, the original fitting rows after any `subset`. This does not compute residuals for new or re-ordered observations.
`model`	a fitted model for which a `stats::residuals()` method is available. S3 method dispatch is performed on the `model` argument.
`value`	character; the name of the variable in which model residuals will be stored.
`...`	additional arguments passed to methods.

Value

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

See Also

gamm4_support

`add_residuals.gam` *Add residuals from a GAM to a data frame*

Description

Add residuals from a GAM to a data frame

Usage

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

Arguments

`data`	a data frame containing values for the variables used to fit the model. Passed to `stats::predict()` as `newdata`.
`model`	a fitted model for which a `stats::predict()` method is available. S3 method dispatch is performed on the `model` argument.
`value`	character; the name of the variable in which model predictions will be stored.
`type`	character; the type of residuals to return. See `mgcv::residuals.gam()` for options, or use `"surrogate"` for models fitted with `mgcv::ocat()` or `mgcv::multinom()`. Nominal surrogate residuals form a matrix column retaining all category labels, including category 0.
`...`	additional arguments passed to `mgcv::residuals.gam()` or, for `type = "surrogate"`, to `surrogate_residuals()`, including `seed`.

Value

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

Examples

```
load_mgcv()

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

##
add_residuals(df, m)
```

add_sizer *Add indicators of significant change after SiZeR*

Description

Add indicators of significant change after SiZeR

Usage

```
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

```
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)
```

ale *Accumulated local effects of a model covariate*

Description

Calculate one-dimensional accumulated local effects (ALE) of a numeric covariate, combining all model terms involving that covariate.

Usage

```
ale(object, ...)

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

## S3 method for class 'gam'
ale(
  object,
  focal,
  data = NULL,
  scale = c("link", "response", "linear_predictor"),
  n_bins = 40L,
  breaks = NULL,
  parameter = NULL,
  which_eta = NULL,
  center = TRUE,
  interval = c("confidence", "simultaneous"),
  ci_level = 0.95,
  unconditional = FALSE,
  frequentist = FALSE,
  n_sim = 10000L,
  n_cores = 1L,
  seed = NULL,
  block_size = 10000L,
  exclude = NULL,
  envir = NULL,
  ...
)

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

## S3 method for class 'scam'
ale(object, ..., unconditional = FALSE)
```

Arguments

object	a model fitted by mgcv::gam() or mgcv::bam().

`...`	named prediction controls passed to `mgcv::predict.gam()`. Data, type, term selection, row handling and uncertainty arguments are reserved.
`focal`	character; the name of one raw numeric model covariate.
`data`	optional reference data. Each row receives equal weight. By default, use the observations retained during fitting. Responses are not required. Supply raw covariates, including inputs to model expressions.
`scale`	character; `"link"` (default), `"response"`, or `"linear_predictor"`, an alias for `"link"`.
`n_bins`	positive integer; requested number of intervals. Empirical quantiles (`type = 1`) define boundaries; ties can reduce the number used.
`breaks`	optional numeric interval boundaries, overriding `n_bins`. These must increase strictly, span exactly the reference range, and produce no empty intervals.
`parameter`	character; one distribution parameter, using the names and aliases accepted by `link()`. `NULL` selects the first parameter.
`which_eta`	optional integer selecting a linear predictor instead of `parameter`. If both are supplied they must select the same predictor.
`center`	logical; subtract the empirical mean of the interpolated ALE curve over the reference observations? If `FALSE`, anchor the curve at zero at the first boundary.
`interval`	character; `"confidence"` for pointwise intervals or `"simultaneous"` for bands covering all returned boundaries of this curve.
`ci_level`	numeric; interval coverage, strictly between zero and one.
`unconditional`	logical; include smoothing parameter uncertainty if available. Otherwise `get_vcov()` warns and uses the uncorrected covariance.
`frequentist`	logical; use the frequentist coefficient covariance?
`n_sim`, `n_cores`	positive integers; simulation count and number of cores for simultaneous intervals. Ignored for pointwise intervals.
`seed`	optional integer seed for simultaneous intervals. An explicit seed preserves the caller's random-number state around interval simulation.
`block_size`	positive integer; maximum reference rows in each prediction block. Blocks can run in parallel when workers are configured; see Details.
`exclude`	character; model terms to exclude, as in `mgcv::predict.gam()`.
`envir`	optional environment for functions and constants in model expressions; see `smooth_estimates()`.

Details

Within each interval, predict at its two boundaries for every reference observation in that interval, retaining its other covariates. Average the upper-minus-lower prediction differences, then accumulate these averages. Intervals are right closed, with the minimum included in the first interval. The returned curve is linearly interpolated between boundaries. Centring subtracts its mean evaluated at the reference observations, rather than an unweighted mean over boundaries. Fitting weights do not weight these averages.

On the response scale, transform endpoint predictions before differencing. Transforming a link-scale ALE curve does not produce response-scale ALE. Response-scale effects are differences in natural parameter units, not predicted parameter values, and can be negative.

Distributional families `gaulss`, `gammals`, `gumbls`, `gevlss`, `shash`, `twlss`, and `ziplss` are supported. Natural units mean standard deviation for Gaussian scale, positive scale or dispersion for logged scales, natural Tweedie power, positive sinh-arcsinh tail parameter, and Poisson rate or probability of presence for `ziplss`. Location parameters need not be response means. Overall means combining several parameters, multinomial probabilities, multivariate covariance parameters and grouped-family outputs are unsupported. For other families, link-scale ALE is available for a single linear predictor; response-scale ALE requires a supported scalar mean family.

Supported response-scale parameters are:

- `gaulss`: location (mean) and scale (standard deviation);
- `gammals`: location (mean) and scale (dispersion);
- `gumbls` and `gevlss`: location and positive scale, plus GEV shape;
- `shash`: location, positive scale, skewness and positive tail parameter (named `kurtosis` to match gratia's parameter names);
- `twlss`: location (mean), power and scale (dispersion);
- `ziplss`: location (Poisson rate) and `pi` (probability of presence).

Raw-data recovery can require `envir` when a model was fitted locally and its formula no longer retains the environment containing the fitting data. Alternatively, supply reference `data` explicitly.

Coefficient gradients are averaged, accumulated and centred in the same way as the effects, retaining covariance across intervals and the centring constant. Link-scale gradients are exact; response-scale intervals use the delta method. Both interval types condition on the reference observations and boundaries. Simultaneous coverage concerns the returned boundaries, not all points between them. No uncertainty for the reference distribution is added.

Formula offsets are included and reevaluated at interval boundaries. Offsets supplied only through the fitting function's `offset` argument are omitted, following explicit-newdata prediction semantics. BAM predictions use `discrete = FALSE`, including for models fitted with `discrete = TRUE`.

Prediction blocks use an existing pool of workers configured with `mirai::daemons()`, when mirai and carrier are installed. Set `block_size` small enough to give each worker a block; serial and parallel calculations retain the same block boundaries and accumulation order. Worker startup and model transfer can outweigh the benefit for small models or reference data. The caller owns the pool; `ale()` does not start or stop workers. `n_cores` controls coefficient simulation for simultaneous intervals, not the number of prediction workers. Workers use their configured BLAS thread limits.

Data recovery and validation, centring, covariance calculations and interval simulation remain in the caller. Simultaneous bands use one shared set of coefficient deviations for all boundaries, preserving results and the caller's random-number state across worker counts. Models requiring custom prediction methods, family callbacks, contrasts or formula functions that workers cannot safely evaluate retain serial prediction. Standard arithmetic and common base and stats transformations can run on workers; local helpers, stochastic expressions, custom S3 transformation methods and other package functions retain serial prediction. Method checks do not invoke active or lazy caller bindings.

This implementation is written independently from the published ALE definition, without using code from mgcViz or ALEPlot.

Value

A tibble of class "ale", with .focal and .parameter identifying the curve, .value giving boundary coordinates, .ale the effects, .se their standard errors, .crit the interval critical value, and .lower_ci and .upper_ci the interval limits. .bin identifies the interval ending at each boundary and .n counts its reference rows. At the initial boundary both are zero. The ale_info attribute records computation settings without retaining the model or reference data.

References

Apley, D. W. and Zhu, J. (2020). Visualizing the effects of predictor variables in black box supervised learning models. Journal of the Royal Statistical Society: Series B, 82, 1059–1086. doi:10.1111/rssb.12377.

See Also

draw.ale(), conditional_values(), response_derivatives()

Examples

```
load_mgcv()
df <- data_sim("eg1", n = 200, seed = 42)
m <- gam(y ~ s(x0) + te(x1, x2, k = c(5, 5)) + s(x3),
  data = df, method = "REML")
a <- ale(m, focal = "x1")
draw(a)

# Standard deviation in its natural units, rather than reciprocal SD
ml <- gam(list(y ~ s(x1), ~ x2), data = df, family = gaulss())
ale(ml, "x2", parameter = "scale", scale = "response") |> draw()
```

---

appraise *Model diagnostic plots*

Description

Model diagnostic plots

Usage

```
appraise(model, ...)

## S3 method for class 'gam'
appraise(
  model,
  method = NULL,
  use_worm = FALSE,
  n_uniform = 10,
  n_simulate = 50,
```

```
  seed = NULL,
  type = NULL,
  n_bins = c("sturges", "scott", "fd"),
  ncol = NULL,
  nrow = NULL,
  guides = "keep",
  level = 0.9,
  ci_col = "black",
  ci_alpha = 0.2,
  point_col = "grey20",
  point_alpha = 1,
  line_col = "red",
  by_family = NULL,
  rank = NULL,
  scalable = NULL,
  scalable_threshold = 10000L,
  qq_bins = 1000L,
  density_bins = c(50L, 50L),
  ppcheck = FALSE,
  n_ppc = 50,
  density_method = c("kde", "histogram"),
  bandwidth = NULL,
  conditioning = NULL,
  ppc_batch_size = 10L,
  ...,
  n_dharma = 250L
)

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

## S3 method for class 'gamlss'
appraise(model, ..., type = "quantile", seed = NULL, conditioning = NULL)

## Default S3 method:
appraise(
  model,
  ...,
  type = NULL,
  seed = NULL,
  conditioning = NULL,
  n_dharma = 250L
)

## S3 method for class 'gamm4'
appraise(model, conditioning = NULL, ...)

## S3 method for class 'gjrm'
```

```
appraise(
  model,
  ...,
  type = "quantile",
  seed = NULL,
  conditioning = NULL,
  response = NULL,
  ppcheck = FALSE,
  by_family = NULL,
  n_dharma = 250L
)

## S3 method for class 'gamm'
appraise(model, conditioning = NULL, ...)

## S3 method for class 'gratia_diagnostic_proxy'
appraise(model, type = "quantile", ...)

## S3 method for class 'lme'
appraise(model, conditioning = NULL, ...)

## S3 method for class 'merMod'
appraise(model, conditioning = NULL, ...)

## S3 method for class 'sdmTMB'
appraise(
  model,
  conditioning = NULL,
  component = "response",
  field_method = c("mle-mvn", "mle-eb", "mle-mvn-integrated"),
  ...
)
```

Arguments

`model`	a fitted model supported by diagnostic_support, including lm, glm, gam/bam, gamm/gamm4, glmmTMB, univariate GJRM and sdmTMB.
`...`	arguments passed to `patchwork::wrap_plots()`.
`method`	character or `NULL`; how to calculate reference quantiles. Probability integral transform (PIT) and quantile residuals use fixed uniform and standard-normal references, respectively. For quantile residuals, `"normal"` also selects that fixed reference; other named methods require traditional residuals. For deviance, response and Pearson residuals: • `"uniform"` applies the fitted inverse cumulative distribution function (CDF) to randomised uniform quantiles and averages the resulting ordered residuals over `n_uniform` draws. • `"simulate"` generates `n_simulate` response data sets at the fitting covariates. Responses are generated together when needed to preserve depen-

dence, and their ordered residuals supply reference values and bands.

- `"marginal"` approximates the pooled distribution of individual residuals, without bands for dependence. This is selected for BAM GEE and PQL correlations that do not define a joint response distribution. Explicit joint simulation is unavailable in those cases.
- `"normal"` uses a normal approximation fitted to the residual location and scale. For dependent responses it omits bands based on independence.
- `"direct"` is deprecated in favour of `"uniform"`.

For ordinary GAMs, `NULL` selects `"simulate"` for Tweedie and `"uniform"` otherwise. Missing inverse CDFs cause a switch to simulation; missing response generators can cause a switch to a normal approximation for independent responses. Dependent responses use joint simulation when available, or marginal checks otherwise. For grouped families, `NULL` selects simulation if every component has a generator, or inverse-CDF sampling with a message if every component has an inverse CDF. If neither is available, it gives an error; a normal approximation must be requested explicitly.

`use_worm` logical; should a worm plot be drawn in place of the QQ plot?

`n_uniform` numeric; number of inverse-CDF draws to average when `method = "uniform"`.

`n_simulate` numeric; number of data sets to simulate from the estimated model when using the simulation method (`method = "simulate"`) for QQ plots.

`seed` numeric; the random number seed to use for `method = "simulate"` and `method = "uniform"`, and for randomised quantile residuals and posterior predictive draws.

`type` character or `NULL`; residual type: `"deviance"`, `"response"`, `"pearson"`, `"pit"`, `"quantile"`, `"dharma"`, or `"surrogate"` (for `ocat()` or `multinom()` GAMs). `NULL` uses the model-specific default: deviance for ordinary lm/glm/GAM fits, and quantile for supported grouped families, GJRM, sdmTMB and full mixed-model checks. glmmTMB uses Pearson residuals conditionally and response residuals marginally. Extracted GAM components follow the ordinary GAM defaults. See diagnostic_support.

`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 < level < 1`. Used for quantile reference bands and traditional `method = "simulate"` or `"normal"` bands. Also controls posterior predictive intervals for discrete responses with `ppcheck = TRUE`.

`ci_alpha, ci_col`

Deprecated; use `gratia_theme_get()` theme settings instead. Colour and transparency of QQ reference intervals, including bands for quantile residuals.

`point_col, point_alpha`

Deprecated; use `gratia_theme_get()` theme settings instead. 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. Pooled grouped residual panels map component identity to colour instead of using the constant `point_col`.

`line_col` Deprecated; use `gratia_theme_get()` theme settings instead. Colour specification for the 1:1 line in the QQ plot and the reference line in the residuals vs linear predictor plot. Also used for the observed distribution when `ppcheck = TRUE`.

`by_family` logical or `NULL`; separate indexed grouped-family components into facets. The default pools quantile residuals with component colours and facets traditional residuals. `TRUE` and `FALSE` explicitly select the layout. In `appraise()`, observed-versus-fitted values always use component facets. Standalone `observed_fitted_plot()` facets by default and permits explicit pooling. Ignored for ordinary models.

`rank` logical or `NULL`; use within-component ranks on the horizontal axis of residual-versus-predictor plots. The default ranks pooled grouped plots and retains original predictors in facets. `TRUE` or `FALSE` overrides this independently of the residual type and layout. Only finite observations included in each component determine its ranks; average ties are used.

`scalable` Logical or `NULL`; `NULL` automatically uses scalable plots at or above `scalable_threshold` retained fitting observations. `TRUE` or `FALSE` forces scalable or original rendering.

`scalable_threshold`
Positive integer; automatic switching threshold.

`qq_bins` Maximum number of QQ/worm points per family component, including endpoints.

`density_bins` Positive integer or pair giving heatmap grid dimensions.

`ppcheck` logical; replace the residual histogram with a `posterior_predictive_plot()`? Defaults to `FALSE`. Requires a model and family supported by that function. Replicated distributions use `ci_col`. Discrete posterior points and interval bars default to opacity `0.8`, while continuous density curves use `0.2`. Explicitly supplying `ci_alpha` overrides these defaults as well as the QQ reference interval opacity.

`n_ppc` positive integer; number of posterior predictive data sets when `ppcheck = TRUE`. Independent of the QQ plot's `n_simulate`.

`density_method` Conditional-density estimator: `"kde"` (the default) uses `KernSmooth::bkde2D()` with Gaussian kernels; `"histogram"` uses unsmoothed rectangular bins. Only affects the residual heatmap.

`bandwidth` Optional pair of positive bandwidths for the predictor and residual axes, in their plotted units. `NULL` uses normal-reference bandwidths, adjusted for two dimensions, with a fallback for constant axes. Used only for `density_method = "kde"` and conditional density.

`conditioning` character or `NULL`; `"conditional"` holds fitted random effects fixed, `"marginal"` averages over new effects, and `"gam_component"` uses the extracted GAM. Availability and defaults depend on the model; see diagnostic_support.

`ppc_batch_size` positive integer; number of posterior predictive draws to process together for scalable plots. Changing the batch size does not change results for a fixed seed.

n_dharma	positive integer; number of response simulations for type = "dharma". This is separate from n_simulate, which controls traditional QQ reference calculations. Defaults to 250.
response	NULL uses all responses; a single integer selects one. For mgcv::gfam() models, the integer indexes the family list and selects observations assigned to that component, including when families repeat. For joint GJRM models, it selects a response in model order. Selection is independent of by_family.
component	character; for delta sdmTMB models, select the complete "response", occurrence ("binomial"), or "positive" component. Positive-component checks use only observations with positive responses.
field_method	how to treat fitted sdmTMB fields: hold one posterior draw fixed ("mle-mvn"), hold estimates fixed ("mle-eb"), or average over fitted-field uncertainty ("mle-mvn-integrated"). See quantile_residuals() and diagnostic_support.

Details

Use type = "dharma" for simulation-based PIT residuals calculated internally using gratia's response generators; no DHARMa or mgcViz dependency is needed. These residuals lie in [0, 1], including exact endpoints, and do not require a response CDF. n_dharma controls their simulation count. All residual panels share one calculated vector. QQ and worm plots require method = NULL and use uniform reference quantiles, with approximate pointwise bands for independent responses. Bands are omitted for dependent responses. Censored and multivariate outcomes are unsupported. See dharma_residuals() for the calculation.

Use type = "surrogate" for mgcv::ocat() and mgcv::multinom() GAMs. The former uses a scalar latent logistic residual; the latter retains all category utilities, including category 0, as separate panels with centred Gumbel references. QQ and worm plots require method = NULL. Each appraisal shares one residual draw across panels, with the same draw in ordinary and scalable displays. Multinomial residuals are plotted against their own category probabilities; observed-versus-fitted panels compare category indicators with those probabilities. Pointwise bands are calculated separately for each utility. See surrogate_residuals().

With type = NULL, ordinary lm/glm/GAM fits use deviance residuals. Models fitted with mgcv::gfam() use randomised quantile (Dunn–Smyth) residuals when every active component supplies a supported CDF and probability limits for discrete or censored observations. Otherwise, a message identifies unsupported components and all components use deviance residuals; requesting type = "quantile" explicitly is an error when any component is unsupported. Components never mix residual scales. Invalid encodings or numerical CDF failures are errors.

Quantile residuals use each component's fitted parameters, dispersion and weights, including binomial trial counts and Gaussian precision weights. Gamma (including precision weights), negative binomial, beta, scaled-t, Tweedie, inverse Gaussian, ordinal and zero-inflated Poisson components are supported. Unrecognised custom-family bounds remain unsupported. A custom CDF must also support the probabilities just below discrete values and at censoring limits. Replacements for recognised family CDFs must use the same arguments and probability conventions. Scalar uncensored cnorm(), clog() and cpois() components are supported; genuine censoring requires recoverable, aligned censoring bounds. Matrix censored responses rejected by gfam() and upstream fitting failures are not repaired.

Use response = NULL to check every grouped-family component or a single integer to select a component by its position in the mgcv::gfam() family list. Repeated families remain distinct. Se-

lection filters fitting observations without refitting the model and retains original component labels. `by_family` independently controls the layout of the selected data. With an explicit seed, selected residuals match the corresponding rows of an all-component calculation whenever all components support that calculation.

`by_family = NULL` pools quantile residuals with colours identifying indexed family components, including repeated uses of the same family. Traditional residuals use component facets with free axes by default. The pooled residual-versus-predictor plot uses within-component predictor ranks: `(rank(eta, ties.method = "average") - 0.5) / n`, where `n` counts the finite observations included for that component; constants map to `0.5`. Facets retain original linear predictors. Use `rank` to override this choice independently of residual type or layout. The links and original predictor values can have different meanings across components; ranks compare positions within components, not effects on a common response scale. Pooled histograms use common bin edges and component-specific densities.

Quantile QQ and worm plots compare to fixed standard-normal quantiles, with an identity line or zero line. Independent responses have exact pointwise reference bands for ordered residuals. For dependent responses, bands use simulations that preserve dependence when a joint generator is available; otherwise they are omitted with an explanation. Bands condition on fitted parameters, exclude estimation uncertainty and do not provide simultaneous coverage. References calculated from simulation ranks also use simulated bands; see diagnostic_support. Worm plots subtract the same theoretical quantiles from residuals and reference bands. `appraise()` computes one seeded residual vector shared by its residual panels and always separates observed-versus-fitted values by component, because response units differ. Standalone `observed_fitted_plot()` allows explicit pooling. Ordinal panels use expected category codes.

Scalable plots aggregate all observations before rendering. Residual heatmaps show conditional density; observed-versus-fitted heatmaps show counts. Point styling applies to QQ/worm points, and heatmap fill scales can be changed with ggplot scales. QQ references average draws as they are generated or compute exact pooled quantiles using temporary files above the configured memory threshold; see diagnostic_support. Supported univariate GJRM::gamlss models use quantile residuals by default.

If the response expectation does not exist, the observed-versus-fitted panel explains why it is unavailable. Other supported panels remain usable. Prepared diagnostic objects can be given a new conditioning or field choice; the model is prepared again so all panels use the requested choice.

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()`, `observed_fitted_plot()`, and optionally `posterior_predictive_plot()`.

gamm4_support

Examples

```
load_mgcv()
```

```
# Simulate an additive Gaussian response and fit a GAM
dat <- data_sim("eg1", n = 200, scale = 2, seed = 2)
mod <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = dat, method = "REML")

# Inspect residuals, fitted values and the residual distribution
appraise(mod, seed = 42)
```

`as_mcmc_draws` *Convert posterior samples for MCMC diagnostics*

Description

Coefficient matrices retain their chain mapping in an attribute. Sampled quantities use their `.chain`, `.iteration`, and `.draw` columns. A plain matrix without a mapping is treated as one chain; chain boundaries are never inferred from its number of rows. Filtering must leave the same consecutive iterations in every chain. Missing observations remain missing variables.

Usage

```
as_mcmc_draws(
  x,
  format = c("draws_array", "draws_df", "mcmc.list", "array"),
  value = NULL
)
```

Arguments

`x`	A coefficient matrix from `post_draws()` or a table from `fitted_samples()`, `posterior_samples()`, `smooth_samples()`, or `derivative_samples()`.
`format`	One of `"draws_array"`, `"draws_df"`, `"mcmc.list"`, or `"array"`. The first two require posterior; `"mcmc.list"` requires coda.
`value`	Column containing sampled values. By default the appropriate column is selected from `.fitted`, `.response`, `.value`, or `.derivative`.

Value

An iteration by chain by variable array, a posterior draws object, or a coda list of chains. Variable names identify sampled quantities and their observation rows, parameters, or smooths.

`assemble` *Prepare plots via* `ggplot2` *and assembles them as a list*

Description

Generic function for assembling plot objects created from R objects, using the `ggplot2` package.

Usage

```
assemble(object, ..., label_control = NULL)

## S3 method for class 'gamm4'
assemble(object, ...)
```

Arguments

`object`	and R object to plot.
`...`	arguments passed to other methods.
`label_control`	Optional layout settings created by `draw_label_control()`. These apply across drawing methods without changing their default appearance.

Value

A list of `ggplot2::ggplot()` objects.

Label layout

Pass `label_control = draw_label_control(...)` to `draw()` to wrap discrete axis labels and legend text, stagger discrete tick labels, or adjust legend and colour-bar layout. Settings apply to each panel, including unwrapped plot lists. See `draw_label_control()` for the available settings.

Author(s)

Gavin L. Simpson

See Also

gamm4_support

`assemble.gam` *Plot smooth and parametric terms from a fitted GAM*

Description

Plots smooth and parametric contributions from a fitted GAM model using `ggplot2::ggplot()`. For models fitted by `mgcv::gamm()`, `assemble()` and `draw()` use the fitted `gam` component.

Usage

```
## S3 method for class 'gam'
assemble(
  object,
  data = NULL,
  select = NULL,
  parametric = NULL,
  terms = NULL,
  residuals = FALSE,
  scales = c("free", "fixed"),
  ci_level = 0.95,
  n = 100,
  n_2d = 50,
  n_3d = 16,
  n_4d = 4,
  unconditional = FALSE,
  overall_uncertainty = TRUE,
  constant = NULL,
  fun = NULL,
  dist = 0.1,
  rug = TRUE,
  distinct_rug = TRUE,
  contour = TRUE,
  grouped_by = deprecated(),
  ci_alpha = 0.2,
  ci_col = "black",
  smooth_col = "black",
  resid_col = "steelblue3",
  contour_col = "black",
  n_contour = NULL,
  partial_match = FALSE,
  discrete_colour = NULL,
  discrete_fill = NULL,
  continuous_colour = NULL,
  continuous_fill = NULL,
  position = "identity",
  angle = NULL,
  ncol = NULL,
```

```
  nrow = NULL,
  guides = "keep",
  widths = NULL,
  heights = NULL,
  crs = NULL,
  default_crs = NULL,
  lims_method = "geometry_bbox",
  wrap = TRUE,
  caption = TRUE,
  envir = NULL,
  surface = c("raster", "tile", "3d"),
  surface_3d = list(),
  ci_3d = FALSE,
  ci_3d_alpha = 0.15,
  ci_3d_col = "grey60",
  back_transform = FALSE,
  re_levels = NULL,
  mrf_plot = c("effects", "qq"),
  interval = c("confidence", "simultaneous"),
  n_sim = 10000,
  n_cores = 1,
  seed = NULL,
  interval_method = c("auto", "gaussian", "bootstrap", "shape"),
  frequentist = FALSE,
  shape_source = c("gaussian", "bootstrap"),
  ...,
  variables = NULL,
  simultaneous_method = c("simulation", "tube"),
  simultaneous_domain = NULL,
  tube_control = list(),
  cyclic = c("cartesian", "polar"),
  polar_args = list(),
  grouped = c("sz", "fs"),
  show_ci = c("factor-by", "sz"),
  time_varying_style = c("ribbon", "pointrange")
)

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

## S3 method for class 'scam'
assemble(object, ..., overall_uncertainty = FALSE, centre = c("model", "plot"))
```

Arguments

object	a fitted GAM, the result of a call to mgcv::gam().
data	an optional data frame that is used to supply the data at which the smooths will be evaluated and plotted. This is usually not needed, but is an option if you need

fine control over exactly what data are used for plotting.

`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 or `NULL`; plot parametric terms also? The default, `NULL`, includes parametric terms when `select = NULL`, but excludes them when smooths are selected explicitly. Use `TRUE` to include parametric terms regardless of `select`, or `FALSE` to draw only smooths (the previous default). The `terms` argument selects which parametric effects are plotted; `select` selects only smooths.

`terms` character; which model parametric terms should be drawn? The Default of `NULL` will plot all parametric terms that can be drawn. Interactions show their individual contributions to the linear predictor; main effects and intercepts are not added. Multivariate parametric terms use the grid and surface controls below, as described in `parametric_effects()`.

`residuals` logical; should partial residuals for a smooth be drawn? Ignored for anything but a simple univariate smooth. If the model does not provide working residuals and working weights, warns once and draws the smooths without partial residuals.

`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.

`n` numeric; the number of points over the range of the covariate at which to evaluate a univariate smooth.

`n_2d` numeric; the number of points along each of the first two axes of a smooth surface, including surface panels of higher-dimensional smooths. The default is 50 in plotting and plot-preparation functions. If `NULL`, use `n` instead. Ignored when evaluation `data` are supplied. Factor levels are retained, and curves with only one continuous covariate use `n`.

`n_3d` numeric; the number of new observations to generate for the third dimension of a 3D smooth. Factor conditioning dimensions retain at most this many levels, taking the first levels in fitted order. Explicit `re_levels` selections take precedence; supplied `data` are unchanged.

`n_4d` numeric; the number of new observations to generate for the dimensions higher than 2 (!) of a kD smooth ($k >= 4$). For example, if the smooth is a 4D smooth, each of dimensions 3 and 4 will get `n_4d` new observations. For factor conditioning dimensions, this is the maximum number of levels retained per dimension, taking the first in fitted order. Explicit `re_levels` selections take precedence; supplied `data` are unchanged.

`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 errors of smooth and parametric effects? For parametric effects this includes the intercept variance and its covariance with the term, without changing the partial estimates. Each term uses the intercept of its own linear predictor. Native scasm models are routed to `draw.scasm()` with `overall_uncertainty = FALSE`. An explicit request for unavailable mean uncertainty warns once. The stored model class is unchanged.

`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. For simultaneous bands it must be monotone over the entire band; endpoint transformation alone does not preserve non-monotone band coverage. Function `fun` will be applied after adding any `constant`, if provided.

`dist`
: numeric; if greater than 0, this is used to determine when a location is too far from data to be plotted when plotting 2-D smooths. The data are scaled into the unit square before deciding what to exclude, and `dist` is a distance within the unit square. See `mgcv::exclude.too.far()` for further details.

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

`distinct_rug`
: logical; should the data for the rug plots be reduced to the distinct values?

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

`grouped_by`
: Deprecated; use `grouped`. Explicit `TRUE` adds `"factor-by"` to `grouped`; `FALSE` removes it, retaining the other types.

`ci_alpha`
: Deprecated; use `gratia_theme_get()` theme settings instead. Numeric; alpha transparency for confidence or simultaneous interval.

`ci_col`
: Deprecated; use `gratia_theme_get()` theme settings instead. Colour specification for the confidence/credible intervals band. Affects the fill of the interval.

`smooth_col`
: Deprecated; use `gratia_theme_get()` theme settings instead. Colour specification for the smooth line.

`resid_col`
: Deprecated; use `gratia_theme_get()` theme settings instead. Colour specification for the partial residuals.

`contour_col`
: Deprecated; use `gratia_theme_get()` theme settings instead. 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()`.

`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.

`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 non-sf layers added to spherical plots. The default, `NULL`, uses 4326 (WGS84). Internally generated surface cells, contours and observation points have their own geographic CRS, with longitude on the horizontal axis and latitude on the vertical axis. See `ggplot2::coord_sf()` for more details.

`lims_method`
: character; affects how spherical plot axis limits are determined. The default, `"geometry_bbox"`, uses the rendered geometry. Passed to `ggplot2::coord_sf()`. Spherical surfaces use explicit geographic cells with valid polar and date-line boundaries. Orthographic plots clip cells, contours and observation points to the visible hemisphere before projection. Explicit prediction data on a rectilinear grid retain their actual spacing; scattered or singleton coordinates are drawn as points, with a message, because they do not define surface cells. Incomplete observation rug coordinates are omitted. Grouped factor-by spherical surfaces use separate facets for each factor level.

`wrap`
: logical; wrap plots as a patchwork? If `FALSE`, return a list of ggplot objects, one for each displayed panel. Separate factor levels can produce several plots for one smooth term.

`caption`
: logical; show the smooth type in the caption of each plot?

`envir`
: an optional environment for looking up data, functions and constants. Defaults to the model's evaluation environment.

`surface`
: character; one of `"raster"` (the default), `"tile"`, or `"3d"`. Raster surfaces keep large PDF plots compact; tiles are borderless vector rectangles. Three-dimensional surfaces use the optional ggcube package, with the displayed partial effect as height and fill. Numeric smooth and parametric surfaces, including conditioning facets and soap films, are supported. Curves and factor estimates keep their usual display; spherical smooths retain their geographic rendering. Covariate scale changes that produce irregular spacing use rectangles bounded by adjacent coordinate midpoints for flat surfaces.

surface_3d — named list of view options passed to ggcube::coord_3d(), for example list(yaw = 45, persp = FALSE). The default list() uses the default view with ggcube::light(backface_scale = 1), preserving equal horizontal units for isotropic smooths. Supply light = ggcube::light(...) in this list to override lighting; ggcube::light(method = "none") disables it. An explicitly supplied light replaces the entire default light. Only the coordinate system is used; ggcube's theme settings are not applied. These options affect all 3D panels in a call. For different views per term, modify individual plots returned by assemble(). The dist entry here controls perspective; the top-level dist argument controls masking.

ci_3d — logical; add lower and upper interval surfaces to numeric smooth and parametric effect panels when surface = "3d"? Defaults to FALSE. Applies across supported model classes and conditioning facets. Other panel types, including standard errors, retain their usual display. Smooth bounds follow interval; parametric bounds remain pointwise.

ci_3d_alpha — Deprecated; use gratia_theme_get() theme settings instead. Numeric between 0 and 1; opacity of both interval surfaces, independent of ci_alpha. Defaults to 0.15.

ci_3d_col — Deprecated; use gratia_theme_get() theme settings instead. One R colour; fill of both interval surfaces, independent of ci_col and the estimate's continuous fill scale. Defaults to "grey60".

back_transform — logical; plot covariates on their original scale? Defaults to FALSE, using model coordinates such as log(x) for a term containing log(x). With TRUE, recover the raw covariates, generate a grid on their original scale and evaluate the model expressions forwards. Applies to axes, rugs, partial-residual locations and conditioning facets, including tensor products and parametric interactions. Each smooth axis must depend on one distinct original model covariate. Functions such as foo(x) are supported when their raw inputs and functions can be recovered; no inverse function is needed. Missing raw data or unavailable functions produce an error; supply data or envir as appropriate. With supplied prediction data, unavailable raw fitting observations omit rugs and partial residuals; prediction rows are never used as observations. For precomputed smooth estimates, prepare raw grids with smooth_estimates(model, back_transform = TRUE) before drawing with back_transform = TRUE. Raw inputs supplied through data also retain their coordinates. Transformed-only estimates cannot be relabelled on the raw scale, including grids carrying typical values from include_all = TRUE. Transformed soap-film boundaries lack retained raw coordinates and must be drawn with back_transform = FALSE. Parametric estimates retain both scales; multi-column basis expansions retain raw axes on either setting. Does not transform estimated effects or their intervals; use fun for that.

re_levels — named list of factor level labels to retain in tensor products with random-effect margins, for example list(species = c("a", "b")). Fitted factor order and ordered status are retained.

mrf_plot — character; display Markov random field smooths as area-labelled estimates and intervals ("effects", the default), or a descriptive Gaussian quantile plot ("qq"). MRF effects are correlated and need not have equal marginal variances; the QQ

display is not a test of independent normal effects. Specialist maps and graph plots are outside the scope of this display.

`interval` character; `"confidence"` (default) draws pointwise intervals; `"simultaneous"` uses a band across, by default, the finite evaluation grid of each smooth, including its slices and factor levels. Separate smooths are not a joint coverage set. `"credible"` aliases `"confidence"`, reflecting the approximate Bayesian credible interpretation of default-covariance bands. Parametric panels retain pointwise intervals and are labelled accordingly when simultaneous smooth bands are requested. Bands appear where the renderer displays interval bounds, including 3D effect surfaces when `ci_3d = TRUE`. Random-effect QQ plots retain their usual display. `simultaneous_method = "tube"` calibrates supported univariate bands over a continuous interval; see simultaneous_bands.

`n_sim, n_cores` positive integers; coefficient draws and simulation cores for simultaneous bands. Parallel simulation requires OpenMP support.

`seed` integer or `NULL`; an explicit simulation seed preserves the caller's RNG state.

`interval_method`
character; "auto" uses stored scasm bootstrap draws for pointwise intervals when available, and Gaussian intervals otherwise. Use "gaussian" to request covariance-based limits or "bootstrap" to require stored draws. Simultaneous bootstrap bands are not supported. `"shape"` applies `shape_intervals()` to supported scasm smooth components, using the initial limits chosen by `shape_source`.

`frequentist` logical; use frequentist covariance for smooth-component intervals? The default uses approximate Bayesian covariance. Parametric effect intervals retain their existing Bayesian covariance calculation.

`shape_source` character; initial uncertainty for `interval_method = "shape"`: `"gaussian"` (the default) or stored `"bootstrap"` pointwise limits. Bootstrap sources require `frequentist = FALSE` and `unconditional = FALSE` because the stored draws hold smoothing fixed.

`...` Additional smooth plotting arguments, including `title`, `subtitle`, `xlab` and `ylab`. `subtitle` also applies to parametric panels; use `NULL` for automatic predictor labels or `""` to suppress them.

`variables` Character vector giving every covariate of the selected smooths exactly once, in plotting order: x, y, then the conditioning covariate for 3D smooths, or facet rows and columns for 4D smooths. When supplied, selected smooths must use the same covariates. Use fitted coordinate names, such as `"log(x)"` for a transformed covariate. The default, `NULL`, favours a two-variable tensor margin for x and y, otherwise retaining fitted order. Grid resolution follows this order. Supplied `data` are evaluated unchanged. Drawing saved estimates can change the order but uses only points already evaluated; evaluate again to change grid resolution or conditioning values. Continuous covariates must precede factors. Spherical smooths retain their latitude/longitude convention and do not support reordering.

`simultaneous_method`
character; calibration for simultaneous bands. The default, `"simulation"`, uses coefficient draws on the evaluation grid. `"tube"` uses a conservative Gaussian threshold-crossing bound over a continuous interval for supported one-dimensional cubic splines. See simultaneous_bands.

simultaneous_domain
: numeric pair of increasing finite endpoints, or a named list of such pairs keyed by spline coordinate. For tube bands, NULL uses the range of returned evaluation coordinates, including gaps. All returned points must lie inside the interval and the interval must lie inside the fitted spline support. Requires tube calibration.

tube_control
: named list controlling numerical tube calibration: rel_tol (default 1e-6), abs_tol (1e-8), subdivisions (1000L), and root_tol (1e-8). Requires tube calibration. Simulation controls n_sim, n_cores (or ncores), and seed are ignored by tube calibration.

cyclic
: character; use Cartesian coordinates ("cartesian", the default) or polar coordinates ("polar") for one-dimensional cyclic smooths. Other smooths and parametric terms keep their usual displays.

polar_args
: named list of options passed to ggplot2::coord_radial(). Accepts start, reverse, inner.radius, r.axis.inside, rotate.angle and clip. The radial axis is inside the circle by default (r.axis.inside = TRUE); supplied values override this default. Other options use their ggplot2 defaults. Requires cyclic = "polar" when nonempty. The angular variable, fitted cycle boundaries, full-circle extent and angular expansion are set by gratia.

grouped
: character or FALSE; factor smooth types whose curves share a panel. Choose any of "factor-by", "sz", and "fs", or FALSE to separate all their curves. The default, c("sz", "fs"), overlays sz and fs curves and separates factor-by curves. "factor-by" includes ordered factors. In reference-plus-deviation models the reference smooth remains separate; only fitted nonreference deviations are grouped. For sz terms with several factors, separate plots show each evaluated combination of levels. Applies to curves with one continuous covariate; existing surface displays are unchanged. Separate curves use the usual univariate styling: a black line and a translucent black interval by default, with colours and opacity controlled by the smooth theme settings.

show_ci
: character or FALSE; factor smooth types whose curves show intervals. Choose any of "factor-by", "sz", and "fs", or FALSE to hide all their ribbons. The default, c("factor-by", "sz"), retains existing interval visibility; including "fs" enables its intervals. Controls display only, independently of interval and grouped. Ordinary smooths and parametric effects retain their intervals.

time_varying_style
: Display sdmTMB time-varying coefficients using a line, points and interval ribbon ("ribbon", the default), or points and interval bars with a connecting line ("pointrange"). Other model terms keep their usual displays.

centre
: character; for SCAM models, "model" retains each fitted smooth contribution. "plot" subtracts the evaluation-grid mean from each bivariate smooth to match scam::plot.scam(). The same linear transformation is applied to its covariance. Univariate curves and smooth_estimates() retain the fitted centring for either setting.

Details

Smooth and parametric plot subtitles identify the model part or distributional parameter as described in draw.gam(). Custom labels, including subtitle, can be passed through .... Use cyclic =

`"polar"` to draw one-dimensional cyclic smooths with polar coordinates; `polar_args` customizes the coordinate display as described in `draw.gam()`.

Value

A list of `ggplot2::ggplot()` objects.

Label layout

Pass `label_control = draw_label_control(...)` to `draw()` to wrap discrete axis labels and legend text, stagger discrete tick labels, or adjust legend and colour-bar layout. Settings apply to each panel, including unwrapped plot lists. See `draw_label_control()` for the available settings.

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")

# assemble plots of all smooths
assemble(m1)

# can be passed to patchwork::wrap_plots()
suppressPackageStartupMessages(library("patchwork"))
assemble(m1) |> wrap_plots(ncol = 2, nrow = 2)

# Prepare individual plots from a GAMM
m2 <- gamm(y ~ s(x0) + x1, data = df1, method = "REML")
gamm_plots <- assemble(m2)
gamm_plots[[1]]
```

`basis` *Basis expansions for smooths*

Description

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

Usage

```
basis(object, ...)

## 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,
  envir = NULL,
  ...,
  variables = NULL,
  unweighted = FALSE
)

## 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,
  ...,
  variables = NULL,
  unweighted = FALSE
)

## 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,
  ...,
  variables = NULL,
  unweighted = FALSE
)

## S3 method for class 'gamm4'
basis(
  object,
  select = NULL,
  term = deprecated(),
  data = NULL,
  n = 100,
  n_2d = 50,
  n_3d = 16,
  n_4d = 4,
  partial_match = FALSE,
  ...,
  variables = NULL,
  unweighted = FALSE
)

## Default S3 method:
basis(
  object,
  data,
  knots = NULL,
  constraints = FALSE,
  at = NULL,
  diagonalize = FALSE,
  coefficients = NULL,
  envir = NULL,
  ...,
  variables = 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 a univariate smooth.
`n_2d`	numeric; the number of points along each of the first two axes of a smooth surface, including surface panels of higher-dimensional smooths. The default is 50 in plotting and plot-preparation functions. If `NULL`, use `n` instead. Ignored when evaluation `data` are supplied. Factor levels are retained, and curves with only one continuous covariate use `n`.
`n_3d`	numeric; the number of new observations to generate for the third dimension of a 3D smooth.
`n_4d`	numeric; the number of new observations to generate for the dimensions higher than 2 (!) of a kD smooth (k >= 4). For example, if the smooth is a 4D smooth, each of dimensions 3 and 4 will get `n_4d` new observations.
`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.
`envir`	an optional environment supplying functions and constants used in model expressions. The available model formula environment is used when `NULL`. Covariate observations should be supplied in `data`.
`variables`	character; smooth covariates in plotting order. Supply every covariate exactly once. The first two define surface axes and the remaining covariates define conditioning facets. Selected smooths must have the same covariates. If `NULL`, use the usual smooth plotting order.
`unweighted`	logical; for fitted models, return basis functions without multiplying by fitted coefficients? The default, `FALSE`, returns the coefficient-weighted contributions. With `TRUE`, `draw()` shows the shared basis once for each factor-by set.
`knots`	a list or data frame with named components containing knots locations. Names must match the covariates for which the basis is required. See `mgcv::smoothCon()`.
`constraints`	logical; should identifiability constraints be applied to the smooth basis. See argument `absorb.cons` in `mgcv::smoothCon()`.
`at`	a data frame containing values of the smooth covariate(s) at which the basis should be evaluated.
`diagonalize`	logical; if `TRUE`, reparameterises the smooth such that the associated penalty is an identity matrix. This has the effect of turning the last diagonal elements of the penalty to zero, which highlights the penalty null space.
`coefficients`	numeric; vector of values for the coefficients of the basis functions.

Details

When evaluating a fitted smooth at explicit data, missing covariates required by that smooth produce `NA` basis values. Missing values in other model variables do not discard otherwise evaluable rows.

The covariate order chosen by `variables` also determines which automatically generated axes use `n_2d`, `n_3d`, and `n_4d`. Supplied `data` or `at` values are unchanged. Set a resolution argument to `NULL` to use `n` instead.

Value

A tibble with class `basis`. Its `plot_variables` attribute stores covariate orders by smooth label for subsequent plotting. When the basis is shared across factor levels, `factor_by_groups` records the smooth labels that `draw()` combines. The returned rows retain each level.

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, ...)

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

## S3 method for class 'scam'
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.

See Also

gamm4_support

Examples

```
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 simulation observation error (i.e. not every bird present at a location would be detected). The data set (bird_move) consists of the variables count, latitude, week and species.

Format

A data frame

Source

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

boundary *Extract the boundary of a soap film smooth*

Description

[Experimental]

Usage

```
## S3 method for class 'gamm4'
boundary(x, ...)

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

gamm4_support

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

---

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

Description

Extract coefficients from a fitted scam model.

Usage

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

Arguments

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

`compare_smooths` *Compare smooths across models*

Description

Compare smooths across models

Usage

```
compare_smooths(
  model,
  ...,
  select = NULL,
  smooths = deprecated(),
  n = 100,
  n_2d = 50,
  data = NULL,
  unconditional = FALSE,
  overall_uncertainty = TRUE,
  partial_match = FALSE,
  envir = NULL,
  dist = NULL
)
```

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 a univariate smooth.
`n_2d`	numeric; the number of points along each of the first two axes of a smooth surface, including surface panels of higher-dimensional smooths. The default is 50 in plotting and plot-preparation functions. If `NULL`, use `n` instead. Ignored when evaluation `data` are supplied. Factor levels are retained, and curves with only one continuous covariate use `n`.
`data`	a data frame of covariate values at which to evaluate the smooth.

`unconditional` logical; if `TRUE` (and only if `frequentist == FALSE`) then the bayesian smoothing parameter uncertainty-corrected covariance matrix is returned, if available. Whether it is available depends on which smoothness selection method was used to fit the model.

`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.

`envir` an optional environment supplying functions and constants used in model expressions. The available model formula environment is used when `NULL`. Covariate observations should be supplied in `data`.

`dist` numeric; if greater than 0, this is used to determine when a location is too far from data to be plotted when plotting 2-D smooths. The data are scaled into the unit square before deciding what to exclude, and `dist` is a distance within the unit square. See `mgcv::exclude.too.far()` for further details.

Details

Two-variable smooths are evaluated at common locations. Automatic grids span the intersection of the models' observed covariate ranges; factor levels must overlap. Supplied `data` are evaluated as given, without filling gaps. Each selected smooth label must occur exactly once in every model and use the same covariates, including any `by` variable. Numeric and logical `select` refer to the primary model; matching labels are then used in every model.

Estimates are contributions of individual smooths to the linear predictor, on its link scale, with the centring constraints used in each fitted model. They exclude the intercept and other model terms. Different fitting data can change the centring, so a surface difference can include a constant shift. It is the difference between these fitted smooth contributions, rather than a difference between fitted responses.

The result retains one row per model and smooth, with estimates in the `data` list column. For distributional models, `.parameter` retains the label for the linear predictor, while `.model` identifies the input model. Use `draw.compare_smooths()` for faceted surface comparisons or estimated surface differences. Smooths with more than two covariates can be evaluated but cannot be drawn by this method.

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)

# Simulate a noisy response over two continuous covariates, x and z
surf <- data_sim("eg2", n = 100, seed = 7)
# Fit two small bases to the same surface data
coarse <- gam(y ~ s(x, z, k = 10), data = surf, method = "REML")
fine <- gam(y ~ s(x, z, k = 15), data = surf, method = "REML")
# Compare surfaces on shared axes and a shared colour scale
surfaces <- compare_smooths(coarse, fine, n_2d = 20)
draw(surfaces)
# Show coarse minus fine, with a colour scale centred on zero
draw(surfaces, show = "difference", pair = c("coarse", "fine"))
```

`conditional_differences` *Conditional differences of fitted values from a GAM*

Description

Compare full fitted values between pairs of factor levels, conditional on covariate values. Parametric effects and smooths are included unless excluded explicitly. Unlike `difference_smooths()`, these are differences of predictions, rather than differences of selected smooth functions.

Usage

```
conditional_differences(
  model,
  by,
  condition = NULL,
  data = NULL,
  scale = c("response", "link", "linear_predictor"),
  ...
)

## S3 method for class 'gam'
conditional_differences(
  model,
  by,
  condition = NULL,
  data = NULL,
  scale = c("response", "link", "linear_predictor"),
  n_vals = NULL,
  ci_level = 0.95,
```

```
  complete = TRUE,
  unconditional = FALSE,
  uncertainty = c("delta", "simulation"),
  n_sim = 10000,
  seed = NULL,
  n_cores = 1,
  method = c("gaussian", "mh", "inla", "user"),
  draws = NULL,
  mvn_method = c("mvnfast", "mgcv"),
  burnin = 1000,
  thin = 1,
  t_df = 40,
  rw_scale = 0.25,
  envir = NULL,
  ...,
  interval = c("confidence", "simultaneous"),
  simultaneous_scope = c("contrast", "all")
)

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

## S3 method for class 'scam'
conditional_differences(model, ..., unconditional = FALSE)
```

Arguments

model	a fitted GAM object.
by	character vector naming factors that define comparison groups. For multiple factors, groups are joint combinations of their levels and all pairs of groups are compared. To compare one factor within levels of another, put the latter in condition instead.
condition	character vector or list specifying evaluation covariates, as in conditional_values(). The first non-comparison covariate supplies the plotting axis. Up to four non-comparison conditions are supported. Named elements for factors in by restrict their comparison levels, e.g. list("x", treatment = c("control", "high")). Other covariates are held at typical_values(). At least one non-comparison condition is required. Group two numeric covariates in the first non-comparison element to request a surface, for example list(c("x1", "x2"), "stratum"). A nested list can supply explicit axis values or helpers, as in conditional_values(). Both axes count towards the four-covariate limit. Remaining conditions supply facets, skipping categorical colour.
data	optional data frame used to generate condition values and determine covariate classes. Explicit values in condition take precedence. Other covariates use the fitted model's typical values. The rows of data are not used directly as a prediction grid.
scale	character; which scale should predictions be returned on?

`...`	arguments passed to prediction, such as `exclude`. Arguments `type`, `newdata`, `se.fit`, and `terms` are not supported here.
`n_vals`	positive integer or `NULL`; number of values to generate along each numeric axis. For `conditional_values()` and `conditional_differences()`, `NULL` (the default) uses 100 for curves and 50 per axis for surfaces. Explicit values or helpers in `condition` override generated values. Other numeric conditions use their five-number summaries unless explicit values or helpers are supplied.
`ci_level`	numeric; a number on interval (0,1) giving the coverage for credible intervals.
`complete`	logical; include all combinations of comparison and conditioning factors? With `FALSE`, retain only combinations observed in the fitted data, and compare groups only within shared conditioning strata. This does not restrict numeric grids to overlapping observed ranges.
`unconditional`	logical; include smoothing-parameter uncertainty in the Bayesian covariance, if available, for delta intervals and Gaussian draws. This argument does not modify MH or user-supplied draws.
`uncertainty`	character; `"delta"` uses analytic link-scale or delta-method response-scale standard errors and normal intervals. `"simulation"` uses shared posterior coefficient draws, with equal-tailed quantiles for pointwise intervals or calibrated symmetric simultaneous bands.
`n_sim`	integer; number of draws for posterior simulation or calibration of delta-method simultaneous intervals. Ignored for pointwise delta intervals and for simulation with `method = "user"`, which uses all supplied draws.
`seed`	integer or `NULL`; seed for posterior simulation or delta-method simultaneous calibration. An explicit seed preserves the caller's RNG state. With `NULL`, delta-method calibration advances the current RNG state; posterior simulation follows `fitted_samples()` seed handling.
`n_cores`	integer; number of cores used by `mvnfast::rmvn()` for delta-method simultaneous calibration, or passed to `fitted_samples()`.
`method`	character; posterior sampler used by `fitted_samples()`. `"gaussian"` uses the Gaussian posterior approximation; `"mh"` uses Metropolis-Hastings; `"user"` uses supplied coefficient draws. `"inla"` is reserved by the sampling machinery but is not implemented.
`draws`	matrix of posterior coefficient draws, one draw per row in model coefficient order, used with `method = "user"`. At least two are required.
`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.
`burnin`	numeric; the length of any initial burn in period to discard. See `mgcv::gam.mh()`.
`thin`	positive integer; retain every `thin` iterations after burn-in.
`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()`.

envir — an optional environment supplying functions and constants used in model expressions. The available model formula environment is used when NULL. Covariate observations should be supplied in data.

interval — character; "confidence" (the default) gives pointwise intervals; "simultaneous" gives simultaneous intervals over the evaluated covariate combinations.

simultaneous_scope — character; "contrast" calibrates a separate band for each pair, jointly over all its evaluation points and conditioning strata. "all" calibrates one band jointly over every returned comparison and evaluation point. Ignored for pointwise intervals.

Details

Differences are group 1 minus group 2, with groups ordered by model factor levels. Both groups use the same values of all other covariates. On the response scale the inverse link is applied to each prediction before subtraction. Shared terms can therefore affect response-scale differences even when they cancel on the link scale.

Intervals describe uncertainty in conditional mean differences, not observation noise. Simulation uses fitted_samples() once for the entire prediction grid, retaining dependence between predictions by subtracting within draws. The estimate is always the difference of fitted predictions. Pointwise simulation intervals need not be centred on this estimate.

Simultaneous intervals use the ci_level quantile of the maximum absolute standardized deviation over the selected coverage set. They are symmetric about .diff, with half-width .crit * .se. For uncertainty = "delta", one shared batch of zero-mean Gaussian coefficient deviations is generated from the Bayesian covariance matrix. Response-scale differences use a first-order Taylor approximation. The covariance must be positive semidefinite; singular covariance matrices are supported. Sampler arguments method, draws, mvn_method, burnin, thin, t_df, and rw_scale apply only to uncertainty = "simulation".

For uncertainty = "simulation", deviations are posterior difference draws minus .diff, standardized by their sample standard deviations. Draws are transformed to the response scale before differencing; invalid inverse-link domains or response means cause an error rather than discarded draws. Zero-variance draws that disagree with .diff cannot calibrate a band and cause an error. Deterministic rows agreeing with .diff have zero width.

Simultaneous coverage is approximate posterior coverage on the finite evaluation grid, not between its points or outside its range, nor exact frequentist coverage. Increase n_vals to evaluate a denser grid. Bands are not clipped to the response-difference range (e.g. [-1, 1] for probabilities). They need not contain the equal-tailed pointwise simulation intervals.

exclude sets the named term contributions to zero. In particular, excluding random effects does not integrate over their distribution. Offsets follow mgcv::predict.gam() semantics. Supported models are gam and bam fits with a single linear predictor and an ordinary scalar inverse-link mean (including negative binomial, Tweedie, beta regression and scaled t families). SCAM fits with supported single-predictor families also support link- and response-scale differences, with delta or simulation uncertainty and pointwise or simultaneous intervals. SCAM delta intervals use the local covariance of transformed coefficients. Gaussian simulation draws unrestricted fitting coefficients and applies the fitted positivity transformation; user draws must already be in transformed coordinates and satisfy the coefficient constraints. MH sampling is unavailable for SCAM fits, and smoothing-parameter uncertainty is unsupported; use unconditional = FALSE.

Value

A tibble of class "conditional_differences", with .contrast, .diff, .se, .lower_ci, .upper_ci, and evaluation covariates. .se is on the requested scale; for simulation it is the standard deviation of difference draws. A single comparison factor has .level_1 and .level_2 columns. Multiple factors have <factor>_1 and <factor>_2 columns. Attributes record by, condition, scale, uncertainty, ci_level, exclude, interval, simultaneous_scope, and plotting channels. Simultaneous results also contain .crit, the calibrated critical value, constant within each coverage set.

See Also

gamm4_support

Examples

```
load_mgcv()
# Simulate group-specific curves with Gaussian noise and fit factor-by smooths
df <- data_sim("eg4", n = 300, seed = 2)
m <- gam(y ~ fac + s(x2, by = fac) + s(x0), data = df, method = "REML")

# Compare fitted responses over x2, including the parametric group effects
cd <- conditional_differences(m, by = "fac", condition = "x2")
cd
draw(cd)
```

---

conditional_moments *Conditional moments of a GAM response distribution*

Description

Construct covariate combinations using conditional_values() conventions, then estimate response moments with fitted_moments(). Other covariates are held at typical values; observations are not averaged over those covariates.

Usage

```
conditional_moments(
  model,
  condition = NULL,
  data = NULL,
  moments = c("mean", "variance", "cv"),
  n_vals = 100,
  complete = TRUE,
  ...
)

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

```
  model,
  condition = NULL,
  data = NULL,
  moments = c("mean", "variance", "cv"),
  n_vals = 100,
  complete = TRUE,
  ...
)
```

Arguments

model	a fitted GAM object.
condition	Character vector or list specifying covariates and optional values. The first covariate supplies the plotting axis, followed by colour, facet columns and facet rows. Grouped surface conditions are not supported.
data	optional data frame used to generate condition values and determine covariate classes. Explicit values in condition take precedence. Other covariates use the fitted model's typical values. The rows of data are not used directly as a prediction grid.
moments	Character vector selecting "mean", "variance", "sd", "cv", "skewness", "kurtosis", or "excess_kurtosis". "all" requests all seven. CV is SD divided by a strictly positive mean; kurtosis is the fourth central moment divided by squared variance.
n_vals	Positive integer; number of values to generate along a numeric plotting axis. Defaults to 100. Explicit condition values or helpers override generated values; other numeric conditions use five-number summaries.
complete	logical; if TRUE (the default), retain all combinations of conditions. If FALSE, retain only combinations of factor levels among the variables named in condition that occur in the fitted model's data. Numeric conditions are not filtered, and variables absent from condition remain at their typical values. Supplied data does not change which combinations count as observed. An error is raised if no combinations remain.
...	Arguments to fitted_moments(), including uncertainty, nuisance, n_draws, seed, scores, and exclude.

Value

A tibble of class "conditional_moments", with the columns returned by fitted_moments() and the plotting channels of conditional_values().

Examples

```
load_mgcv()
m <- gam(mpg ~ s(wt, k = 5), data = mtcars, method = "REML")
conditional_moments(m, condition = "wt", moments = c("mean", "variance")) |>
  draw()
```

`conditional_values` *Conditional predictions from a GAM*

Description

Generate predicted values from a GAM, conditional upon supplied values of covariates. `conditional_values()` is modelled after `marginaleffects::plot_predictions()`, but with an intentionally simpler, more restrictive functionality. The intended use case is for quickly visualizing predicted values from a fitted GAM on the response scale. For more complex model predictions, you are strongly encouraged to use `marginaleffects::plot_predictions()`.

Usage

```
conditional_values(
  model,
  condition = NULL,
  data = NULL,
  scale = c("response", "link", "linear_predictor"),
  ...
)

## S3 method for class 'gam'
conditional_values(
  model,
  condition = NULL,
  data = NULL,
  scale = c("response", "link", "linear_predictor"),
  n_vals = NULL,
  ci_level = 0.95,
  complete = TRUE,
  ...
)

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

## S3 method for class 'scam'
conditional_values(model, ..., unconditional = FALSE)
```

Arguments

`model`	a fitted GAM object.
`condition`	either a character vector or a list supplying the names of covariates, and possibly their values, to condition on. The order of the values determines how these are plotted via the `draw.conditional_values()` method; the first element is mapped to the *x* channel, the second element to the *colour* channel, the third to `ggplot2::facet_wrap()` **if no fourth element is present**, if present, the fourth

element is mapped to the rows and the third element is mapped to the columns of `ggplot2::facet_grid()`. To request a surface, group two numeric covariates in the first element, for example `list(c("x1", "x2"), "treatment")`. The grouped covariates supply the horizontal and vertical axes, in that order. The remaining elements supply facet columns and rows; there is no categorical colour channel. Use a nested list to supply axis values or helpers, for example `list(list(x1 = c(0, 0.5, 1), x2 = "quartile"), "treatment")`. At most four covariates are allowed, counting both surface axes.

`data` optional data frame used to generate condition values and determine covariate classes. Explicit values in `condition` take precedence. Other covariates use the fitted model's typical values. The rows of `data` are not used directly as a prediction grid.

`scale` character; which scale should predictions be returned on?

`...` arguments passed to `fitted_values()`.

`n_vals` positive integer or `NULL`; number of values to generate along each numeric axis. For `conditional_values()` and `conditional_differences()`, `NULL` (the default) uses 100 for curves and 50 per axis for surfaces. Explicit values or helpers in `condition` override generated values. Other numeric conditions use their five-number summaries unless explicit values or helpers are supplied.

`ci_level` numeric; a number on interval (0,1) giving the coverage for credible intervals.

`complete` logical; if `TRUE` (the default), retain all combinations of conditions. If `FALSE`, retain only combinations of factor levels among the variables named in `condition` that occur in the fitted model's data. Numeric conditions are not filtered, and variables absent from `condition` remain at their typical values. Supplied `data` does not change which combinations count as observed. An error is raised if no combinations remain.

`unconditional` logical; include smoothing parameter uncertainty in the Bayesian covariance matrix, if available. Otherwise a warning is issued and the uncorrected covariance is used.

Value

A data frame (tibble) of class `"conditional_values"`.

Author(s)

Gavin L. Simpson

See Also

gamm4_support

Examples

```
load_mgcv()
# Simulate an additive Gaussian response and fit a GAM
df <- data_sim("eg1", n = 200, seed = 2)
m <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = df, method = "REML")
```

```
# Vary x2 while holding the other covariates at representative values
cv <- conditional_values(m, condition = "x2")
cv
draw(cv)
```

`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. If missing, all parameters are considered.
`level`	numeric, `0 < level < 1`; the confidence level of the point-wise or simultaneous interval. The default is `0.95` for a 95% interval. It must be finite. If a vector is supplied, the first value is used with a warning.
`type`	character; the type of interval to compute. One of `"confidence"` for point-wise intervals, or `"simultaneous"` for simultaneous intervals.
`nsim`	integer; the number of simulations used in computing the simultaneous intervals.
`ncores`	number of cores for generating random variables from a multivariate normal distribution. Passed to `mvnfast::rmvn()`. Parallelization will take place only if OpenMP is supported (but appears to work on Windows with current `R`).
`...`	additional arguments for methods

Value

a data frame with components:

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

Author(s)

Gavin L. Simpson

Examples

```
load_mgcv()

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

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

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

## simultaneous interval for smooth term of x2

x2_sint <- confint(fd,
  parm = "x2", type = "simultaneous",
  nsim = 10000, ncores = 2
)

x2_sint
```

confint.gam *Point-wise and simultaneous confidence intervals for smooths*

Description

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

Usage

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

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

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

## S3 method for class 'scam'
confint(
  object,
  parm = smooths(object),
  ...,
  unconditional = FALSE,
  overall_uncertainty = FALSE
)
```

Arguments

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

parm — which parameters (smooth terms) are to be given intervals as a vector of terms. If missing, all parameters are considered, although this is not currently implemented.

level — numeric, `0 < level < 1`; the confidence level of the point-wise or simultaneous interval. The default is `0.95` for a 95% interval.

data — data frame; 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. `"credible"` aliases `"confidence"`, for approximate Bayesian credible intervals using the default covariance.

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

shift — logical; should the intercept of the smooth's linear predictor be added to the smooth? Predictors without an intercept are shifted by zero. Only the intercept is added; other parametric effects, including factor effects associated with a factor by smooth, are excluded. For full fitted values and their confidence intervals, use `fitted_values()`.

transform — logical; should the smooth be evaluated on a transformed scale? For generalised models, this involves applying the inverse of the link function used to fit the model. Alternatively, the name of, or an actual, function can be supplied to transform the smooth and it's confidence interval.

unconditional — logical; if `TRUE` (and only if `frequentist == FALSE`) then the bayesian smoothing parameter uncertainty-corrected covariance matrix is returned, if available. Whether it is available depends on which smoothness selection method was used to fit the model.

frequentist — logical; if `FALSE`, the default, the bayesian covariance matrix is returned, otherwise the frequentist covariance matrix.

overall_uncertainty — logical; should the uncertainty in the model constant term be included in the standard error of the evaluate values of the smooth?

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

partial_match — logical; should matching `parm` use a partial match or an exact match? Can only be used if `length(parm)` is `1`.

... — additional arguments passed to `smooth_estimates()`, including `interval_method` to select Gaussian, stored scasm bootstrap, or shape-enforced intervals, and `shape_source` to choose their initial source. `simultaneous_method`, `simultaneous_domain`, and `tube_control` request and configure continuous-interval tube bands; see simultaneous_bands.

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.

Details

For SCAM fits, `confint()` returns intervals for smooth functions, with `parm` selecting smooth labels. It uses transformed-coefficient Gaussian uncertainty conditional on the fitted smoothing parameters. By default, intercept uncertainty is excluded. These intervals do not enforce shape constraints on their limits.

Value

a tibble with components:

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

Note

`overall_uncertainty` applies to both pointwise and simultaneous intervals. Each simultaneous band covers the evaluation points of one fitted smooth by default, or its continuous domain with tube calibration; separate smooths do not form a joint coverage set. Native scasm fits default to `overall_uncertainty = FALSE` and use stored bootstrap draws for pointwise intervals when available, as in `smooth_estimates()`. Shifting and transforming a smooth preserves its interval limits, including asymmetric bootstrap limits. Shape-enforced intervals describe unshifted smooth components on the model scale; `shift = TRUE` and effect transformations are unsupported.

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, data = NULL, ...)

## S3 method for class 'gamm4'
data_combos(object, ...)
```

Arguments

`object`	a fitted model object.
`...`	arguments passed to methods.
`vars`	terms to include or exclude from the returned object. Uses tidyselect principles.
`complete`	logical; should all combinations of factor levels be returned? If `FALSE`, only those combinations of levels observed in the model are retained.
`data`	an optional data frame supplying covariate classes. By default, these are taken from the stored model frame; typical values come from the fitted model's covariate summaries.

See Also

gamm4_support

`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. The sampling distribution can be applied to all the example types.

Usage

```
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"),
  family = NULL,
  predictors = NULL,
  parameters = list(),
  observation = NULL
)
```

Arguments

`model`	character; the name of a model, such as `"additive"` or its original name `"eg1"`. See Details for all names and aliases.
`n`	numeric; the number of observations to simulate.
`scale`	numeric; for normal responses, the standard deviation; for Poisson and negative binomial responses, a multiplier of the true function; for binary responses, a multiplier after subtracting 5; for Gamma and Tweedie responses, the dispersion. Ignored for ordinal responses. Legacy numeric vectors are recycled across observations. Use `parameters` with the family-object interface.
`theta`	numeric; the dispersion parameter θ 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 `dist %in% c("ocat", "ordered categorical")`.

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 as per dist.
family	an instantiated family object, such as poisson() or mgcv::gaulss(). Cannot be combined with explicitly supplied dist, scale, theta, power, n_cat, cuts or gfam_families. See data_sim_families for coverage and generating parameters.
predictors	NULL or a function of the generated data, including raw true-function columns. Return a finite numeric vector for one linear predictor or a matrix with one row per observation and one column per linear predictor, in the family's order. These values are on the link scale. NULL uses calibrated example predictors; see data_sim_families.
parameters	a named list of natural-scale generating parameters. Parameters fixed in the family constructor are used directly. Fitting starting values are not generating parameters: supply the corresponding entries explicitly. See data_sim_families for names and defaults.
observation	NULL or a named list with left, right and/or interval_width, for censored families. Limits and widths have length one or n. Complete responses are generated before censoring. See data_sim_families for response encodings.

Details

data_sim() can simulate data from several underlying models of known true functions. Descriptive names and their original short names are interchangeable, giving identical output for the same arguments and seed. The available options currently are:

- "additive" or "eg1": a four term additive true model. This is the classic Gu & Wahba four univariate term test model. See gw_functions for more details of the underlying four functions.
- "bivariate_surface" or "eg2": a bivariate smooth true model.
- "varying_coefficient" or "eg3": an example containing a continuous by smooth (varying coefficient) true model. The model is $\hat{y}_i = f_2(x_{2i})x_{1i}$ where the function $f_2()$ is $f_2(x) = 0.2 * x^{11} * (10 * (1 - x))^6 + 10 * (10 * x)^3 * (1 - x)^{10}$.
- "factor_by" or "eg4": a factor by smooth true model. The true model contains a factor with 3 levels, where the response for the *n*th level follows the *n*th Gu & Wahba function (for $n \in 1, 2, 3$).
- "additive_factor" or "eg5": an additive plus factor true model. The response is a linear combination of the Gu & Wahba functions 2, 3, 4 (the latter is a null function) plus a factor term with four levels.
- "additive_group_effects" or "eg6": an additive model plus four group intercepts with fixed realised values 3, 6, 9, 12.
- "additive_correlated" or "eg7": a version of the model in "eg1", but where the covariates are correlated.

- "gu_wahba_f2" or "gwf2": a model where the response is Gu & Wahba's $f_2(x_i)$ plus noise.
- "luo_wahba_f6" or "lwf6": a model where the response is Luo & Wahba's "example 6" function $sin(2(4x-2)) + 2exp(-256(x-0.5)^2)$ 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 the distributions allowed by dist, including repeated names. Defaults use Gaussian standard deviation 0.5, Gamma dispersion 0.5, Tweedie dispersion 1.5, negative binomial size 3 and Tweedie power 1.5. Historical grouped ordinal cuts are c(1, 2, 2.5). Explicit scale, theta, power, n_cat and cuts override these settings. A supplied scale is the Gaussian standard deviation and the Gamma or Tweedie dispersion. Identity, logit and log links are used for normal, binary and positive-mean responses, respectively. Ordinal responses use a logistic latent variable.

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.

Value

A tibble containing responses, covariates and true-function columns. Legacy f is the predictor after its distribution-specific transformation; component columns retain their original function values. With family, f is the first linear predictor, y can be a matrix column, and the "truth" attribute contains eta (all linear predictors), parameters (natural distribution parameters), functions (the original template), response (complete responses before censoring), and family. Grouped families also record index and component-specific parameter lists. Binomial family objects return proportions and a trials column for fitting weights. Cox models return observed times and a status column. An explicit seed restores the caller's RNG state, including its absence; seed = NULL advances the caller's RNG normally.

Shape-constrained examples

Models named "sc_increasing", "sc_decreasing", "sc_convex", "sc_concave", "sc_positive", "sc_increasing_convex", "sc_increasing_concave", "sc_decreasing_convex" and "sc_decreasing_concave" provide univariate reference shapes on [0, 1]. "sc_additive" combines four terms; "sc_tensor" is increasing in two coordinates; "sc_location_scale" has a concave log-mean and increasing log dispersion. The latter defaults to family = mgcv::gammals(); other shape templates default to Gaussian noise with standard deviation 0.2. See data_sim_shapes for functions, derivatives and mgcv::scasm() recipes.

References

Gu, C., Wahba, G., (1993). Smoothing Spline ANOVA with Component-Wise Bayesian "Confidence Intervals." *J. Comput. Graph. Stat.* **2**, 97–117.

Luo, Z., Wahba, G., (1997). Hybrid adaptive splines. *J. Am. Stat. Assoc.* **92**, 107–116.

See Also

data_sim_families, data_sim_shapes

Examples

```
data_sim("additive", n = 100, seed = 1)

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

`data_sim_families` *Generating responses with mgcv family objects*

Description

`data_sim()` accepts instantiated standard and mgcv family objects. The example template supplies covariates and true functions; `predictors` supplies the linear predictors; `parameters` supplies generating values that are not themselves modelled by a predictor. The `"truth"` attribute records all predictors, distribution parameters and complete responses.

Scalar families

The parameter names below belong inside `parameters = list(...)`. Parameters are scalar unless explicitly described as observation-level. Beta draws use the constructor's `eps` endpoint adjustment, matching mgcv's generator. This adjustment can matter when the beta law places substantial probability very close to zero or one; `eps` is recorded in the truth.

Family	Generating parameters
`gaussian()`	`dispersion` = 1, the non-negative variance
`poisson()`, `cpois()`	None
`binomial()`	`trials` = 1, positive integer trial counts
`Gamma()`, `inverse.gaussian()`	`dispersion` = 1
`nb()`, `negbin()`	`theta`, positive size; `Inf` gives the Poisson limit
`tw()`, `Tweedie()`	`power` in (1, 2), and `dispersion` = 1
`betar()`	Positive `precision`
`scat()`	`df` above the family's `min.df`, and positive t scale `sigma`
`ocat()`	Finite increasing `cuts`, one fewer than the categories
`ziP()`	`intercept` and natural-scale `slope` above the constructor's `b`
`cnorm()`, `clog()`	Positive `sigma`, Gaussian SD or logistic scale
`bcg()`	`lambda` = `0` and positive log-response SD `sigma`

Dispersion and trials can have length one or `n`. All other scalar-family parameters are constant across observations. Fixed constructor parameters are used when available; conflicting explicit parameters are errors. Otherwise the corresponding generating values are required, including cut points for `ocat(R = ...)`. For example, both `family = mgcv::nb(theta = 3)` and `family = mgcv::nb(), parameters = list(theta` specify size 3. Constructor starting values for an estimated parameter are not used as generating truth. A size range in `negbin(c(1, 10))` also requires an explicit generating `theta`.

For `ziP()`, `lambda = exp(eta)` and `P(Y > 0) = 1 - exp(-exp(intercept + slope * eta))`. Positive observations follow a Poisson distribution conditioned on being positive. The natural `slope` includes the constructor's lower bound `b`; the second entry of `ziP(theta = ...)` is instead a log excess above `b`. For fixed ordinal spacings, mgcv fixes the first cut at -1; for example `ocat(theta = c(1, 5))` supplies cuts `c(-1, 0, 5)`. For free cuts whose first value is `a`, mgcv's fixed first cut of -1 represents the same probabilities after shifting the predictor by `-(a + 1)`. Include the corresponding component intercept in grouped fitting recipes.

Quasi families do not define a complete sampling distribution and are rejected. Nonzero Box-Cox powers are currently rejected: some Gaussian transformed responses would lie outside the inverse transformation's domain.

Families with several linear predictors

The columns returned by `predictors` follow mgcv's order and links, including their constructor-specific bounds. These are link-scale predictors, not the `mu` arguments of mgcv's random-number generators.

Family	Predictor order and natural parameters recorded in truth
`gaulss()`	Mean, SD predictor; records `mu`, `sigma`
`gammals()`	Log mean, log-dispersion predictor; records `mu`, `dispersion`
`gumbls()`	Location, log-scale predictor; records `location`, `mu` (mean), `sigma`
`gevlss()`	Location, log scale, shape predictor; records `location`, `sigma`, `shape`
`shash()`	Location, scale predictor, skewness, log tail; records `location`, `sigma`, `skewness`, `tail`
`twlss()`	Mean predictor, power predictor, log dispersion; records `mu`, `power`, `dispersion`
`ziplss()`	Log Poisson intensity, complementary log-log presence predictor; records `lambda`, `presence`
`multinom(K)`	K logits relative to category 0; records category `probabilities`
`mvn(d)`	d response means; `parameters$covariance` defaults to the identity matrix

These families need no scalar generating parameters except `mvn()`'s optional positive definite covariance matrix. In `shash()`, the transformed normal is multiplied by `tail * sigma`, following mgcv's parameterisation. In `gumbls()`, the mean is location plus Euler's constant times `sigma`; the first predictor specifies the location. For `gevlss()`, the default shape link restricts shape to (-1, 0.5); the identity link allows any finite shape. In `ziplss()`, positive counts follow a zero-truncated Poisson distribution.

Default predictors

Let g be the template's `f` before distribution transformations. Scalar Gaussian, t, Gaussian-censoring, logistic-censoring and Box-Cox families use natural location g. Binomial and beta means are

plogis((g - 5) / 5); ordinal location is g - mean(g); the ziP() predictor is g / 5. Other scalar means are exp(g / 5). Each mean is transformed with the supplied family's link. If it is outside that link's domain, supply custom predictors rather than relying on a default.

For general families, default predictors are: (g, 0) for gaulss() and gumbls(); (g / 5, 0) for gammals() and ziplss(); (g, 0, linkfun(0)) for gevlss() (zero shape under the supplied shape link); (g, 0, 0, 0) for shash(); and (g / 5, 0, 0) for twlss(). Multivariate normal means repeat g. Multinomial logit j is (g - mean(g)) / 5 * j / K. The sc_location_scale template instead supplies its own log mean and log dispersion for gammals().

Censoring and survival

For cnorm(), clog(), cpois() and bcg(), observation can contain finite left and right detection limits and a positive interval_width. Responses outside detection limits are left or right censored. Other responses are placed in rounding intervals wholly inside those limits; values in intervals crossing a limit are retained exactly. Omitting observation returns complete responses.

Matrix responses use equal entries for exact observations, (limit, -Inf) for left censoring, (limit, Inf) for right censoring, and (lower, upper) for intervals. bcg() instead uses zero in the second entry for left censoring; its limits must be positive. Poisson detection limits must be non-integer, and interval widths must be integer numbers of counts. Its intervals use half-integer endpoints to avoid ambiguity at count values.

cox.ph() uses Weibull cumulative baseline hazard H0(t) = (t / baseline_scale)^baseline_shape, with both baseline parameters defaulting to 1. Event times have survival exp(-H0(t) * exp(eta)). Independent exponential censoring has censor_rate = 0.2; zero disables censoring. Default eta is (g - mean(g)) / 5. Fit using weights = status. True event and censoring times are retained in the truth parameter list.

Grouped families

Supply family = mgcv::gfam(list(...)), optionally with parameters = list(components = list(...), index = ...). components contains one parameter list per family-list position. index is one integer position per row (a scalar selects the same family for every row). Without index, every component is represented at least once and remaining positions are sampled uniformly; n must be at least the number of families. Repeated family types retain separate indices and parameters.

The default shared predictor is g / 5; custom predictors return one column. Scalar response families are supported; general families are rejected. Censored-family components return complete, uncensored responses. Fit using cbind(y, index) as the response and weights = trials if a component is binomial with more than one trial. Row-level dispersion and trial vectors in component parameter lists align with the full generated data, or can have the component's row count. Truth parameters record each component's rows.

See Also

data_sim(), data_sim_shapes, mgcv::family.mgcv()

Examples

```
# A beta response: precision is explicitly chosen, rather than estimated.
d <- data_sim(family = mgcv::betar(theta = 12), seed = 2)
```

```
attr(d, "truth")$parameters$precision

# Binomial proportions: use trials as the fitting weights.
d <- data_sim(family = binomial(), parameters = list(trials = 10), seed = 2)

# Mean and standard deviation have separate predictors.
d <- data_sim("gwf2", family = mgcv::gaulss(), seed = 2,
  predictors = function(d) cbind(d$f, log(0.2 + d$x)))

# Fit a censored normal model to the matrix response.
d <- data_sim("sc_increasing", family = mgcv::cnorm(theta = 0.2), seed = 2,
  observation = list(left = 0.1, right = 0.9))

m <- mgcv::gam(y ~ s(x), family = mgcv::cnorm(), data = d, method = "REML")
```

`data_sim_shapes` *Reference functions for shape-constrained GAMs*

Description

`data_sim()` supplies analytic reference functions for `mgcv::scasm()`. All covariates are uniform on `[0, 1]`. Univariate templates return `f1`, the true function, and its analytic derivatives `d1` and `d2`. The response contains sampling variation; the derivative columns describe the truth.

Univariate templates

Model suffix after `sc_`	Function	`xt` in `s(x, bs = "sc", xt = ...)`
`increasing`	`plogis(8 * (x - 0.5))`	`"m+"`
`decreasing`	`1 - plogis(8 * (x - 0.5))`	`"m-"`
`convex`	`(x - 0.5)^2`	`"c+"`
`concave`	`1 - (x - 0.5)^2`	`"c-"`
`positive`	`0.2 + exp(-16 * (x - 0.5)^2)`	`"+"`, with no intercept
`increasing_convex`	`exp(2 * x) / exp(2)`	`c("m+", "c+")`
`increasing_concave`	`log1p(4 * x) / log(5)`	`c("m+", "c-")`
`decreasing_convex`	`exp(-2 * x)`	`c("m-", "c+")`
`decreasing_concave`	`1 - x^2`	`c("m-", "c-")`

Remove the intercept for the positivity recipe so the constrained smooth represents the whole function. An unconstrained intercept combined with a centred smooth does not express the same positivity requirement.

Several terms and distribution parameters

`sc_additive` adapts mgcv's example. Its raw terms are `gw_f0(x0)`, `2 * plogis(20 * (x1 - 0.4))`, `gw_f2(x2)` and `10 * (x3 - 0.55)^4`. The true predictor is their sum minus 1, divided by 4. The

second term is increasing and the fourth is convex. Use `predictors = function(d) d$f` with `family = poisson()` to sample mgcv's Poisson example.

`sc_tensor` uses `f = x + z + x * z`, increasing in both covariates. Its `dx` and `dz` columns give the true partial derivatives.

`sc_location_scale` uses log mean `sin(pi * x) / 2` and log dispersion `-2 + z`, returned in `f` and `f_scale`. It defaults to `gammals()` and records derivatives in `d1`, `d2` and `d_scale`. Constraints act on the specified predictors; in particular the concavity concerns the log mean.

The recipes use the default `bs = 0` for fitting without bootstrap replicates. To obtain native constraint-respecting bootstrap coefficient draws, fit with a positive `bs`, such as `bs = 100`.

See Also

`data_sim()`, data_sim_families, `mgcv::scasm()`

Examples

```
load_mgcv()
d <- data_sim("sc_increasing", n = 200, seed = 2)
m <- scasm(y ~ s(x, bs = "sc", xt = "m+", k = 8), data = d)

d <- data_sim("sc_positive", n = 200, seed = 2)
m <- scasm(y ~ s(x, bs = "sc", xt = "+", k = 8) - 1, data = d)


d <- data_sim("sc_additive", n = 240, family = poisson(), seed = 2,
  predictors = function(d) d$f)
m <- scasm(y ~ s(x0, k = 6) + s(x1, bs = "sc", xt = "m+", k = 6) +
  s(x2, k = 6) + s(x3, bs = "sc", xt = "c+", k = 6),
  family = poisson(), data = d)

d <- data_sim("sc_tensor", seed = 2)
m <- scasm(y ~ te(x, z, bs = c("sc", "sc"), xt = list("m+", "m+"),
  k = c(6, 6)), data = d)

d <- data_sim("sc_location_scale", seed = 2)
m <- scasm(list(y ~ s(x, bs = "sc", xt = "c-"),
  ~s(z, bs = "sc", xt = "m+")), family = gammals(), data = d)
```

`data_slice` *Prepare a data slice through model covariates*

Description

Prepare a data slice through model covariates

Usage

```
data_slice(object, ...)

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

## S3 method for class 'data.frame'
data_slice(object, ..., .observed_only = FALSE, .by = NULL)

## S3 method for class 'gam'
data_slice(
  object,
  ...,
  data = NULL,
  envir = NULL,
  .observed_only = FALSE,
  .by = NULL
)

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

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

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

Arguments

`object`	an R model object.
`...`	<dynamic-dots> User supplied variables defining the data slice. Arguments passed via `...` need to be *named*.
`.observed_only`	logical or character; should the data slice be trimmed to those combinations of the variables specified that are observed in `object`. If `TRUE`, the observed combinations of variables mentioned in `...` are matched against those in `object` and filtered to return only those combinations. If `FALSE`, no filtering is done. If `.observed_only` is a character vector, on those variables named in the vector are used to in the comparison with the combinations in `object`.
`.by`	<tidy-select> Optional grouping variables. With `.by = NULL` (the default), expressions are evaluated over the full data set. Otherwise, expressions are evaluated separately within each observed group and the resulting grids are combined. Grouping variables are included automatically; expressions for these variables in `...` select groups and are evaluated once over the full data set. Unobserved combinations are omitted, and selecting no observed groups is an error.
`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`.

Details

A data slice is the data set that results where one (or more covariates) is varied systematically over some or all of its (their) range or at a specified subset of values of interest, while any remaining covariates in the model are held at fixed, representative values. This is known as a *reference grid* in package **emmeans** and a *data grid* in the **marginaleffects** package.

For GAMs, any covariates not specified via `...` will take representative values determined from the data used to fit the model as follows:

- for numeric covariates, the value in the fitting data that is closest to the median value is used,
- for factor covariates, the modal (most frequently observed) level is used, or the first level (sorted as per the vector returned by `base::levels()` if several levels are observed the same number of times.

These values are already computed when calling `gam()` or `bam()` for example and can be found in the `var.summary` component of the fitted model. Function `typical_values()` will extract these values for you if you are interested.

Convenience functions `evenly()`, `ref_level()`, and `level()` are provided to help users specify data slices. `ref_level()`, and `level()` also ensure that factor covariates have the correct levels, as needed by `mgcv::predict.gam()` for example.

For an extended discussion of `data_slice()` and further examples, see `vignette("data-slices", package = "gratia")`.

With `.by`, unspecified covariates retain their overall representative values. Specify an expression such as `z = mean(z)` to use a group-specific value instead. Factor levels and ordered status are preserved, and the result is an ungrouped tibble. `.observed_only` filters exact matches within each group; interpolated values need not match observed values.

Grouping limits the default range of `evenly()` to each group's marginal range. It does not ensure joint support for multiple continuous covariates. Explicit values or `lower` and `upper` bounds can still request extrapolation.

See Also

The convenience functions `evenly()`, `ref_level()`, and `level()`. `typical_values()` for extracting the representative values used for covariates in the model but not named in the slice.

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))
```

`derivative_samples` *Posterior expectations of derivatives from an estimated model*

Description

Posterior expectations of derivatives from an estimated model

Usage

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

derivative_samples(model, ...)

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

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

## S3 method for class 'gam'
derivative_samples(
  model,
  ...,
  data = NULL,
  n_draws = NULL,
  chains = NULL,
  seed = NULL,
  method = c("gaussian", "mh", "inla", "user"),
  n_cores = 1L,
  n_threads = 1L,
  burnin = 1000,
  thin = 1L,
  t_df = 40,
```

```
  rw_scale = 0.25,
  frequentist = FALSE,
  unconditional = FALSE,
  draws = NULL,
  diagnostics = FALSE,
  mh_backend = c("mgcv", "gratia", "family"),
  mvn_method = c("mvnfast", "mgcv"),
  block_size = 10000L,
  dense_limit = 5e+06,
  focal = NULL,
  order = 1L,
  type = c("forward", "backward", "central"),
  scale = c("response", "linear_predictor"),
  n_grid = 200L,
  eps = NULL,
  envir = environment(formula(model))
)

## S3 method for class 'scam'
derivative_samples(
  model,
  ...,
  data = NULL,
  n_draws = NULL,
  chains = NULL,
  seed = NULL,
  method = c("gaussian", "mh", "inla", "user"),
  n_cores = 1L,
  n_threads = 1L,
  burnin = 1000,
  thin = 1L,
  t_df = 40,
  rw_scale = 0.25,
  frequentist = FALSE,
  unconditional = FALSE,
  draws = NULL,
  diagnostics = FALSE,
  mh_backend = c("mgcv", "gratia", "family"),
  mvn_method = c("mvnfast", "mgcv"),
  block_size = 10000L,
  dense_limit = 5e+06,
  focal = NULL,
  order = 1L,
  type = c("forward", "backward", "central"),
  scale = c("response", "linear_predictor"),
  n_grid = 200L,
  eps = NULL,
  envir = environment(formula(model))
```

```
)
```

Arguments

model	a fitted model to compute derivative samples for
...	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 derivatives of fitted values. If supplied, all but one variable must be held at a constant value.
n_draws	positive integer; total retained draws across all chains. Required when generating draws. With method = "user", the supplied matrix determines the count; an explicit count must agree with its rows.
chains	positive integer or NULL; number of chains. NULL uses one chain, or the chain mapping in supplied draws. n_draws must be divisible by chains. Burn-in is applied separately to each MH chain.
seed	numeric or NULL; seed for the complete sampling operation. An explicit seed restores the caller's random-number state on exit.
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 Hastings 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.
n_cores	positive integer; maximum concurrent MH chains using mirai, capped at chains. Existing workers are reused; otherwise temporary workers are created and cleaned up. Only used by method = "mh".
n_threads	positive integer; threads within Gaussian sampling using mvnfast. Defaults to one. This is separate from concurrent MH chains.
burnin	numeric; the length of any initial burn in period to discard. See mgcv::gam.mh().
thin	positive integer; retain every thin iterations after burn-in.
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().
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(). SCAM models do not support this correction; use unconditional = FALSE.
draws	matrix; user supplied posterior draws to be used when method = "user".
diagnostics	logical; calculate coefficient R-hat, effective sample sizes, and Monte Carlo standard errors using the optional posterior package.

mh_backend — character; "mgcv" preserves mgcv::gam.mh(). "gratia" uses compiled common-family likelihoods. "family" uses a compiled MH loop with callbacks to the fitted family object. Both support independent-observation bam fits. Smoothing parameters and parameters outside the fitted coefficient vector remain fixed. LSS scale and shape coefficients are sampled jointly with location. The family backend caches a full design matrix, smoothing penalty, and proposal factorisation once for all chains.

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.

block_size — positive integer; maximum observation rows per block in the gratia MH evaluator. Large designs are evaluated without storing a full matrix.

dense_limit — numeric; maximum number of design-matrix elements to cache for the gratia MH backend. Use zero to force blockwise evaluation.

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 must be supplied.

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".

n_grid — positive integer; resolution for constructing evaluation data if data not supplied

eps — a positive finite number giving the absolute finite-difference step, or NULL (the default) to choose it automatically. The automatic step is the fitted range of the differentiation coordinate multiplied by .Machine$double.eps^(1 / (order + p)), where p = 2 for central differences and p = 1 for forward or backward differences. A constant coordinate uses its absolute value (or one if zero) instead of its range. The step is rounded upwards and bounded below so that adding it to the coordinates produces a representable change. Stored transformed values or raw covariate summaries supply the range; supplied data are used if neither is available. This is a scale-aware heuristic, not an error bound; unusual function scales or domain boundaries may require explicit eps. For first central differences the two points are separated by eps; for second central differences they are each eps from the target.

envir — the environment within which to recreate the data used to fit model.

Details

For LSS models, response-scale derivatives differentiate the expected response returned by fitted_samples(), retaining the joint uncertainty in every parameter that affects that expectation. Predictor-scale derivatives refer to the first linear predictor. Smooth-component derivatives and derivatives of the response expectation are different quantities.

Rows with unavailable required covariates retain NA derivatives by default; explicit na.action requests follow response_derivatives(). Non-finite derivative calculations at available rows

raise an error, including when only some coefficient draws are affected. Response-scale derivatives require an existing response mean; scaled t families need fitted degrees of freedom greater than one.

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

See Also

gamm4_support

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_draws = 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)
}
```

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 = NULL,
  interval = c("confidence", "simultaneous"),
  n_sim = 10000,
  level = 0.95,
  unconditional = FALSE,
  frequentist = FALSE,
  offset = NULL,
  ncores = 1,
  partial_match = FALSE,
  ...,
  newdata = NULL,
  envir = NULL,
  wrt = c("smooth", "covariate"),
  focal = NULL,
  interval_method = c("auto", "gaussian", "bootstrap"),
  simultaneous_method = c("simulation", "tube"),
  simultaneous_domain = NULL,
  tube_control = list()
)

## S3 method for class 'gamm4'
```

```
derivatives(object, ...)

## S3 method for class 'scam'
derivatives(object, ..., unconditional = FALSE)
```

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	a positive finite number giving the absolute finite-difference step, or NULL (the default) to choose it automatically. The automatic step is the fitted range of the differentiation coordinate multiplied by .Machine$double.eps^(1 / (order + p)), where p = 2 for central differences and p = 1 for forward or backward differences. A constant coordinate uses its absolute value (or one if zero) instead of its range. The step is rounded upwards and bounded below so that adding it to the coordinates produces a representable change. Stored transformed values or raw covariate summaries supply the range; supplied data are used if neither is available. This is a scale-aware heuristic, not an error bound; unusual function scales or domain boundaries may require explicit eps. For first central differences the two points are separated by eps; for second central differences they are each eps from the target.
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
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.

envir — optional environment for functions and constants in model expressions; see smooth_estimates().

wrt — differentiation coordinate: "smooth" (the default) uses the smooth expression, such as log(x); "covariate" uses a raw covariate, such as x, and reevaluates expressions at each finite-difference point.

focal — name of the differentiation coordinate, or a vector with one name per selected smooth. For raw-covariate derivatives it must be supplied when a smooth depends on multiple raw inputs. A single name can be used for all selected univariate smooths.

interval_method — character; use stored scasm bootstrap draws for pointwise intervals with "auto" (the default), or explicitly select "gaussian" or "bootstrap". Bootstrap simultaneous bands are unavailable.

simultaneous_method — character; calibration for simultaneous bands. The default, "simulation", uses coefficient draws on the evaluation grid. "tube" uses a conservative Gaussian threshold-crossing bound over a continuous interval for supported one-dimensional cubic splines. See simultaneous_bands.

simultaneous_domain — numeric pair of increasing finite endpoints, or a named list of such pairs keyed by spline coordinate. For tube bands, NULL uses the range of returned evaluation coordinates, including gaps. All returned points must lie inside the interval and the interval must lie inside the fitted spline support. Requires tube calibration.

tube_control — named list controlling numerical tube calibration: rel_tol (default 1e-6), abs_tol (1e-8), subdivisions (1000L), and root_tol (1e-8). Requires tube calibration. Simulation controls n_sim, n_cores (or ncores), and seed are ignored by tube calibration.

Value

A tibble, currently with the following variables:

- .smooth: the smooth each row refers to,
- .by: the name of any factor by variable involved in the smooth,
- .fs: the name of any random factor variable involved in the smooth,
- .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); NA for bootstrap limits,
- .lower_ci: the lower bound of the confidence or simultaneous interval,
- .upper_ci: the upper bound of the confidence or simultaneous interval.
- plus one or more columns of data containing the values of covariates at which the derivative was evaluated.

Parallel calculation

With mirai and carrier installed, configure workers with `mirai::daemons()` to distribute calculations across selected smooths. `difference_smooths()` distributes factor-level comparisons instead. Without configured workers, or with only one calculation, evaluation is serial. These functions leave the caller's worker pool running.

Model selection, evaluation of model expressions and random draws are prepared in the caller. Workers receive the coordinates and coefficient columns needed for their calculation. Custom prediction methods are evaluated in the caller, including custom tensor margins and factor-smooth bases; workers receive their prediction matrices. Simultaneous intervals use successive draws in selection order, so `set.seed()` gives the same results with or without workers, independently of their number or completion order. `derivatives()` uses `set.seed()`; the other functions also accept `seed`, which restores the caller's random number state on exit.

`smooth_samples()` draws coefficients jointly once, then reuses those draws for every smooth, preserving dependence and chain identities. Sampling finishes before smooth calculations start, so MH chains and smooth workers are not nested. Sampling thread controls keep their existing meanings. Workers use their configured BLAS thread settings; when using a threaded BLAS, limit its threads before starting daemons to avoid competing threads.

Parallel work is most useful for expensive bases or interval calculations. Small grids can be slower because of transfer and scheduling costs. Large grids and many draws also require memory for worker inputs and returned results, in addition to the final output.

Note

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

Author(s)

Gavin L. Simpson

See Also

`difference_smooths()` to compute derivatives of differences between factor-by smooths with `order = 1` or `order = 2`.

gamm4_support

Examples

```
load_mgcv()

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

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


if (requireNamespace("mirai", quietly = TRUE) &&
    requireNamespace("carrier", quietly = TRUE)) {
```

```
  library("mirai")
  library("gratia")
  local({
    mirai::daemons(2)
    on.exit(mirai::daemons(0))
    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)
```

`dharma_residuals` *Simulation-based PIT residuals*

Description

Calculate DHARMa-style residuals using gratia's fitted response generators. No DHARMa or mgcViz installation is needed.

Usage

```
dharma_residuals(model, n = 250L, seed = NULL, conditioning = NULL, ...)
```

Arguments

`model`	a fitted model supported by diagnostic_support, with a generator for comparable scalar responses.
`n`	positive integer; number of response datasets to simulate.
`seed`	integer or `NULL`; random seed for randomised residuals. With `NULL`, calculations use the current random-number state. An explicit seed preserves the caller's state, including when none was previously initialised.
`conditioning`	character or `NULL`; hold fitted random effects fixed (`"conditional"`), average over new effects (`"marginal"`), or use the extracted GAM (`"gam_component"`), where supported. See diagnostic_support.
`...`	model-specific diagnostic options, including sdmTMB's `component` and `field_method`.

Details

For each observation, calculate the fractions of simulated responses strictly below it and below or equal to it. The residual is sampled uniformly between these fractions when simulations tie with the observation. Without ties it is the fraction strictly below the observation. This implements DHARMa's PIT idea internally, using gratia's response distributions and conditioning rules.

Residuals lie in `[0, 1]`. An observation below every simulated response has residual zero; one above every simulated response has residual one. Endpoints are retained. Increasing `n` improves the resolution of the empirical CDF. Continuous responses without ties have residuals on the grid `0:n / n`.

Responses are generated together to retain any modelled dependence. When only marginal response draws are available, these residuals describe the marginal distributions. Censored observations and multivariate responses are unsupported. Binomial successes and failures are compared as proportions with the model's fitting trial counts. A response CDF and finite response moments are not required.

Use `type = "dharma"` and `n_dharma` in diagnostic plots. QQ and worm plots use uniform reference quantiles. Independent responses have approximate pointwise order-statistic bands: the finite simulation grid means the continuous uniform reference is approximate. Bands are omitted for dependent responses because independence-based intervals do not describe their order statistics. These checks use fitted parameters without refitting the model.

An explicit `seed` preserves the caller's random-number state. With `seed = NULL`, simulation and tie randomisation advance the current state.

Value

A numeric vector in observation order. With `na.exclude`, excluded observations are restored as `NA`; with `na.omit`, only fitting rows remain.

References

Hartig, F. DHARMa: Residual Diagnostics for Hierarchical (Multi-Level / Mixed) Regression Models. doi:10.32614/CRAN.package.DHARMa.

See Also

`appraise()`, `qq_plot()`, `worm_plot()`, `quantile_residuals()`

Examples

```
fit <- mgcv::gam(count ~ spray, family = poisson(),
  data = InsectSprays, method = "REML")
head(dharma_residuals(fit, n = 250, seed = 42))
appraise(fit, type = "dharma", n_dharma = 250, seed = 42)
```

---

diagnostic_support *Model diagnostic support and conventions*

Description

Diagnostics use each model's fitted response distribution, observation order and choice of conditioning. The same calculations then produce residuals, reference values and plots across fitting packages.

Details

A conditional check holds fitted random effects fixed. A marginal check averages over their fitted distribution. When responses share random effects or residual correlation, they must be simulated together to preserve that dependence; this is called joint response simulation below. Whether responses are independent depends on the chosen conditioning, not just the model class.

Coverage:

PIT means probability integral transform: the fitted cumulative distribution function (CDF) maps each observation to a probability. At discrete values, that probability is randomised over the corresponding probability mass. Quantile residuals transform PIT values to standard-normal scores. Surrogate residuals for `mgcv::ocat()` models transform the same randomised probabilities to logistic scores. Their reference has location zero and scale one, and fitting weights do not rescale it. They hold fitted random effects fixed and use the same observation alignment and seed conventions. For `mgcv::multinom()` GAMs, surrogate residuals are vectors of centred Gumbel utility errors, one for every category including 0, following Cheng, Wang and Zhang (2021). Conditional utility draws preserve category identity and are shared across ordinary and scalable panels. Each component has its own QQ/worm reference and pointwise bands; residual panels use that category's fitted probability, and covariate panels separate categories. Select them explicitly with `type = "surrogate"`; see `surrogate_residuals()`. A posterior predictive check (PPC) compares observed responses with simulated data sets that also include approximate uncertainty in fitted parameters.

This table applies to the supported response families described below.

Fitter	PIT/quantile	Uniform QQ	Simulated QQ
lm/glm	Yes	Yes	Yes
mgcv::gam	Yes	Depends on family	Yes
mgcv::bam	Marginal families	Marginal or joint	Gaussian AR1; otherwise marginal fo
gamm/nlme::lme	Conditional/marginal	Depends on independence	Gaussian; PQL limits
gamm4/lme4::merMod	Conditional/marginal	Depends on independence	Joint, including GLMMs
glmmTMB	Conditional/marginal	Depends on independence	Joint, including non-Gaussian
GJRM::gamlss	Supported univariate families	Yes	Yes
GJRM::gjrm	Per-response families	Per response	Per response
sdmTMB	Depends on treatment of fields	Depends on independence	Joint, including delta

- `lm`/`glm`: weighted Gaussian and supported response families, with their fitted dispersion. Quasi families do not specify a response distribution, so they do not support PIT/quantile residuals or PPCs.
- `gam`/`bam`: ordinary, distributional and grouped families, including weighted Gamma, inverse Gaussian, `twlss`, ordinal and `ziP` distributions. Multivariate-normal PITs check each response separately; simulations retain covariance between responses. Multinomial PIT/quantile residuals accumulate category probabilities in mgcv's numeric coding order, `0, ..., K`. Nominal multinomial GAMs also support utility surrogate residuals in ordinary and scalable plots. Utility residuals need no ordering of categories. Traditional residual checks and PPCs remain available.
- SCAM models support ordinary and scalable posterior predictive plots for supported response families. Smoothing-parameter uncertainty is not currently supported for SCAM models; use `unconditional = FALSE`.

- gamm/gamm4 and direct lme/merMod fits use their full predictors. Gaussian nlme variance weights and residual covariance are retained. Non-Gaussian gamm fits use penalized quasi-likelihood (PQL). Without residual correlation, these support approximate response checks and PPCs based on their fitted working parameters. PQL residual correlation acts on the working response used for fitting; it does not specify a joint distribution for the observed responses.
- glmmTMB: conditional Gaussian, Poisson, binomial, NB1/NB2, beta, Gamma, Tweedie, log-normal and supported truncated distributions, including fitted zero inflation and dispersion. Marginal simulations also provide reference samples for other families that glmmTMB can simulate. Unsupported conditional family CDFs remain unavailable. The package's own joint simulator preserves temporal AR1, spatial covariance and correlated random slopes.
- Univariate GJRM::gamlss: uncensored N, LN, GA, GAi, P, NBI, NBII, BE, LO, WEI, IG, GU and rGU. These include beta, logistic, Weibull, inverse Gaussian and both Gumbel orientations. GAi is accepted by the diagnostic adapter, but some GJRM versions do not offer it in gamlss(). Probabilities use the final likelihood parameters. This corrects stored NBII probabilities that use a Poisson CDF in some upstream versions. Other families, censored and survival models are not supported by this adapter. PIT/quantile, response, Pearson and DHARMa-style residuals are available for all these families. Deviance residuals are available for N, GA, GAi and P only. Quantile QQ plots use method = NULL or "normal"; PIT and DHARMa-style QQ plots require method = NULL. Response and Pearson QQ plots support inverse-CDF ("uniform"), simulated ("simulate"), normal approximation ("normal") and pooled marginal ("marginal") references. Observed-versus-fitted panels use response means, including the Weibull and Gumbel corrections. Residuals are plotted against the first predictor, which controls Weibull scale or logistic/Gumbel location in those families. Posterior predictive checks draw both equations' coefficients together, including beta's logit-linked scale parameter.
- Joint GJRM::gjrm: ordinary uncensored bivariate models and ordinary trivariate binary models support checks of each response distribution. Supported margins are the univariate GJRM families listed above and binary responses with probit, logit or complementary log-log links. Each response uses the final fitted parameters from the joint model. A supported response can be selected even if another margin is unsupported. With a fixed seed, selected-response residuals and reference simulations match the same response in an all-response call. appraise() returns four panels, each faceted by response. Standalone diagnostic plots retain these response groups, including when both responses have the same family. Use response to select one response. These checks assess the marginal response distributions. Copula checks, joint response simulation and posterior predictive checks are unavailable. Selection, switching, ordinal, fixed-margin, censored and survival models are not supported by these diagnostic methods. Diagnostics also reject models that use a modeled response in another response's mean formula or in an auxiliary formula. The complete set of reference simulations and appraisal panels requires conditioning through the fitted copula, which these methods do not provide. Smooth plots remain available.
- sdmTMB: supported single-response and delta families retain spatial, spatiotemporal, grouping, time-varying and smooth effects. Standard deltas and deltas with the coupled Poisson link support checks of the complete response, occurrence and positive component. sdmTMB's transformations from internal parameters to response parameters must be available. Student-t checks use the fitted location, scale and fixed or estimated degrees of freedom. Pearson residuals require degrees of freedom above two; PIT/quantile residuals and predictive checks remain available with infinite variance. Predictive plots also support native response families

whose analytic residual distributions are unavailable.

Selecting a response:

Native grouped-family GAM and joint GJRM diagnostics accept `response = NULL` for all responses or a single integer for one response. For `mgcv::gfam()`, this integer indexes the family list and selects observations assigned to that component. Repeated instances of the same family have distinct indices. For GJRM, it indexes responses in model order.

Selection leaves `by_family` independent: grouped-family quantile residuals remain pooled by default, traditional residuals use facets, and explicit `TRUE` or `FALSE` chooses the layout. Appraisal observed-versus-fitted panels always use component facets. Original component labels and row identities remain available after selection.

Selected grouped-family residuals contain retained fitting observations only. The all-component call preserves the usual `na.exclude` restoration of omitted rows. With an explicit seed, selected residuals match the corresponding rows of the all-component calculation when all components support it. Selecting a supported component can also bypass an unsupported component. Grouped-family posterior predictive checks accept the same selector and retain joint coefficient draws for the fitted model.

Response encodings and residual definitions:

Factor-valued binomial responses are Bernoulli observations, even when likelihood weights exceed one. Numeric proportions use the fitted trial counts; paired success/failure counts retain their grouped-binomial encoding. Default response simulations and CDFs follow that distinction. Explicit GAM sampling weights can change trial counts or response precision, including Gaussian AR1 precision, without changing fitted coefficient uncertainty. MASS negative-binomial GLM families use their fitted size parameter. The native gaulss likelihood ignores fitting weights, so its default diagnostic references use the fitted inverse standard deviation directly. Explicit sampling precision weights remain available; see the sampling documentation for how these change the response distribution.

Native beta-regression simulation clips draws at epsilon and one minus epsilon. These endpoints therefore have probability mass. Diagnostics include that mass in their CDFs and randomisation and use moments of the same clipped distribution. This is labelled an approximation: the fitting likelihood evaluates the underlying beta density at epsilon-adjusted observations. Distribution functions supplied by callers are preserved.

GEV and sinh-arcsinh checks use stable response means and correct residual centering and signs for both observed and simulated responses. Their deviance magnitudes follow the fitting package's definitions: GEV uses the square root of its saturated log-likelihood contrast; sinh-arcsinh uses zero log density as its reference. If a sinh-arcsinh density exceeds one, that contrast is negative. Deviance checks then give an error rather than replace the residual with zero; response and PIT/quantile checks remain available. These deviance definitions do not produce comparable standardised scores across families. Use PIT/quantile residuals for uniform/standard-normal references.

GEV means and variances require shape below one and one-half, respectively. Signed deviance checks also require a finite mean and shape at least minus one. PIT/quantile checks do not require these moments. When the response mean does not exist, appraisal explains why the observed-versus-fitted panel is unavailable and retains the PIT/quantile panels.

nlme calculations retain covariance between random effects, including nested groups and block designs. They also retain the contrasts and observation order used for fitting, including after subsetting or missing-value removal. Changing global contrast options after fitting does not change the diagnostics.

Conditioning and dependence:

For response references, `conditioning = "conditional"` usually holds fitted random effects and nuisance parameters fixed. `"marginal"` simulates new effects from their fitted distributions and averages over them. Setting random effects to zero is not a substitute for this averaging, especially with nonlinear response links. sdmTMB has additional field choices described below. PPCs separately include approximate uncertainty in fitted parameters.

`"gam_component"` explicitly uses the extracted GAM component. gamm and gamm4 diagnostics and posterior predictive checks default to the full model with conditional conditioning. gamm4 response simulations use the same default; the existing gamm response simulator retains its GAM-component default. Conditional response references hold fitted effects fixed, while conditional PPCs include uncertainty in those fitted effects. Specify `"marginal"` to simulate new effects instead.

Conditional glmmTMB/lme4 response likelihoods factorize given all random effects, including temporal and spatial effects. Marginal responses may be dependent because they share those effects. An independent effect for each observation need not create dependence between observations. The calculations check covariance implied by the fitted random-effect design, including correlated slopes and covariance cancellation. Gaussian nlme residual correlation can remain after conditioning on random effects; variance weights alone do not introduce correlation.

A conditional check at estimated effects assesses the response distribution at those estimates. It does not provide a calibrated test of random-effect covariance or of the whole fitted model. Marginal joint references assess response variation implied by new simulated effects. A QQ plot checks the distribution of residuals; inspect their ordering or spatial pattern as well to assess dependence. Models are not refitted when calculating reference bands.

BAM working correlation:

Gaussian identity-link AR1 BAM retains joint response simulation, with new independent series starting at `AR.start` boundaries. For non-Gaussian or nonidentity models fitted with `discrete = TRUE` and nonzero `rho`, BAM applies AR1 to working residuals as a generalized estimating equations (GEE) approximation. This does not specify a unique joint response distribution. The restriction applies to supported `extended.family` and GLM families. See https://stat.ethz.ch/R-manual/R-devel/library/mgcv/html/bam.html.

Fitted marginal CDFs, inverse CDFs and PIT/quantile residuals remain useful. For traditional residuals, the default QQ plot and a requested inverse-CDF QQ plot use `method = "marginal"`. This approximates the pooled distribution of individual residuals by numerical sampling, without asserting independence or supplying dependence bands. Explicit `method = "simulate"`, joint response simulation and PPCs give an explanatory error. No additional dependence model, such as a copula, is introduced. An explicitly requested `"normal"` approximation is available without independence-based bands; it is never the default here.

`residual_correlation()` and `residual_correlation_plot()` provide approximate checks of correlation between residuals at different lags. For non-Gaussian BAM, the stored `std.rsd` residuals differ from the weighted working residuals used in fitting. Adjusting working residuals for the fitted AR1 correlation (whitening) or using `std.rsd` can reveal remaining serial structure. Neither supplies a generative response model or calibrated p-values. Series boundaries and gaps from missing observations are retained, and no significance bands based on independence are drawn. Lags count observation positions, not spatial distance or irregular elapsed time. Specify series and order the fitting data to suit the dependence being checked.

sdmTMB field choices:

With fixed-effect coefficients and covariance parameters held at their maximum likelihood estimates (MLEs):

- `field_method = "mle-mvn"` (default) follows sdmTMB's recommended approach: draw the fitted fields together from their approximate posterior distribution, then hold that draw fixed for conditional PITs and response references. Response errors are independent given those fields. Reference bands use the same draw as the observed residuals.
- `"mle-eb"` holds empirical-Bayes field estimates fixed. This matches sdmTMB's residual construction at fitted fields, which its documentation does not recommend for goodness-of-fit testing. Shrinkage and estimation can distort the resulting residual distribution.
- `"mle-mvn-integrated"` averages over approximate uncertainty in fitted fields using sdmTMB's joint response simulations. Residuals are calculated from ranks in these simulations rather than from one fixed draw of fields. This reuses the fitting data and can give conservative checks; treat the results as approximate posterior predictive checks.
- `conditioning = "marginal"` instead simulates new spatial, spatiotemporal and grouping effects. Simulation at fitting observations also renews other supported field and time-varying effects; smooths remain fitted. `field_method` does not cause another field draw in this mode.

`gratia.diagnostics.field_seed` (default 1) makes the posterior field draw reproducible independently of PIT randomisation. All panels use the same draw. To inspect further draws, change this option and present the results together. The sdmTMB definitions and recommendations are documented at https://sdmtmb.github.io/sdmTMB/reference/residuals.sdmTMB.html and https://sdmtmb.github.io/sdmTMB/articles/residual-checking.html.

PPCs use a separate approximation that draws fixed-effect, nuisance and field parameters together. Marginal PPCs simulate new fields at each parameter draw; smooths retain their posterior draws. Plot captions identify conditioning. Parameter estimation, the Gaussian approximation to the posterior and finite simulation error affect these checks. None of these modes guarantees that goodness-of-fit tests attain their nominal false-positive rate after fitting.

Calculating reference values and bands:

Analytic PIT/quantile QQ plots use uniform/standard-normal reference quantiles. For ideal independent responses, bands are pointwise intervals for ordered residuals. For dependent responses, bands come from joint simulations (`n_simulate`) when a joint generator exists. Otherwise the bands are omitted and the plot explains why. Bands condition on the chosen fitted distribution and fields. They do not provide simultaneous coverage or account for refitting.

When a joint simulator is available but marginal CDFs or moments cannot be calculated analytically, one reference sample of simulated data sets supplies means, variances and residual ranks. With `R` reference draws, PIT is `(number below + U * (ties + 1))/(R + 1)`, where `U` is uniform on `(0, 1)`. The extra rank position is for the observation being checked, which keeps normal scores finite even outside the simulated range. Observed and simulated responses use the same reference sample, centering and residual definitions. These ranks need not equal the empirical CDF at an observation.

`options(gratia.diagnostics.marginal_draws = 249L)` sets the reference sample size (minimum 19). Increase it to assess sensitivity to Monte Carlo error. `gratia.diagnostics.marginal_seed` (default 1) makes this sample reproducible independently of residual randomisation. A finite sample limits tail resolution and moment accuracy. Moment estimates require finite population moments: glmmTMB Student-t response residuals require df > 1, and Pearson residuals require df > 2. An undefined moment is not replaced by a finite sample estimate. PIT/quantile checks and joint draws remain available, but an undefined mean cannot be shown as a fitted expectation. Bands

for residuals calculated from simulation ranks use joint draws and the same rank transformation, even if the underlying responses are independent. Marginal deviance residuals remain unavailable where the integrated likelihood has no defined deviance; use response, Pearson or quantile residuals instead.

DHARMa-style residuals:

`dharma_residuals()` and `type = "dharma"` implement simulation-based PIT residuals using gratia's existing generators, without requiring DHARMa or mgcViz. Responses are ranked against `n_dharma` simulated datasets (default 250), with randomisation within ties and exact zero and one endpoints. This calculation differs from the existing simulation-rank formula above: it uses the empirical fractions below and below or equal to the observation, without an extra rank position. Existing PIT and quantile residuals retain their current definitions.

The calculation needs comparable scalar response draws, but no CDF or finite response moments. Existing model adapters retain their conditioning defaults. Marginal-only generators provide marginal distribution checks. Censored and multivariate observations are unsupported. Independent responses use uniform QQ references with approximate pointwise bands because the simulation count is finite; dependent responses omit independence-based bands.

Remaining limitations and extensions:

Inverse CDFs with direct lower/upper and log-probability arguments preserve extreme interior quantiles. Older functions that accept only lower-tail probabilities cannot always represent an extreme upper-tail probability. If conversion rounds it to an endpoint and produces a false infinite quantile, an explanatory error is raised. This limits extreme upper quantiles of the compound-Poisson Tweedie distribution. The Gamma boundary and inverse-Gaussian solver accept log probabilities directly. True probability endpoints remain valid.

BAM/PQL working correlations need a scientifically specified joint response model before joint simulation or PPC can be added. Additional GJRM and other fitting-package families should supply their parameters to the shared response distributions. Censored/survival PPCs need a way to reapply censoring to each simulated response. Nominal multivariate PITs need a documented category ordering or a different multivariate diagnostic. Unrecognised random-effect structures remain unsupported. Diagnostics do not repair `na.pass` fits where missing responses cannot be aligned with predictions and residuals; ordinary `na.omit`/`na.exclude` fits are covered. MCMC, model refitting and calibrated dependence tests would require additional sampling or refitting methods.

Large data sets and family functions:

Like mgcViz, these diagnostics use residuals with known reference distributions where available and aggregate large data sets before plotting. All observations and QQ endpoints contribute. Kernel density estimates and histograms remain available as separate views. See https://mfasiolo.github.io/mgcViz/reference/qq.gamViz.html. mgcViz approximates QQ plots on a discrete grid; gratia aggregates plot points but preserves exact pooled reference quantiles. mgcViz also plots differences between empirical and model-implied conditional densities. Adding such plots would require comparable smoothing bandwidths and a validated conditional reference. The existing density and histogram views retain information about scale misspecification rather than rescaling every residual to unit variance. See https://mfasiolo.github.io/mgcViz/reference/l_densCheck.html.

Uniform references average ordered residuals as draws are generated, without storing all draws. Simulated QQ uses exact pooled quantiles computed as by `stats::quantile(type = 7)` and

pointwise intervals for each residual rank. Above `options(gratia.diagnostics.max_cells = 1e7)`, sorted draws are written to temporary files and merged exactly. This limits simulation memory use at the cost of disk I/O. One response vector and the reference output must still fit in memory. Scalable PPCs summarise every draw on the same grid; discrete responses with more than 512 distinct values use labelled probabilities for 512 bins. `posterior_samples()` returns the raw response draws. Coefficient and random-effect draws still need memory proportional to the number of parameters times the number of draws.

The internal `fix_family_rd()`, `fix_family_cdf()` and `fix_family_qf()` functions still add `$rd`, `$cdf` and `$qf` to mgcv-compatible families while preserving functions supplied by callers. The fitted-distribution code adapts their arguments locally. Families can supply a `gratia_diagnostics` function for response moments, probability bounds and other distribution information; see family_diagnostics for the interface. Other fitting packages can use the same standard response distributions, or combine them by family index, truncate them or add point masses without changing the plotting calculations.

See Also

`appraise()`, `quantile_residuals()`, `qq_plot()`, `scalable_qq_plot()`, `posterior_predictive_plot()`, gamm4_support

`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. First and second derivatives of these differences, including partial derivatives across smooth surfaces, can also be estimated.

Usage

```
difference_smooths(model, ...)

## S3 method for class 'gam'
difference_smooths(
  model,
  select = NULL,
  smooth = deprecated(),
  n = 100,
  n_2d = 50,
  ci_level = 0.95,
  data = NULL,
  group_means = FALSE,
  partial_match = TRUE,
  unconditional = FALSE,
  frequentist = FALSE,
```

```
  envir = NULL,
  ...,
  interval = c("confidence", "simultaneous"),
  n_sim = 10000,
  n_cores = 1,
  seed = NULL,
  variables = NULL,
  n_3d = 16,
  n_4d = 4,
  simultaneous_method = c("simulation", "tube"),
  simultaneous_domain = NULL,
  tube_control = list(),
  order = 0L,
  focal = NULL,
  type = c("central", "forward", "backward"),
  eps = NULL,
  wrt = c("smooth", "covariate")
)

## S3 method for class 'scam'
difference_smooths(model, ..., unconditional = FALSE)
```

Arguments

model	A fitted model.
...	arguments passed to other methods. Not currently used.
select	character, logical, or numeric; which smooths to compare. If NULL, the default, then all model smooths are factor-smooth interactions are compared. 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. Careful selection is needed because it is not allowed to compare smooths of different covariates or of different factor-by variables. For character select, specific named smooths cane be provided, in which case, the exact names of the smooths (as given by smooths(), for example, can be specified, and partial_match must be set to FALSE.
smooth	**[Deprecated]** Use select instead.
n	numeric; the number of points at which to evaluate the difference between pairs of smooths.
n_2d	numeric; the number of points along each of the first two axes of a smooth surface, including surface panels of higher-dimensional smooths. The default is 50 in plotting and plot-preparation functions. If NULL, use n instead. Ignored when evaluation data are supplied. Factor levels are retained, and curves with only one continuous covariate use n.
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? Must be `FALSE` when `order > 0`.

`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?

`envir` an optional environment supplying functions and constants used in model expressions. The available model formula environment is used when `NULL`. Covariate observations should be supplied in `data`.

`interval` character; `"confidence"` (the default) gives pointwise intervals. `"simultaneous"` gives simultaneous intervals for differences of smooths at the supplied covariate combinations, separately for each pair of factor levels.

`n_sim` positive integer; number of coefficient draws used for simultaneous intervals. Ignored for pointwise intervals.

`n_cores` positive integer; number of cores used by `mvnfast::rmvn()` for simultaneous intervals. Parallel execution requires OpenMP support.

`seed` integer or `NULL`; optional random seed for simultaneous intervals. An explicit seed preserves the caller's random number state. With `NULL`, the current random number state is used and advanced.

`variables` Character vector giving every covariate of the selected smooths exactly once, in plotting order: x, y, then the conditioning covariate for 3D smooths, or facet rows and columns for 4D smooths. When supplied, selected smooths must use the same covariates. Use fitted coordinate names, such as `"log(x)"` for a transformed covariate. The default, `NULL`, favours a two-variable tensor margin for x and y, otherwise retaining fitted order. Grid resolution follows this order. Supplied `data` are evaluated unchanged. Drawing saved estimates can change the order but uses only points already evaluated; evaluate again to change grid resolution or conditioning values. Continuous covariates must precede factors. Spherical smooths retain their latitude/longitude convention and do not support reordering.

`n_3d, n_4d` numeric; the number of conditioning values for smooths of three or four covariates. The defaults are 16 and 4, respectively. If `NULL`, use `n`. Ignored when evaluation `data` are supplied.

`simultaneous_method`
character; calibration for simultaneous bands. The default, `"simulation"`, uses coefficient draws on the evaluation grid. `"tube"` uses a conservative Gaussian threshold-crossing bound over a continuous interval for supported one-dimensional cubic splines. See simultaneous_bands.

`simultaneous_domain`
numeric pair of increasing finite endpoints, or a named list of such pairs keyed by spline coordinate. For tube bands, `NULL` uses the range of returned evaluation coordinates, including gaps. All returned points must lie inside the interval and the interval must lie inside the fitted spline support. Requires tube calibration.

`tube_control` named list controlling numerical tube calibration: `rel_tol` (default `1e-6`), `abs_tol` (`1e-8`), `subdivisions` (`1000L`), and `root_tol` (`1e-8`). Requires tube calibration. Simulation controls `n_sim`, `n_cores` (or `ncores`), and `seed` are ignored by tube calibration.

`order` numeric; `0` (the default) estimates smooth differences; `1` and `2` estimate their first and second derivatives.

`focal` character; one differentiation coordinate shared by all compared smooths. Inferred for univariate smooths; required for multivariate smooths when `order > 0`.

`type` character; finite-difference scheme for derivatives. One of `"central"` (the default), `"forward"`, or `"backward"`.

`eps` a positive finite number giving the absolute finite-difference step, or `NULL` (the default) to choose it automatically. The automatic step is the fitted range of the differentiation coordinate multiplied by `.Machine$double.eps^(1 / (order + p))`, where `p = 2` for central differences and `p = 1` for forward or backward differences. A constant coordinate uses its absolute value (or one if zero) instead of its range. The step is rounded upwards and bounded below so that adding it to the coordinates produces a representable change. Stored transformed values or raw covariate summaries supply the range; supplied `data` are used if neither is available. This is a scale-aware heuristic, not an error bound; unusual function scales or domain boundaries may require explicit `eps`. For first central differences the two points are separated by `eps`; for second central differences they are each `eps` from the target.

`wrt` differentiation coordinate: `"smooth"` (the default) uses the smooth expression, such as `log(x)`; `"covariate"` uses a raw covariate, such as `x`, and reevaluates expressions at each finite-difference point.

Details

With the default simulation calibration, simultaneous intervals cover smooth differences at the evaluated covariate combinations for each pair of factor levels, with approximate posterior probability `ci_level` when using the Bayesian covariance. They do not provide joint coverage across all pairs, outside the evaluation set, or between its points. With `data = NULL`, the evaluation set is generated using `n`, `n_2d`, `n_3d`, and `n_4d`. Differences and interval limits are on the linear predictor scale, including group means when requested. For supported univariate cubic splines, `simultaneous_method = "tube"` instead covers a continuous interval, including between returned points; see simultaneous_bands. Coverage remains separate for each pair.

`variables` determines the order used to generate and display the evaluation grid. The first two covariates are the `x` and `y` axes. The third covariate defines facets for three-dimensional smooths, or facet rows for four-dimensional smooths; the fourth defines facet columns. Use `data` to choose particular conditioning values. The resolved order is saved with the estimates for use by `draw()`.

The intervals use the joint coefficient covariance selected by `unconditional` and `frequentist`, retaining covariance between the two smooths. The covariance must be positive semidefinite. With `frequentist = TRUE`, simulation instead uses the frequentist covariance of the coefficient estimators. `n_sim`, `n_cores`, and `seed` are ignored for pointwise intervals. An explicit seed scopes the entire call: each pair uses a new batch of coefficient draws, and the caller's random number state is restored on exit. With `seed = NULL`, the current random number state is used and advanced.

With `order > 0`, the estimate is the difference between the derivatives of the selected smooths, on the linear predictor scale. Equivalently, it is the derivative of their difference. The estimate and its uncertainty are computed from the difference of the derivative matrices, retaining covariance between the smooths. Group means are excluded, and tube intervals are unavailable for derivatives; use simulation for simultaneous intervals.

For a multivariate smooth, `focal` identifies the coordinate to differentiate. All other independent coordinates are fixed at their value in each evaluation row. They can vary between rows, so `data` may describe a full surface or several slices. `order = 2` differentiates twice with respect to `focal`; mixed partial derivatives are not supported. Generated grids adjust the focal range for the finite-difference scheme; supplied evaluation locations are preserved.

With `wrt = "covariate"`, model expressions are reevaluated after changing the raw focal covariate. Generated raw grids use `smooth_data()` and require each axis to depend on one distinct original covariate. Supply raw `data` for other transformations. With `wrt = "smooth"`, only the named smooth coordinate is changed. Derivative controls are ignored when `order = 0`. Raw-coordinate comparisons match the same original covariates, excluding the comparison factor. Evaluated smooth coordinates may differ between factor levels when a transformation involves that factor. The finite-difference step must fit within the fitted focal range for generated grids, including grids with only one focal value.

Value

A tibble of class `"difference_smooth"`. Columns `.smooth`, `.by`, `.level_1`, and `.level_2` identify each comparison; `.diff` contains the estimate for level 1 minus level 2, with `.se`, `.lower_ci`, `.upper_ci`, and evaluation covariates. For `order > 0`, `.diff` contains the derivative difference and `.focal` names its coordinate. The `derivative_info` attribute records `order`, `focal`, `wrt`, `type`, and the resolved step `eps`. The `plot_variables` attribute records the plotting axes.

Parallel calculation

With mirai and carrier installed, configure workers with `mirai::daemons()` to distribute calculations across selected smooths. `difference_smooths()` distributes factor-level comparisons instead. Without configured workers, or with only one calculation, evaluation is serial. These functions leave the caller's worker pool running.

Model selection, evaluation of model expressions and random draws are prepared in the caller. Workers receive the coordinates and coefficient columns needed for their calculation. Custom prediction methods are evaluated in the caller, including custom tensor margins and factor-smooth bases; workers receive their prediction matrices. Simultaneous intervals use successive draws in selection order, so `set.seed()` gives the same results with or without workers, independently of their number or completion order. `derivatives()` uses `set.seed()`; the other functions also accept `seed`, which restores the caller's random number state on exit.

`smooth_samples()` draws coefficients jointly once, then reuses those draws for every smooth, preserving dependence and chain identities. Sampling finishes before smooth calculations start, so MH chains and smooth workers are not nested. Sampling thread controls keep their existing meanings. Workers use their configured BLAS thread settings; when using a threaded BLAS, limit its threads before starting daemons to avoid competing threads.

Parallel work is most useful for expensive bases or interval calculations. Small grids can be slower because of transfer and scheduling costs. Large grids and many draws also require memory for worker inputs and returned results, in addition to the final output.

See Also

derivatives(), partial_derivatives()

Examples

```
load_mgcv()
# Simulate group-specific curves with Gaussian noise and fit factor-by smooths
df <- data_sim("eg4", n = 300, seed = 42)
m <- gam(y ~ fac + s(x2, by = fac) + s(x0), data = df, method = "REML")

# Compare the centred smooth shapes for each pair of factor levels
sm_dif <- difference_smooths(m, select = "s(x2)")
sm_dif
draw(sm_dif)

# Compare rates of change instead of smooth values, using the same fit
slope_dif <- difference_smooths(m, select = "s(x2)", order = 1)
slope_dif
```

---

dispersion *Dispersion parameter for fitted model*

Description

[Experimental]

Usage

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

## S3 method for class 'scam'
dispersion(model, ...)

dispersion(model, ...)

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

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

Arguments

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

See Also

gamm4_support

`draw` *Generic plotting via* `ggplot2`

Description

Generic plotting via `ggplot2`

Usage

```
draw(object, ..., label_control = NULL)
```

Arguments

`object`	and R object to plot.
`...`	arguments passed to other methods.
`label_control`	Optional layout settings created by `draw_label_control()`. These apply across drawing methods without changing their default appearance.

Details

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

Value

A `ggplot2::ggplot()` object.

Label layout

Pass `label_control = draw_label_control(...)` to `draw()` to wrap discrete axis labels and legend text, stagger discrete tick labels, or adjust legend and colour-bar layout. Settings apply to each panel, including unwrapped plot lists. See `draw_label_control()` for the available settings.

Author(s)

Gavin L. Simpson

`draw.ale` *Plot accumulated local effects*

Description

Plot saved ALE estimates and intervals, without recomputing predictions.

Usage

```
## S3 method for class 'ale'
draw(
  object,
  ci = TRUE,
  ref_line = TRUE,
  alpha = 0.2,
  xlab = NULL,
  ylab = NULL,
  ...
)
```

Arguments

`object`	a tibble of class `"ale"`, produced by `ale()`.
`ci`	logical; show the saved interval ribbon?
`ref_line`	logical; draw a horizontal reference line at zero?
`alpha`	Deprecated; use `gratia_theme_get()` theme settings instead. Numeric; opacity of the interval ribbon.
`xlab, ylab`	optional axis labels. Defaults identify the focal covariate and selected parameter. The computation scale appears in the caption. Titles and subtitles are omitted. Use `ggplot2::labs()` to change or remove the caption, for example `+ ggplot2::labs(caption = NULL)`.
`...`	reserved for future use.

Value

A `ggplot2::ggplot()` object.

Label layout

Pass `label_control = draw_label_control(...)` to `draw()` to wrap discrete axis labels and legend text, stagger discrete tick labels, or adjust legend and colour-bar layout. Settings apply to each panel, including unwrapped plot lists. See `draw_label_control()` for the available settings.

See Also

`ale()`

Examples

```
load_mgcv()
df <- data_sim("eg1", n = 100, seed = 2)
m <- gam(y ~ s(x2), data = df, method = "REML")
ale(m, "x2") |> draw()
```

`draw.basis` *Plot basis functions*

Description

Plots basis functions using ggplot2

Usage

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

Arguments

`object`	an object, the result of a call to `basis()`.
`legend`	logical; should a legend by 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	Deprecated; use gratia_theme_get() theme settings instead. Colour specification for contour lines.
...	arguments passed to other methods. Not used by this method.
variables	Character vector giving every covariate of the selected smooths exactly once, in plotting order: x, y, then the conditioning covariate for 3D smooths, or facet rows and columns for 4D smooths. When supplied, selected smooths must use the same covariates. Use fitted coordinate names, such as "log(x)" for a transformed covariate. The default, NULL, favours a two-variable tensor margin for x and y, otherwise retaining fitted order. Grid resolution follows this order. Supplied data are evaluated unchanged. Drawing saved estimates can change the order but uses only points already evaluated; evaluate again to change grid resolution or conditioning values. Continuous covariates must precede factors. Spherical smooths retain their latitude/longitude convention and do not support reordering.

Details

For smooths of three or four covariates, each basis function has its own plot. The first two covariates define the surface axes. Three-covariate smooths have wrapped facets for the third covariate; four-covariate smooths have the third covariate in rows and the fourth in columns. ncol and nrow control the outer arrangement of plots. Fill scales share a range within each smooth. Use variables in basis() to choose the axes before grid generation, or here to rearrange existing evaluations without resampling. Surface plots include contour lines by default, matching draw.gam(). Use contour = FALSE to draw surfaces without contour lines.

Factor-by bases from smooth specifications are drawn once for the whole factor-by set, using evaluations from all available levels. For fitted models, basis(model) returns coefficient-weighted contributions, which remain separate for each level. Use basis(model, unweighted = TRUE) to draw the shared basis once. Subsetting the basis before drawing restricts the plot to the retained evaluations and basis functions.

Value

A patchwork object.

Label layout

Pass label_control = draw_label_control(...) to draw() to wrap discrete axis labels and legend text, stagger discrete tick labels, or adjust legend and colour-bar layout. Settings apply to each panel, including unwrapped plot lists. See draw_label_control() for the available settings.

Author(s)

Gavin L. Simpson

Examples

```
load_mgcv()
# Simulate an additive Gaussian response and fit a GAM
df <- data_sim("eg1", n = 200, seed = 42)
m <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = df, method = "REML")

# Plot the coefficient-weighted basis functions for one smooth
bf <- basis(m, select = "s(x2)")
draw(bf)
```

---

draw.compare_smooths *Plot comparisons of smooths*

Description

Plot comparisons of smooths

Usage

```
## S3 method for class 'compare_smooths'
draw(
  object,
  ncol = NULL,
  nrow = NULL,
  guides = "collect",
  select = NULL,
  partial_match = FALSE,
  show = c("models", "difference"),
  pair = NULL,
  ci_level = 0.95,
  surface = c("raster", "tile"),
  contour = TRUE,
  n_contour = NULL,
  continuous_fill = NULL,
  ...
)
```

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; guide handling for the combined plot, passed to patchwork::wrap_plots(). The default is "collect".

`select`, `partial_match`
: select smooths from the comparison, using the conventions of `compare_smooths()`. Numeric and logical selections follow the order of unique smooth labels in `object`.

`show`
: character; `"models"` (default) overlays one-variable smooths and draws two-variable smooths in side-by-side model facets with fixed axes and a shared fill scale. `"difference"` draws the estimated difference for selected two-variable smooths, using a fill scale centred on zero.

`pair`
: two model names or integer indices in input order. In difference mode, subtract the second model from the first. With exactly two models, the default uses their input order; with more models, supply `pair`.

`ci_level`
: numeric; one finite value strictly between 0 and 1, giving the coverage of the pointwise normal intervals for one-variable smooths. The default is `0.95`. Use `ci_level` to change the interval level; `coverage` and `level` are not accepted, even when `ci_level` is also supplied.

`surface`
: character; `"raster"` or `"tile"` for surface displays. Irregular numeric grids use rectangles bounded by coordinate midpoints.

`contour`
: logical; add contour lines? These require two numeric axes and at least one grid cell with four finite corner values that are not all equal.

`n_contour`
: numeric; the number of contour bins. Will result in `n_contour - 1` contour lines being drawn. See `ggplot2::geom_contour()`.

`continuous_fill`
: a suitable fill scale to be used when plotting continuous variables.

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

Details

Difference plots match both coordinates and any retained conditioning covariates. Locations missing from either model, and missing estimates, remain missing. Multiple conditioning values are drawn in separate panels. No interval is calculated for the difference: estimates from models fitted to the same observations can be correlated, and their separate covariance matrices do not supply the covariance between models. See `compare_smooths()` for the scale and centring of the compared smooth contributions.

Label layout

Pass `label_control = draw_label_control(...)` to `draw()` to wrap discrete axis labels and legend text, stagger discrete tick labels, or adjust legend and colour-bar layout. Settings apply to each panel, including unwrapped plot lists. See `draw_label_control()` for the available settings.

`draw.conditional_differences` *Plot conditional differences*

Description

Display conditional differences as curves with pointwise or simultaneous intervals and a zero reference line, or as surfaces over two grouped numeric covariates. Each comparison is shown in a separate facet. Surfaces map the difference to fill and, in 3D, height, with a fill range symmetric around zero. Additional conditions supply facets for surfaces, or colour and facets for curves, as in `draw.conditional_values()`.

Usage

```
## S3 method for class 'conditional_differences'
draw(
  object,
  facet_scales = "fixed",
  discrete_colour = NULL,
  discrete_fill = NULL,
  xlab = NULL,
  ylab = NULL,
  ...,
  surface = c("raster", "tile", "3d"),
  surface_3d = list(),
  continuous_fill = NULL,
  contour = TRUE,
  n_contour = NULL,
  ci_3d = FALSE,
  zlab = NULL
)
```

Arguments

`object`	an object returned by `conditional_differences()`.
`facet_scales`	character; should facets have the same axis scales across facets? See `ggplot2::facet_wrap()` for details. Options are: `"fixed"` (default), `"free_x"`, `"free_y"`, and `"free"`.
`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.
`xlab`	character; label for the x axis of the plot.
`ylab`	character; label for the prediction axis of a curve, or the second covariate axis of a surface.
`...`	arguments passed to `draw.conditional_values()`.

surface	character; "raster" (default), "tile", or "3d". Applies to grouped numeric conditions. Three-dimensional surfaces require the optional ggcube package. Irregular flat grids use rectangles bounded by adjacent coordinate midpoints.
surface_3d	named list of view options passed to ggcube::coord_3d(), for example list(yaw = 45, persp = FALSE). The default view uses ggcube::light(backface_scale = 1). A supplied light replaces that lighting specification. The same view applies to all panels; gratia retains its plot theme.
continuous_fill	a suitable fill scale to be used when plotting continuous variables.
contour	logical; draw contour lines on surfaces? Defaults to TRUE.
n_contour	numeric; the number of contour bins. Will result in n_contour - 1 contour lines being drawn. See ggplot2::geom_contour().
ci_3d	logical; add the returned lower and upper interval surfaces when surface = "3d"? Defaults to FALSE.
zlab	character; label for the surface fill legend and 3D height axis. Defaults to the prediction label, or the difference and its scale for conditional differences.

Label layout

Pass label_control = draw_label_control(...) to draw() to wrap discrete axis labels and legend text, stagger discrete tick labels, or adjust legend and colour-bar layout. Settings apply to each panel, including unwrapped plot lists. See draw_label_control() for the available settings.

Examples

```
load_mgcv()
df <- data_sim("eg4", seed = 2)
m <- gam(y ~ fac + s(x2, by = fac), data = df, method = "REML")
draw(conditional_differences(m, by = "fac", condition = "x2"))
```

---

draw.conditional_moments

Draw conditional response moments

Description

Draw conditional response moments

Usage

```
## S3 method for class 'conditional_moments'
draw(
  object,
  facet_scales = "fixed",
  discrete_colour = NULL,
  discrete_fill = NULL,
```

```
  xlab = NULL,
  ylab = NULL,
  ...
)
```

Arguments

object	A tibble returned by conditional_moments().
facet_scales	character; should facets have the same axis scales across facets? See ggplot2::facet_wrap() for details. Options are: "fixed" (default), "free_x", "free_y", and "free".
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.
xlab	character; label for the x axis of the plot.
ylab	Character; label for the response moment axis.
...	Additional arguments passed to patchwork::wrap_plots().

Value

A patchwork containing one plot per moment and response/category.

Label layout

Pass label_control = draw_label_control(...) to draw() to wrap discrete axis labels and legend text, stagger discrete tick labels, or adjust legend and colour-bar layout. Settings apply to each panel, including unwrapped plot lists. See draw_label_control() for the available settings.

---

draw.conditional_values

Plot conditional predictions

Description

Curves show predictions and intervals. Grouped numeric conditions show a surface, with predictions mapped to fill and, for surface = "3d", height. Remaining conditions create separate facets. Flat surfaces show estimates; ci_3d = TRUE adds the returned lower and upper interval surfaces in 3D.

Usage

```
## S3 method for class 'conditional_values'
draw(
  object,
  facet_scales = "fixed",
  discrete_colour = NULL,
  discrete_fill = NULL,
```

```
  xlab = NULL,
  ylab = NULL,
  ...,
  surface = c("raster", "tile", "3d"),
  surface_3d = list(),
  continuous_fill = NULL,
  contour = TRUE,
  n_contour = NULL,
  ci_3d = FALSE,
  zlab = NULL
)
```

Arguments

object	an object of class "conditional_values", the result of a call to conditional_values().
facet_scales	character; should facets have the same axis scales across facets? See ggplot2::facet_wrap() for details. Options are: "fixed" (default), "free_x", "free_y", and "free".
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.
xlab	character; label for the x axis of the plot.
ylab	character; label for the prediction axis of a curve, or the second covariate axis of a surface.
...	additional arguments; currently unused.
surface	character; "raster" (default), "tile", or "3d". Applies to grouped numeric conditions. Three-dimensional surfaces require the optional ggcube package. Irregular flat grids use rectangles bounded by adjacent coordinate midpoints.
surface_3d	named list of view options passed to ggcube::coord_3d(), for example list(yaw = 45, persp = FALSE). The default view uses ggcube::light(backface_scale = 1). A supplied light replaces that lighting specification. The same view applies to all panels; gratia retains its plot theme.
continuous_fill	a suitable fill scale to be used when plotting continuous variables.
contour	logical; draw contour lines on surfaces? Defaults to TRUE.
n_contour	numeric; the number of contour bins. Will result in n_contour - 1 contour lines being drawn. See ggplot2::geom_contour().
ci_3d	logical; add the returned lower and upper interval surfaces when surface = "3d"? Defaults to FALSE.
zlab	character; label for the surface fill legend and 3D height axis. Defaults to the prediction label, or the difference and its scale for conditional differences.

Label layout

Pass label_control = draw_label_control(...) to draw() to wrap discrete axis labels and legend text, stagger discrete tick labels, or adjust legend and colour-bar layout. Settings apply to each panel, including unwrapped plot lists. See draw_label_control() for the available settings.

`draw.derivatives` *Plot derivatives of smooths*

Description

Plot derivatives of smooths

Usage

```
## S3 method for class 'derivatives'
draw(
  object,
  select = NULL,
  scales = c("free", "fixed"),
  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,
  differentiate_factor_smooths = TRUE,
  ...
)

## 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` Deprecated; use `gratia_theme_get()` theme settings instead. Numeric; alpha transparency for confidence or simultaneous interval.

`change_col`, `decrease_col`, `increase_col`

Deprecated; use `gratia_theme_get()` theme settings instead. 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` Deprecated; use `gratia_theme_get()` theme settings instead. 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()`.

`differentiate_factor_smooths`

logical; should colour be used to differentiate the levels of the factors involved in random factor smooths (`bs = "fs"`) or constrained factor smooths (`bs = "sz"`)? If `TRUE`, the factor(s) are used to define groups and mapped to the colour aesthetic.

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

Label layout

Pass `label_control = draw_label_control(...)` to `draw()` to wrap discrete axis labels and legend text, stagger discrete tick labels, or adjust legend and colour-bar layout. Settings apply to each panel, including unwrapped plot lists. See `draw_label_control()` for the available settings.

Examples

```
load_mgcv()
# Simulate an additive Gaussian response and fit a GAM
dat <- data_sim("eg1", n = 200, scale = 1, seed = 42)
mod <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = dat, method = "REML")

# Estimate and plot the rate of change of one smooth
```

```
df <- derivatives(mod, select = "s(x2)", type = "central")
draw(df)
```

`draw.difference_smooth` *Plot differences of smooths*

Description

Plot differences of smooths

Usage

```
## S3 method for class 'difference_smooth'
draw(
  object,
  select = NULL,
  rug = FALSE,
  ref_line = FALSE,
  contour = FALSE,
  contour_col = "black",
  n_contour = NULL,
  ci_alpha = 0.2,
  ci_col = "black",
  smooth_col = "black",
  line_col = "red",
  scales = c("free", "fixed"),
  ncol = NULL,
  nrow = NULL,
  guides = "keep",
  xlab = NULL,
  ylab = NULL,
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  angle = NULL,
  ...,
  variables = 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 — Deprecated; use gratia_theme_get() theme settings instead. 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 — Deprecated; use gratia_theme_get() theme settings instead. Numeric; alpha transparency for confidence or simultaneous interval.

ci_col — Deprecated; use gratia_theme_get() theme settings instead. Colour specification for the confidence/credible intervals band. Affects the fill of the interval.

smooth_col — Deprecated; use gratia_theme_get() theme settings instead. Colour specification for the the smooth or difference line.

line_col — Deprecated; use gratia_theme_get() theme settings instead. Colour specification for drawing reference lines

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()

xlab, ylab, title, subtitle, caption
character; labels with which to annotate plots

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

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

variables — character vector; every covariate of the selected smooth, in plotting order. The first two are the x and y axes; the third defines facets for three covariates, or facet rows for four covariates; the fourth defines facet columns. NULL uses the order saved with the estimates. Drawing rearranges evaluated points without generating a new grid.

Label layout

Pass label_control = draw_label_control(...) to draw() to wrap discrete axis labels and legend text, stagger discrete tick labels, or adjust legend and colour-bar layout. Settings apply to each panel, including unwrapped plot lists. See draw_label_control() for the available settings.

Examples

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

# calculate the differences between pairs of smooths the f_j(x2) term
diffs <- difference_smooths(m, 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.

Label layout

Pass label_control = draw_label_control(...) to draw() to wrap discrete axis labels and legend text, stagger discrete tick labels, or adjust legend and colour-bar layout. Settings apply to each panel, including unwrapped plot lists. See draw_label_control() for the available settings.

Author(s)

Gavin L. Simpson

---

draw.gam *Plot smooth and parametric terms from a fitted GAM*

Description

Plots smooth and parametric contributions from a fitted GAM model using ggplot2::ggplot(). For models fitted by mgcv::gamm(), draw() and assemble() use the fitted gam component.

Usage

```
## S3 method for class 'gam'
draw(
  object,
  data = NULL,
  select = NULL,
  parametric = NULL,
  terms = NULL,
  residuals = FALSE,
  scales = c("free", "fixed"),
  ci_level = 0.95,
  n = 100,
  n_2d = 50,
  n_3d = 16,
  n_4d = 4,
  unconditional = FALSE,
  overall_uncertainty = TRUE,
  constant = NULL,
  fun = NULL,
  dist = 0.1,
  rug = TRUE,
  distinct_rug = TRUE,
  contour = TRUE,
  grouped_by = deprecated(),
  ci_alpha = 0.2,
  ci_col = "black",
  smooth_col = "black",
  resid_col = "steelblue3",
  contour_col = "black",
  n_contour = NULL,
  partial_match = FALSE,
  discrete_colour = NULL,
  discrete_fill = NULL,
  continuous_colour = NULL,
  continuous_fill = NULL,
  position = "identity",
  angle = NULL,
  ncol = NULL,
  nrow = NULL,
  guides = "keep",
  widths = NULL,
  heights = NULL,
  crs = NULL,
  default_crs = NULL,
  lims_method = "geometry_bbox",
  wrap = TRUE,
  caption = TRUE,
  envir = NULL,
```

```
  surface = c("raster", "tile", "3d"),
  surface_3d = list(),
  ci_3d = FALSE,
  ci_3d_alpha = 0.15,
  ci_3d_col = "grey60",
  back_transform = FALSE,
  re_levels = NULL,
  mrf_plot = c("effects", "qq"),
  interval = c("confidence", "simultaneous"),
  n_sim = 10000,
  n_cores = 1,
  seed = NULL,
  interval_method = c("auto", "gaussian", "bootstrap", "shape"),
  frequentist = FALSE,
  shape_source = c("gaussian", "bootstrap"),
  ...,
  variables = NULL,
  simultaneous_method = c("simulation", "tube"),
  simultaneous_domain = NULL,
  tube_control = list(),
  cyclic = c("cartesian", "polar"),
  polar_args = list(),
  grouped = c("sz", "fs"),
  show_ci = c("factor-by", "sz"),
  time_varying_style = c("ribbon", "pointrange")
)

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

## S3 method for class 'scam'
draw(object, ..., overall_uncertainty = FALSE, centre = c("model", "plot"))

## S3 method for class 'scasm'
draw(
  object,
  ...,
  overall_uncertainty = FALSE,
  interval_method = c("auto", "gaussian", "bootstrap", "shape")
)
```

Arguments

object	a fitted GAM, the result of a call to mgcv::gam().
data	an optional data frame that is used to supply the data at which the smooths will be evaluated and plotted. This is usually not needed, but is an option if you need fine control over exactly what data are used for plotting.
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 or `NULL`; plot parametric terms also? The default, `NULL`, includes parametric terms when `select = NULL`, but excludes them when smooths are selected explicitly. Use `TRUE` to include parametric terms regardless of `select`, or `FALSE` to draw only smooths (the previous default). The `terms` argument selects which parametric effects are plotted; `select` selects only smooths.

`terms` character; which model parametric terms should be drawn? The Default of `NULL` will plot all parametric terms that can be drawn. Interactions show their individual contributions to the linear predictor; main effects and intercepts are not added. Multivariate parametric terms use the grid and surface controls below, as described in `parametric_effects()`.

`residuals` logical; should partial residuals for a smooth be drawn? Ignored for anything but a simple univariate smooth. If the model does not provide working residuals and working weights, warns once and draws the smooths without partial residuals.

`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.

`n` numeric; the number of points over the range of the covariate at which to evaluate a univariate smooth.

`n_2d` numeric; the number of points along each of the first two axes of a smooth surface, including surface panels of higher-dimensional smooths. The default is 50 in plotting and plot-preparation functions. If `NULL`, use `n` instead. Ignored when evaluation `data` are supplied. Factor levels are retained, and curves with only one continuous covariate use `n`.

`n_3d` numeric; the number of new observations to generate for the third dimension of a 3D smooth. Factor conditioning dimensions retain at most this many levels, taking the first levels in fitted order. Explicit `re_levels` selections take precedence; supplied `data` are unchanged.

`n_4d` numeric; the number of new observations to generate for the dimensions higher than 2 (!) of a kD smooth ($k >= 4$). For example, if the smooth is a 4D smooth, each of dimensions 3 and 4 will get `n_4d` new observations. For factor conditioning dimensions, this is the maximum number of levels retained per dimension, taking the first in fitted order. Explicit `re_levels` selections take precedence; supplied `data` are unchanged.

`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 errors of smooth and parametric effects? For parametric effects this includes the intercept variance and its covariance with the term, without changing

the partial estimates. Each term uses the intercept of its own linear predictor. Native scasm models are routed to `draw.scasm()` with `overall_uncertainty = FALSE`. An explicit request for unavailable mean uncertainty warns once. The stored model class is unchanged.

`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. For simultaneous bands it must be monotone over the entire band; endpoint transformation alone does not preserve non-monotone band coverage. Function `fun` will be applied after adding any `constant`, if provided.

`dist` numeric; if greater than 0, this is used to determine when a location is too far from data to be plotted when plotting 2-D smooths. The data are scaled into the unit square before deciding what to exclude, and `dist` is a distance within the unit square. See `mgcv::exclude.too.far()` for further details.

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

`distinct_rug` logical; should the data for the rug plots be reduced to the distinct values?

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

`grouped_by` Deprecated; use `grouped`. Explicit `TRUE` adds `"factor-by"` to `grouped`; `FALSE` removes it, retaining the other types.

`ci_alpha` Deprecated; use `gratia_theme_get()` theme settings instead. Numeric; alpha transparency for confidence or simultaneous interval.

`ci_col` Deprecated; use `gratia_theme_get()` theme settings instead. Colour specification for the confidence/credible intervals band. Affects the fill of the interval.

`smooth_col` Deprecated; use `gratia_theme_get()` theme settings instead. Colour specification for the smooth line.

`resid_col` Deprecated; use `gratia_theme_get()` theme settings instead. Colour specification for the partial residuals.

`contour_col` Deprecated; use `gratia_theme_get()` theme settings instead. 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()`.

`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.

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 non-sf layers added to spherical plots. The default, NULL, uses 4326 (WGS84). Internally generated surface cells, contours and observation points have their own geographic CRS, with longitude on the horizontal axis and latitude on the vertical axis. See ggplot2::coord_sf() for more details.

lims_method — character; affects how spherical plot axis limits are determined. The default, "geometry_bbox", uses the rendered geometry. Passed to ggplot2::coord_sf(). Spherical surfaces use explicit geographic cells with valid polar and date-line boundaries. Orthographic plots clip cells, contours and observation points to the visible hemisphere before projection. Explicit prediction data on a rectilinear grid retain their actual spacing; scattered or singleton coordinates are drawn as points, with a message, because they do not define surface cells. Incomplete observation rug coordinates are omitted. Grouped factor-by spherical surfaces use separate facets for each factor level.

wrap — logical; wrap plots as a patchwork? If FALSE, return a list of ggplot objects, one for each displayed panel. Separate factor levels can produce several plots for one smooth term.

caption — logical; show the smooth type in the caption of each plot?

envir — an optional environment for looking up data, functions and constants. Defaults to the model's evaluation environment.

surface — character; one of "raster" (the default), "tile", or "3d". Raster surfaces keep large PDF plots compact; tiles are borderless vector rectangles. Three-dimensional surfaces use the optional ggcube package, with the displayed partial effect as height and fill. Numeric smooth and parametric surfaces, including conditioning facets and soap films, are supported. Curves and factor estimates keep their usual display; spherical smooths retain their geographic rendering. Covariate scale changes that produce irregular spacing use rectangles bounded by adjacent coordinate midpoints for flat surfaces.

surface_3d — named list of view options passed to ggcube::coord_3d(), for example list(yaw = 45, persp = FALSE). The default list() uses the default view with ggcube::light(backface_scale = 1), preserving equal horizontal units for isotropic smooths. Supply light = ggcube::light(...) in this list to override lighting; ggcube::light(method = "none") disables it. An explicitly supplied light replaces the entire default

light. Only the coordinate system is used; ggcube's theme settings are not applied. These options affect all 3D panels in a call. For different views per term, modify individual plots returned by `assemble()`. The `dist` entry here controls perspective; the top-level `dist` argument controls masking.

`ci_3d` logical; add lower and upper interval surfaces to numeric smooth and parametric effect panels when `surface = "3d"`? Defaults to `FALSE`. Applies across supported model classes and conditioning facets. Other panel types, including standard errors, retain their usual display. Smooth bounds follow `interval`; parametric bounds remain pointwise.

`ci_3d_alpha` Deprecated; use `gratia_theme_get()` theme settings instead. Numeric between 0 and 1; opacity of both interval surfaces, independent of `ci_alpha`. Defaults to `0.15`.

`ci_3d_col` Deprecated; use `gratia_theme_get()` theme settings instead. One R colour; fill of both interval surfaces, independent of `ci_col` and the estimate's continuous fill scale. Defaults to `"grey60"`.

`back_transform` logical; plot covariates on their original scale? Defaults to `FALSE`, using model coordinates such as `log(x)` for a term containing `log(x)`. With `TRUE`, recover the raw covariates, generate a grid on their original scale and evaluate the model expressions forwards. Applies to axes, rugs, partial-residual locations and conditioning facets, including tensor products and parametric interactions. Each smooth axis must depend on one distinct original model covariate. Functions such as `foo(x)` are supported when their raw inputs and functions can be recovered; no inverse function is needed. Missing raw data or unavailable functions produce an error; supply `data` or `envir` as appropriate. With supplied prediction `data`, unavailable raw fitting observations omit rugs and partial residuals; prediction rows are never used as observations. For precomputed smooth estimates, prepare raw grids with `smooth_estimates(model, back_transform = TRUE)` before drawing with `back_transform = TRUE`. Raw inputs supplied through `data` also retain their coordinates. Transformed-only estimates cannot be relabelled on the raw scale, including grids carrying typical values from `include_all = TRUE`. Transformed soap-film boundaries lack retained raw coordinates and must be drawn with `back_transform = FALSE`. Parametric estimates retain both scales; multi-column basis expansions retain raw axes on either setting. Does not transform estimated effects or their intervals; use `fun` for that.

`re_levels` named list of factor level labels to retain in tensor products with random-effect margins, for example `list(species = c("a", "b"))`. Fitted factor order and ordered status are retained.

`mrf_plot` character; display Markov random field smooths as area-labelled estimates and intervals (`"effects"`, the default), or a descriptive Gaussian quantile plot (`"qq"`). MRF effects are correlated and need not have equal marginal variances; the QQ display is not a test of independent normal effects. Specialist maps and graph plots are outside the scope of this display.

`interval` character; `"confidence"` (default) draws pointwise intervals; `"simultaneous"` uses a band across, by default, the finite evaluation grid of each smooth, including its slices and factor levels. Separate smooths are not a joint coverage set.

"credible" aliases "confidence", reflecting the approximate Bayesian credible interpretation of default-covariance bands. Parametric panels retain pointwise intervals and are labelled accordingly when simultaneous smooth bands are requested. Bands appear where the renderer displays interval bounds, including 3D effect surfaces when ci_3d = TRUE. Random-effect QQ plots retain their usual display. simultaneous_method = "tube" calibrates supported univariate bands over a continuous interval; see simultaneous_bands.

n_sim, n_cores — positive integers; coefficient draws and simulation cores for simultaneous bands. Parallel simulation requires OpenMP support.

seed — integer or NULL; an explicit simulation seed preserves the caller's RNG state.

interval_method — character; "auto" uses stored scasm bootstrap draws for pointwise intervals when available, and Gaussian intervals otherwise. Use "gaussian" to request covariance-based limits or "bootstrap" to require stored draws. Simultaneous bootstrap bands are not supported. "shape" applies shape_intervals() to supported scasm smooth components, using the initial limits chosen by shape_source.

frequentist — logical; use frequentist covariance for smooth-component intervals? The default uses approximate Bayesian covariance. Parametric effect intervals retain their existing Bayesian covariance calculation.

shape_source — character; initial uncertainty for interval_method = "shape": "gaussian" (the default) or stored "bootstrap" pointwise limits. Bootstrap sources require frequentist = FALSE and unconditional = FALSE because the stored draws hold smoothing fixed.

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

variables — Character vector giving every covariate of the selected smooths exactly once, in plotting order: x, y, then the conditioning covariate for 3D smooths, or facet rows and columns for 4D smooths. When supplied, selected smooths must use the same covariates. Use fitted coordinate names, such as "log(x)" for a transformed covariate. The default, NULL, favours a two-variable tensor margin for x and y, otherwise retaining fitted order. Grid resolution follows this order. Supplied data are evaluated unchanged. Drawing saved estimates can change the order but uses only points already evaluated; evaluate again to change grid resolution or conditioning values. Continuous covariates must precede factors. Spherical smooths retain their latitude/longitude convention and do not support reordering.

simultaneous_method — character; calibration for simultaneous bands. The default, "simulation", uses coefficient draws on the evaluation grid. "tube" uses a conservative Gaussian threshold-crossing bound over a continuous interval for supported one-dimensional cubic splines. See simultaneous_bands.

simultaneous_domain — numeric pair of increasing finite endpoints, or a named list of such pairs keyed by spline coordinate. For tube bands, NULL uses the range of returned evaluation coordinates, including gaps. All returned points must lie inside the interval and the interval must lie inside the fitted spline support. Requires tube calibration.

tube_control — named list controlling numerical tube calibration: rel_tol (default 1e-6), abs_tol (1e-8), subdivisions (1000L), and root_tol (1e-8). Requires tube calibration. Simulation controls n_sim, n_cores (or ncores), and seed are ignored by tube calibration.

cyclic — character; use Cartesian coordinates ("cartesian", the default) or polar coordinates ("polar") for one-dimensional cyclic smooths. Other smooths and parametric terms keep their usual displays.

polar_args — named list of options passed to ggplot2::coord_radial(). Accepts start, reverse, inner.radius, r.axis.inside, rotate.angle and clip. The radial axis is inside the circle by default (r.axis.inside = TRUE); supplied values override this default. Other options use their ggplot2 defaults. Requires cyclic = "polar" when nonempty. The angular variable, fitted cycle boundaries, full-circle extent and angular expansion are set by gratia.

grouped — character or FALSE; factor smooth types whose curves share a panel. Choose any of "factor-by", "sz", and "fs", or FALSE to separate all their curves. The default, c("sz", "fs"), overlays sz and fs curves and separates factor-by curves. "factor-by" includes ordered factors. In reference-plus-deviation models the reference smooth remains separate; only fitted nonreference deviations are grouped. For sz terms with several factors, separate plots show each evaluated combination of levels. Applies to curves with one continuous covariate; existing surface displays are unchanged. Separate curves use the usual univariate styling: a black line and a translucent black interval by default, with colours and opacity controlled by the smooth theme settings.

show_ci — character or FALSE; factor smooth types whose curves show intervals. Choose any of "factor-by", "sz", and "fs", or FALSE to hide all their ribbons. The default, c("factor-by", "sz"), retains existing interval visibility; including "fs" enables its intervals. Controls display only, independently of interval and grouped. Ordinary smooths and parametric effects retain their intervals.

time_varying_style — Display sdmTMB time-varying coefficients using a line, points and interval ribbon ("ribbon", the default), or points and interval bars with a connecting line ("pointrange"). Other model terms keep their usual displays.

centre — character; for SCAM models, "model" retains each fitted smooth contribution. "plot" subtracts the evaluation-grid mean from each bivariate smooth to match scam::plot.scam(). The same linear transformation is applied to its covariance. Univariate curves and smooth_estimates() retain the fitted centring for either setting.

Details

For models with multiple linear predictors, smooth and parametric plot subtitles identify the model part for glmmTMB models or the distributional parameter for mgcv models. Single-predictor models omit these automatic labels; by-variable information and custom subtitles are still shown. Constant glmmTMB dispersion alone does not count as an additional predictor. For example, gaulss() uses "Mean (mu)" and "Standard deviation (sigma)", while twlss() uses "Mean (mu)", "Power (p)" and "Dispersion (phi)". Shared smooths name each predictor to which they contribute. Families without a known parameter mapping use numbered linear predictors. The existing smooth titles

and select values are unchanged, and all smooths are drawn by default. By-variable information is appended to the default subtitle. Supply subtitle through ... to replace it, or subtitle = "" to suppress it; NULL uses the default. Custom title, xlab and ylab can also be supplied through .... Parametric terms use the same predictor labels, including terms shared across predictors. Saved parametric_effects() retain these labels.

Polar plots support cyclic cubic and cyclic P splines, including positive cyclic P splines from scam, factor-by smooths and one-dimensional tensor smooths with a cyclic margin. The covariate forms the angular axis and the partial effect forms the radial axis, retaining their original numerical values and units. By default the lower fitted cycle boundary is at twelve o'clock, with values increasing clockwise. scales = "fixed" uses a common radial effect range. Grouped smooths must have matching cycle boundaries. The radial axis retains ggplot2's usual expansion, leaving space between the interval band and the outer rug. Angular limits have no expansion. Radial tick labels are inside the circle beside their grid rings by default. Set polar_args = list(r.axis.inside = FALSE) to place them outside, or supply a numeric covariate value to position the axis. With automatic evaluation data, n points span both fitted cycle boundaries. Supplied data and saved estimates keep their evaluation points, so partial cycle coverage leaves a gap. Saved estimates require the cyclic_info attribute returned by smooth_estimates(); recompute older estimates to obtain fitted boundaries. Polar plots of transformed cyclic covariates require back_transform = FALSE.

Random-effect tensor plots put continuous covariates before factors. Two continuous covariates form a surface, with remaining factor levels and numeric slices forming conditioning panels, consistently for te(), ti() and t2(). One continuous covariate gives curves for categorical combinations; categorical-only terms give point ranges. Up to four covariates are supported. Numeric conditioning slices use n_3d or n_4d. For factor conditioning dimensions, these controls retain at most that many levels, taking the first levels in fitted order. Explicit re_levels selections take precedence. Supplied data and precomputed estimates retain their rows; select specific factor levels with re_levels, or use data_slice() to fix conditioning covariates. Observation layers do not introduce additional surface panels.

Three-dimensional faces retain the current diverging RdBu fill scale and effect-range rules, shared across conditioning facets. continuous_fill overrides the fill scale. Faces have no outlines; contour_col controls contour lines. Estimate faces are fully opaque. Faces, contours, and floor observations are depth-sorted together, so nearer faces hide the parts behind them. The existing theme and facet styling are retained.

The first two displayed covariates form the horizontal axes, and the effect (or standard error when requested) forms the vertical axis. Additional covariates retain their existing conditioning facets: n_2d controls surface resolution, while n_3d and n_4d control covariate slices. Fixed effect ranges are also used for the vertical axis. With ci_3d = TRUE, translucent lower and upper interval surfaces use the same mesh and masking as the estimate. Their heights expand the vertical range, while the continuous fill scale retains its existing range rules. ci_3d_alpha and ci_3d_col style these surfaces separately from univariate interval ribbons.

Contours follow the displayed mesh, and available observations are projected onto its floor. Observations are not assigned to conditioning slices. Soap boundary loops are also projected onto the floor.

Supplied scattered evaluation data use piecewise-linear Delaunay triangles and require the optional geometry package. Support masking and soap-domain checks require sf. Faces crossing known excluded regions are discarded conservatively. Fully masked slices remain empty; insufficient non-collinear locations or conflicting heights at one location are errors.

Value

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

Label layout

Pass `label_control = draw_label_control(...)` to `draw()` to wrap discrete axis labels and legend text, stagger discrete tick labels, or adjust legend and colour-bar layout. Settings apply to each panel, including unwrapped plot lists. See `draw_label_control()` for the available settings.

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 data and fit a GAM
df1 <- data_sim("eg1", n = 200, dist = "normal", scale = 2, seed = 2)
m1 <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = df1, method = "REML")

# Plot all smooths
draw(m1)

# Draw a complete cycle in polar coordinates, using fitted knot boundaries
m_cycle <- gam(y ~ s(x0, bs = "cc", k = 10), data = df1,
  knots = list(x0 = c(0, 1)), method = "REML")
draw(m_cycle, cyclic = "polar")

# See https://gavinsimpson.github.io/gratia/articles/custom-plotting.html
# for more examples. Use the & operator to modify the theme of all panels.
```

---

draw.gamlss *Plot smooths of models estimated by GJRM*

Description

Provides `draw()` and `assemble()` methods for GAMLSS and joint models fitted by `GJRM::gamlss()` or `GJRM::gjrm()`. `assemble()` returns the individual plots; `draw()` arranges them using `patchwork::wrap_plots()`.

Usage

```
## S3 method for class 'gamlss'
draw(
  object,
  scales = c("free", "fixed"),
  ncol = NULL,
  nrow = NULL,
  guides = "keep",
  widths = NULL,
  heights = NULL,
  ...,
  n = 100,
  envir = NULL,
  select = NULL,
  parametric = NULL,
  terms = NULL,
  partial_match = FALSE,
  wrap = TRUE
)

## S3 method for class 'gamlss'
assemble(
  object,
  scales = c("free", "fixed"),
  ...,
  residuals = FALSE,
  n = 100,
  envir = NULL,
  select = NULL,
  parametric = NULL,
  terms = NULL,
  partial_match = FALSE,
  grouped_by = deprecated(),
  grouped = c("sz", "fs"),
  show_ci = c("factor-by", "sz")
)

## S3 method for class 'gjrm'
draw(
  object,
  scales = c("free", "fixed"),
  ncol = NULL,
  nrow = NULL,
  guides = "keep",
  widths = NULL,
  heights = NULL,
  ...,
  n = 100,
```

```
  envir = NULL,
  select = NULL,
  parametric = NULL,
  terms = NULL,
  partial_match = FALSE,
  wrap = TRUE
)

## S3 method for class 'gjrm'
assemble(
  object,
  scales = c("free", "fixed"),
  ...,
  residuals = FALSE,
  n = 100,
  envir = NULL,
  select = NULL,
  parametric = NULL,
  terms = NULL,
  partial_match = FALSE,
  grouped_by = deprecated(),
  grouped = c("sz", "fs"),
  show_ci = c("factor-by", "sz")
)
```

Arguments

object	a model fitted by GJRM::gamlss() or GJRM::gjrm().
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 assemble.gam()
n	numeric; the number of points over the range of the covariate at which to evaluate a univariate smooth.
envir	an optional environment supplying functions and constants used in model expressions. The available model formula environment is used when NULL. Covariate observations should be supplied in data.
select	Smooth selection across all equations, using the labels from smooths(), or numeric or logical indices in equation and smooth order. Repeated labels receive

suffixes such as `s.1(x)`; existing distinct labels are retained. `partial_match` uses these model-wide labels.

`parametric` logical or `NULL`; plot parametric terms also? The default, `NULL`, includes parametric terms when `select = NULL`, but excludes them when smooths are selected explicitly. Use `TRUE` to include parametric terms regardless of `select`, or `FALSE` to draw only smooths (the previous default). The `terms` argument selects which parametric effects are plotted; `select` selects only smooths.

`terms` character; which model parametric terms should be drawn? The Default of `NULL` will plot all parametric terms that can be drawn. Interactions show their individual contributions to the linear predictor; main effects and intercepts are not added. Multivariate parametric terms use the grid and surface controls below, as described in `parametric_effects()`.

`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.

`wrap` Logical; return an assembled patchwork when `TRUE` (the default), or a flat named list of plots when `FALSE`.

`residuals` logical; should partial residuals for a smooth be drawn? Ignored for anything but a simple univariate smooth. If the model does not provide working residuals and working weights, warns once and draws the smooths without partial residuals.

`grouped_by` Deprecated; use `grouped`. Explicit `TRUE` adds `"factor-by"` to `grouped`; `FALSE` removes it, retaining the other types.

`grouped` character or `FALSE`; factor smooth types whose curves share a panel. Choose any of `"factor-by"`, `"sz"`, and `"fs"`, or `FALSE` to separate all their curves. The default, `c("sz", "fs")`, overlays `sz` and `fs` curves and separates factor-by curves. `"factor-by"` includes ordered factors. In reference-plus-deviation models the reference smooth remains separate; only fitted nonreference deviations are grouped. For `sz` terms with several factors, separate plots show each evaluated combination of levels. Applies to curves with one continuous covariate; existing surface displays are unchanged. Separate curves use the usual univariate styling: a black line and a translucent black interval by default, with colours and opacity controlled by the smooth theme settings.

`show_ci` character or `FALSE`; factor smooth types whose curves show intervals. Choose any of `"factor-by"`, `"sz"`, and `"fs"`, or `FALSE` to hide all their ribbons. The default, `c("factor-by", "sz")`, retains existing interval visibility; including `"fs"` enables its intervals. Controls display only, independently of `interval` and `grouped`. Ordinary smooths and parametric effects retain their intervals.

Value

`assemble()` and `draw(wrap = FALSE)` return a flat named list of plots, ordered by equation and then by term within each equation. Smooth names use the model-wide labels returned by `smooths()`. `draw()` returns a patchwork by default. Both methods return invisible `NULL` when there are no plots to draw.

Label layout

Pass `label_control = draw_label_control(...)` to `draw()` to wrap discrete axis labels and legend text, stagger discrete tick labels, or adjust legend and colour-bar layout. Settings apply to each panel, including unwrapped plot lists. See `draw_label_control()` for the available settings.

Note

For models with multiple linear predictors, smooth and parametric plot subtitles identify the distributional parameter using the parent model's margin and equation order, rather than the working family of the embedded GAMs. Univariate GAMLSS models with one predictor omit these automatic labels. Joint models retain equation labels even when only dependence is estimated. Existing smooth labels, selection, by-variable information and custom subtitles are retained. Pass `surface = "3d", ci_3d = TRUE` through `...` to add interval surfaces to smooths and numeric parametric interactions in any equation. `ci_3d_alpha` and `ci_3d_col` control their opacity and fill colour. Pass `cyclic = "polar"` and optional `polar_args` through `...` for polar plots of supported one-dimensional cyclic smooths in any equation; see `draw.gam()` for the coordinate options.

Joint models use the final joint coefficients and covariance for every embedded GAM, including dependence equations and random-effect smooths. Parametric plot subtitles also identify their response and parameter. For both univariate and joint models, a parametric term repeated across equations has list names such as `x (equation 1)` and `x (equation 2)`; unique terms keep their original names. Grouped factor-by panels also receive an equation suffix when their base name would hide another smooth's selection label. Ordinal cutpoints are excluded from smooth and intercept uncertainty. Fixed-margin fits plot their estimated dependence equation only. Flexible survival equations fitted using SCAM are not yet supported. Working residuals from preliminary GAM fits are unavailable for joint models; requesting partial residuals warns and draws the smooths alone.

Examples

```
if (requireNamespace("GJRM", quietly = TRUE)) {
  load_mgcv()
  # Simulate a nonlinear mean with a binary group effect and Gaussian noise
  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)

  # Fit smooths for the mean and scale, then plot both model components
  m <- GJRM::gamlss(list(y1 ~ x1 + s(x2), ~ s(x3, k = 6)), data = dataSim)
  draw(m)
}
```

`draw.mgcv_smooth` *Plot basis functions*

Description

Plots basis functions using ggplot2

Usage

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

Arguments

`object`	an object, the result of a call to `basis()`.
`legend`	logical; should a legend by 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.

Label layout

Pass label_control = draw_label_control(...) to draw() to wrap discrete axis labels and legend text, stagger discrete tick labels, or adjust legend and colour-bar layout. Settings apply to each panel, including unwrapped plot lists. See draw_label_control() for the available settings.

Author(s)

Gavin L. Simpson

Examples

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

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

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

---

draw.pairwise_concurvity

Plot concurvity measures

Description

Plot concurvity measures

Usage

```
## 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_colour	function; continuous colour (fill) scale to use.
...	arguments passed to other methods.
bar_col	Deprecated; use gratia_theme_get() theme settings instead. Colour specification for the bar colour.
bar_fill	Deprecated; use gratia_theme_get() theme settings instead. Colour specification for the bar fill

Label layout

Pass label_control = draw_label_control(...) to draw() to wrap discrete axis labels and legend text, stagger discrete tick labels, or adjust legend and colour-bar layout. Settings apply to each panel, including unwrapped plot lists. See draw_label_control() for the available settings.

---

draw.parametric_effects

Plot estimated effects for model parametric terms

Description

Plot estimated effects for model parametric terms

Usage

```
## S3 method for class 'parametric_effects'
draw(
  object,
  scales = c("free", "fixed"),
  ci_level = 0.95,
  ci_col = "black",
  ci_alpha = 0.2,
  line_col = "black",
  constant = NULL,
  fun = NULL,
  rug = TRUE,
  position = "identity",
  angle = NULL,
  ...,
  ncol = NULL,
  nrow = NULL,
  guides = "keep",
  contour = TRUE,
  contour_col = "black",
  n_contour = NULL,
  surface = c("raster", "tile", "3d"),
  surface_3d = list(),
  ci_3d = FALSE,
  ci_3d_alpha = 0.15,
  ci_3d_col = "grey60",
  continuous_fill = NULL,
  discrete_colour = NULL,
  discrete_fill = NULL,
  back_transform = FALSE,
  time_varying_style = c("ribbon", "pointrange"),
  subtitle = NULL
)
```

Arguments

object	A `parametric_effects` object, optionally nested.
scales	Use a common effect-axis range for curves and points (`fixed`), or separate ranges (`free`). Surface covariate axes are not affected.
ci_level	numeric between 0 and 1; the coverage of credible interval.
ci_col	Deprecated; use `gratia_theme_get()` theme settings instead. Colour specification for the confidence/credible intervals band. Affects the fill of the interval.
ci_alpha	Deprecated; use `gratia_theme_get()` theme settings instead. Numeric; alpha transparency for confidence or simultaneous interval.
line_col	Deprecated; use `gratia_theme_get()` theme settings instead. 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. For simultaneous bands it must be monotone over the entire band; endpoint transformation alone does not preserve non-monotone band coverage. 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()
contour	logical; should contours be draw on the plot using ggplot2::geom_contour().
contour_col	Deprecated; use gratia_theme_get() theme settings instead. 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().
surface	character; one of "raster" (the default), "tile", or "3d". Raster surfaces keep large PDF plots compact; tiles are borderless vector rectangles. Three-dimensional surfaces use the optional ggcube package, with the displayed partial effect as height and fill. Numeric smooth and parametric surfaces, including conditioning facets and soap films, are supported. Curves and factor estimates keep their usual display; spherical smooths retain their geographic rendering. Covariate scale changes that produce irregular spacing use rectangles bounded by adjacent coordinate midpoints for flat surfaces.
surface_3d	named list of view options passed to ggcube::coord_3d(), for example list(yaw = 45, persp = FALSE). The default list() uses the default view with ggcube::light(backface_scale = 1), preserving equal horizontal units for isotropic smooths. Supply light = ggcube::light(...) in this list to override lighting; ggcube::light(method = "none") disables it. An explicitly supplied light replaces the entire default light. Only the coordinate system is used; ggcube's theme settings are not applied. These options affect all 3D panels in a call. For different views per term, modify individual plots returned by assemble(). The dist entry here controls perspective; the top-level dist argument controls masking.
ci_3d	logical; add lower and upper interval surfaces to numeric smooth and parametric effect panels when surface = "3d"? Defaults to FALSE. Applies across supported model classes and conditioning facets. Other panel types, including standard errors, retain their usual display. Smooth bounds follow interval; parametric bounds remain pointwise.

ci_3d_alpha Deprecated; use gratia_theme_get() theme settings instead. Numeric between 0 and 1; opacity of both interval surfaces, independent of ci_alpha. Defaults to 0.15.

ci_3d_col Deprecated; use gratia_theme_get() theme settings instead. One R colour; fill of both interval surfaces, independent of ci_col and the estimate's continuous fill scale. Defaults to "grey60".

continuous_fill
a suitable fill scale to be used when plotting continuous variables.

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.

back_transform logical; plot covariates on their original scale? Defaults to FALSE, using model coordinates such as log(x) for a term containing log(x). With TRUE, recover the raw covariates, generate a grid on their original scale and evaluate the model expressions forwards. Applies to axes, rugs, partial-residual locations and conditioning facets, including tensor products and parametric interactions. Each smooth axis must depend on one distinct original model covariate. Functions such as foo(x) are supported when their raw inputs and functions can be recovered; no inverse function is needed. Missing raw data or unavailable functions produce an error; supply data or envir as appropriate. With supplied prediction data, unavailable raw fitting observations omit rugs and partial residuals; prediction rows are never used as observations. For precomputed smooth estimates, prepare raw grids with smooth_estimates(model, back_transform = TRUE) before drawing with back_transform = TRUE. Raw inputs supplied through data also retain their coordinates. Transformed-only estimates cannot be relabelled on the raw scale, including grids carrying typical values from include_all = TRUE. Transformed soap-film boundaries lack retained raw coordinates and must be drawn with back_transform = FALSE. Parametric estimates retain both scales; multi-column basis expansions retain raw axes on either setting. Does not transform estimated effects or their intervals; use fun for that.

time_varying_style
Display sdmTMB time-varying coefficients using a line, points and interval ribbon ("ribbon", the default), or points and interval bars with a connecting line ("pointrange"). Other model terms keep their usual displays.

subtitle Character or expression; replace the automatic predictor subtitle on every panel. NULL uses the automatic label and "" suppresses it.

Label layout

Pass label_control = draw_label_control(...) to draw() to wrap discrete axis labels and legend text, stagger discrete tick labels, or adjust legend and colour-bar layout. Settings apply to each panel, including unwrapped plot lists. See draw_label_control() for the available settings.

draw.penalty_df *Display penalty matrices of smooths using* ggplot

Description

Displays the penalty matrices of smooths as a heatmap using ggplot

Usage

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

Arguments

object	a fitted GAM, the result of a call to 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 draw 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 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.
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()
geom	character; either "raster" (the default) or "tile". Raster rendering uses an image for the matrix, which can keep large PDF plots compact. Tile rendering uses individual rectangles, providing vector cells in PDF and SVG output, but can increase file size for large matrices.
...	additional arguments passed to patchwork::wrap_plots().

Label layout

Pass label_control = draw_label_control(...) to draw() to wrap discrete axis labels and legend text, stagger discrete tick labels, or adjust legend and colour-bar layout. Settings apply to each panel, including unwrapped plot lists. See draw_label_control() for the available settings.

Examples

```
load_mgcv()
# Fit different spline bases to simulated Gaussian responses
dat <- data_sim("eg1", n = 200, seed = 42)
m <- gam(y ~ s(x0) + s(x1, bs = "cr") + s(x2, bs = "bs"),
  data = dat, method = "REML")

# Display the penalty matrices for the fitted smooths
draw(penalty(m))
```

---

draw.rootogram *Draw a rootogram*

Description

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

Usage

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

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

Arguments

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

Value

A 'ggplot' object.

Label layout

Pass `label_control = draw_label_control(...)` to `draw()` to wrap discrete axis labels and legend text, stagger discrete tick labels, or adjust legend and colour-bar layout. Settings apply to each panel, including unwrapped plot lists. See `draw_label_control()` for the available settings.

References

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

See Also

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

Examples

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

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

# plot the rootogram
draw(rg)

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

`draw.sampling_diagnostics` *Plot coefficient-chain diagnostics*

Description

Native ggplot2 displays of retained coefficient chains. Trace plots show movement over iterations, including repeated values after MH rejections. Rank plots compare overlaid histogram outlines for the ranks of each coefficient pooled across chains. Chains exploring the same distribution should have approximately uniform rank histograms. These plots complement the numerical summaries; similar-looking chains alone do not establish convergence.

Usage

```
## S3 method for class 'sampling_diagnostics'
draw(
  object,
  type = c("trace", "rank"),
  select = NULL,
  n_bins = 20L,
  scales = NULL,
  ncol = NULL,
  nrow = NULL,
  alpha = 0.7,
  linewidth = 0.4,
  discrete_colour = NULL,
  ...
)
```

Arguments

`object`	A `sampling_diagnostics()` object.

type	Either "trace" or "rank".
select	Coefficients to plot, as exact names, positive integer positions, or a logical vector with one entry per coefficient. The default plots every available coefficient.
n_bins	Positive integer; number of common pooled-rank bins. Used only for rank plots. Tied draws receive their average rank, without jitter.
scales	Facet scales, either "fixed" or "free". By default trace plots have separate coefficient scales and rank plots share a count scale.
ncol, nrow	Number of columns or rows in the panel layout.
alpha	Deprecated; use gratia_theme_get() theme settings instead. Transparency of the chain lines.
linewidth	Deprecated; use gratia_theme_get() theme settings instead. Width of the chain lines.
discrete_colour	An optional ggplot2 discrete colour scale. Defaults follow gratia's smooth plots: Okabe-Ito for up to nine chains, then hue.
...	Additional arguments passed to ggplot2::facet_wrap().

Details

The trace x axis counts retained iterations within each chain, rather than the globally unique draw number. Burn-in is discarded by the samplers and cannot be displayed. With thinning, successive plotted iterations represent successive retained draws, not successive MH transitions.

Rank plots require at least two chains. Ranks are computed separately for each selected coefficient, pooling all chains. A dashed line shows the average count per bin. Independent continuous draws from the same distribution should give approximately flat histograms. Autocorrelation and ties affect the histogram variation; the reference line is not a confidence bound.

Plots always describe coefficient draws, including when diagnostics are extracted from response samples or smooths. For transformed quantities use as_mcmc_draws() with external diagnostic tools. Sampling with diagnostics = TRUE retains coefficient draws for plotting from those outputs. Coefficient matrices can be plotted whenever their sampling metadata are available, even without numerical diagnostic summaries.

Value

A ggplot2::ggplot() object.

Label layout

Pass label_control = draw_label_control(...) to draw() to wrap discrete axis labels and legend text, stagger discrete tick labels, or adjust legend and colour-bar layout. Settings apply to each panel, including unwrapped plot lists. See draw_label_control() for the available settings.

See Also

sampling_diagnostics(), as_mcmc_draws()

Examples

```
load_mgcv()
dat <- data_sim("eg1", n = 100, seed = 42)
model <- gam(y ~ s(x0, k = 5), data = dat, method = "REML")
b <- post_draws(model, n_draws = 400, chains = 4, method = "mh",
  burnin = 100, seed = 281)
diagnostics <- sampling_diagnostics(b)
draw(diagnostics, select = 1:2)
draw(diagnostics, type = "rank", select = 1:2, n_bins = 10)
```

---

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

Description

Plot the result of a call to smooth_estimates()

Usage

```
## S3 method for class 'smooth_estimates'
draw(
  object,
  constant = NULL,
  fun = NULL,
  contour = TRUE,
  grouped_by = deprecated(),
  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 = "geometry_bbox",
  caption = TRUE,
  surface = c("raster", "tile", "3d"),
```

```
  surface_3d = list(),
  ci_3d = FALSE,
  ci_3d_alpha = 0.15,
  ci_3d_col = "grey60",
  back_transform = FALSE,
  re_levels = NULL,
  mrf_plot = c("effects", "qq"),
  interval = NULL,
  ...,
  variables = NULL,
  cyclic = c("cartesian", "polar"),
  polar_args = list(),
  grouped = c("sz", "fs"),
  show_ci = c("factor-by", "sz")
)
```

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. For simultaneous bands it must be monotone over the entire band; endpoint transformation alone does not preserve non-monotone band coverage. 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	Deprecated; use grouped. Explicit TRUE adds "factor-by" to grouped; FALSE removes it, retaining the other types.
contour_col	Deprecated; use gratia_theme_get() theme settings instead. 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	Deprecated; use gratia_theme_get() theme settings instead. Numeric; alpha transparency for confidence or simultaneous interval.
ci_col	Deprecated; use gratia_theme_get() theme settings instead. Colour specification for the confidence/credible intervals band. Affects the fill of the interval.
smooth_col	Deprecated; use gratia_theme_get() theme settings instead. Colour specification for the smooth line.
resid_col	Deprecated; use gratia_theme_get() theme settings instead. Colour specification for the partial residuals.
decrease_col, increase_col	Deprecated; use gratia_theme_get() theme settings instead. 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 Deprecated; use gratia_theme_get() theme settings instead. Numeric; the value to set the linewidth to in ggplot2::geom_line(), used to represent the periods of change.

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 non-sf layers added to spherical plots. The default, NULL, uses 4326 (WGS84). Internally generated surface cells, contours and observation points have their own geographic CRS, with longitude on the horizontal axis and latitude on the vertical axis. See ggplot2::coord_sf() for more details.

lims_method character; affects how spherical plot axis limits are determined. The default, "geometry_bbox", uses the rendered geometry. Passed to ggplot2::coord_sf(). Spherical surfaces use explicit geographic cells with valid polar and date-line boundaries. Orthographic plots clip cells, contours and observation points to the visible hemisphere before projection. Explicit prediction data on a rectilinear grid retain their actual spacing; scattered or singleton coordinates are drawn as points, with a message, because they do not define surface cells. Incomplete observation rug coordinates are omitted. Grouped factor-by spherical surfaces use separate facets for each factor level.

caption logical; show the smooth type in the caption of each plot?

surface character; one of "raster" (the default), "tile", or "3d". Raster surfaces keep large PDF plots compact; tiles are borderless vector rectangles. Three-dimensional surfaces use the optional ggcube package, with the displayed partial effect as height and fill. Numeric smooth and parametric surfaces, including conditioning facets and soap films, are supported. Curves and factor estimates keep their usual display; spherical smooths retain their geographic rendering. Covariate scale changes that produce irregular spacing use rectangles bounded by adjacent coordinate midpoints for flat surfaces.

surface_3d named list of view options passed to ggcube::coord_3d(), for example list(yaw = 45, persp = FALSE). The default list() uses the default view with ggcube::light(backface_scale = 1), preserving equal horizontal units for isotropic smooths. Supply light = ggcube::light(...) in this list to override lighting; ggcube::light(method = "none") disables it. An explicitly supplied light replaces the entire default

light. Only the coordinate system is used; ggcube's theme settings are not applied. These options affect all 3D panels in a call. For different views per term, modify individual plots returned by `assemble()`. The `dist` entry here controls perspective; the top-level `dist` argument controls masking.

`ci_3d` logical; add lower and upper interval surfaces to numeric smooth and parametric effect panels when `surface = "3d"`? Defaults to `FALSE`. Applies across supported model classes and conditioning facets. Other panel types, including standard errors, retain their usual display. Smooth bounds follow `interval`; parametric bounds remain pointwise.

`ci_3d_alpha` Deprecated; use `gratia_theme_get()` theme settings instead. Numeric between 0 and 1; opacity of both interval surfaces, independent of `ci_alpha`. Defaults to `0.15`.

`ci_3d_col` Deprecated; use `gratia_theme_get()` theme settings instead. One R colour; fill of both interval surfaces, independent of `ci_col` and the estimate's continuous fill scale. Defaults to `"grey60"`.

`back_transform` logical; plot covariates on their original scale? Defaults to `FALSE`, using model coordinates such as `log(x)` for a term containing `log(x)`. With `TRUE`, recover the raw covariates, generate a grid on their original scale and evaluate the model expressions forwards. Applies to axes, rugs, partial-residual locations and conditioning facets, including tensor products and parametric interactions. Each smooth axis must depend on one distinct original model covariate. Functions such as `foo(x)` are supported when their raw inputs and functions can be recovered; no inverse function is needed. Missing raw data or unavailable functions produce an error; supply `data` or `envir` as appropriate. With supplied prediction `data`, unavailable raw fitting observations omit rugs and partial residuals; prediction rows are never used as observations. For precomputed smooth estimates, prepare raw grids with `smooth_estimates(model, back_transform = TRUE)` before drawing with `back_transform = TRUE`. Raw inputs supplied through `data` also retain their coordinates. Transformed-only estimates cannot be relabelled on the raw scale, including grids carrying typical values from `include_all = TRUE`. Transformed soap-film boundaries lack retained raw coordinates and must be drawn with `back_transform = FALSE`. Parametric estimates retain both scales; multi-column basis expansions retain raw axes on either setting. Does not transform estimated effects or their intervals; use `fun` for that.

`re_levels` named list of factor level labels to retain in tensor products with random-effect margins, for example `list(species = c("a", "b"))`. Fitted factor order and ordered status are retained.

`mrf_plot` character; display Markov random field smooths as area-labelled estimates and intervals (`"effects"`, the default), or a descriptive Gaussian quantile plot (`"qq"`). MRF effects are correlated and need not have equal marginal variances; the QQ display is not a test of independent normal effects. Specialist maps and graph plots are outside the scope of this display.

`interval` character or `NULL`; the default draws stored interval bounds unchanged, adding pointwise bounds if absent. An explicit choice validates stored interval metadata; `"credible"` aliases `"confidence"`. Simultaneous bands must first be

computed from the fitted model by `smooth_estimates()`. Method and uncertainty controls such as `interval_method`, `shape_source`, `frequentist`, and `unconditional` also require the fitted model; they cannot change intervals when drawing saved estimates.

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

`variables` Character vector giving every covariate of the selected smooths exactly once, in plotting order: x, y, then the conditioning covariate for 3D smooths, or facet rows and columns for 4D smooths. When supplied, selected smooths must use the same covariates. Use fitted coordinate names, such as `"log(x)"` for a transformed covariate. The default, `NULL`, favours a two-variable tensor margin for x and y, otherwise retaining fitted order. Grid resolution follows this order. Supplied `data` are evaluated unchanged. Drawing saved estimates can change the order but uses only points already evaluated; evaluate again to change grid resolution or conditioning values. Continuous covariates must precede factors. Spherical smooths retain their latitude/longitude convention and do not support reordering.

`cyclic` character; use Cartesian coordinates (`"cartesian"`, the default) or polar coordinates (`"polar"`) for one-dimensional cyclic smooths. Other smooths and parametric terms keep their usual displays.

`polar_args` named list of options passed to `ggplot2::coord_radial()`. Accepts `start`, `reverse`, `inner.radius`, `r.axis.inside`, `rotate.angle` and `clip`. The radial axis is inside the circle by default (`r.axis.inside = TRUE`); supplied values override this default. Other options use their ggplot2 defaults. Requires `cyclic = "polar"` when nonempty. The angular variable, fitted cycle boundaries, full-circle extent and angular expansion are set by gratia.

`grouped` character or `FALSE`; factor smooth types whose curves share a panel. Choose any of `"factor-by"`, `"sz"`, and `"fs"`, or `FALSE` to separate all their curves. The default, `c("sz", "fs")`, overlays `sz` and `fs` curves and separates factor-by curves. `"factor-by"` includes ordered factors. In reference-plus-deviation models the reference smooth remains separate; only fitted nonreference deviations are grouped. For `sz` terms with several factors, separate plots show each evaluated combination of levels. Applies to curves with one continuous covariate; existing surface displays are unchanged. Separate curves use the usual univariate styling: a black line and a translucent black interval by default, with colours and opacity controlled by the smooth theme settings.

`show_ci` character or `FALSE`; factor smooth types whose curves show intervals. Choose any of `"factor-by"`, `"sz"`, and `"fs"`, or `FALSE` to hide all their ribbons. The default, `c("factor-by", "sz")`, retains existing interval visibility; including `"fs"` enables its intervals. Controls display only, independently of `interval` and `grouped`. Ordinary smooths and parametric effects retain their intervals.

Details

`cyclic = "polar"` uses fitted cycle boundaries stored in the `cyclic_info` attribute. Evaluation points are unchanged: estimates covering only part of a cycle leave a gap. To draw a complete cycle, supply both fitted boundary values when computing estimates, or use `draw(model, cyclic = "polar")` to generate the full-cycle grid. Recompute older saved estimates with `smooth_estimates()`

if their fitted cycle boundaries are unavailable. See `draw.gam()` for supported smooths and coordinate options.

Label layout

Pass `label_control = draw_label_control(...)` to `draw()` to wrap discrete axis labels and legend text, stagger discrete tick labels, or adjust legend and colour-bar layout. Settings apply to each panel, including unwrapped plot lists. See `draw_label_control()` for the available settings.

Examples

```
load_mgcv()
# Simulate an additive Gaussian response and fit a GAM
df <- data_sim("eg1", n = 200, seed = 21)
m <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = df, method = "REML")

# Evaluate one smooth and plot its estimates and interval
sm <- smooth_estimates(m, select = "s(x2)")
draw(sm)
```

`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().
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	Deprecated; use gratia_theme_get() theme settings instead. Numeric; alpha transparency for confidence or simultaneous interval.
colour	Deprecated; use gratia_theme_get() theme settings instead. 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	Deprecated; use gratia_theme_get() theme settings instead. 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().

Label layout

Pass label_control = draw_label_control(...) to draw() to wrap discrete axis labels and legend text, stagger discrete tick labels, or adjust legend and colour-bar layout. Settings apply to each panel, including unwrapped plot lists. See draw_label_control() for the available settings.

Author(s)

Gavin L. Simpson

Examples

```
load_mgcv()
# Simulate an additive Gaussian response and fit a GAM
dat <- data_sim("eg1", n = 200, scale = 1, seed = 1)
m <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = dat, method = "REML")

# Draw 15 possible curves for the smooth of x2 on a 100-point grid
sm <- smooth_samples(m, select = "s(x2)", n_grid = 100,
  n_draws = 15, seed = 23478)
draw(sm)
```

---

draw.variance_comp *Plot variance components and proportions of term variance*

Description

Draw estimates and their stored confidence intervals, with components on the x-axis by default. No model fitting or posterior sampling is performed.

Usage

```
## S3 method for class 'variance_comp'
draw(
  object,
  type = c("variance", "std_dev", "proportion"),
  select = NULL,
  include_total = TRUE,
  predictor = NULL,
  ncol = NULL,
  nrow = NULL,
```

```
  angle = NULL,
  n_dodge = 1L,
  flip = FALSE,
  ...
)
```

Arguments

object	A result from variance_comp().
type	What to plot: "variance" (the default), "std_dev", or "proportion". Proportions require variance_comp(..., type = "terms").
select	Components to plot, as exact names, positive integer positions, or a logical vector with one entry per row of object. Components are displayed from left to right in the requested order, or from top to bottom with flip = TRUE. Use positions when component names are duplicated.
include_total	Logical; include the total rows of term results? Defaults to TRUE. Selection positions refer to object before totals are removed.
predictor	Linear predictors to plot for type = "proportion", as positive integer positions or total labels ("total", "total.1", and so on). The default plots all predictors, with separate panels when more than one is present. Shared terms appear in each relevant panel.
ncol, nrow	Number of columns or rows for the panels.
angle	Numeric; angle of the component labels in degrees, between -90 and 90. Passed to ggplot2::guide_axis(). The default, NULL, uses the angle specified by the plot theme.
n_dodge	Positive integer; number of rows for component labels, or columns with flip = TRUE. Use, for example, n_dodge = 2 to stagger labels that would overlap. Passed to the n.dodge argument of ggplot2::guide_axis(). Defaults to 1.
flip	Logical; put components on the y-axis? Defaults to FALSE.
...	Not used. Modify the returned plot with ggplot2 layers, scales and themes.

Details

Variance plots square the stored standard-deviation interval bounds so that estimates and intervals use the same scale. Standard-deviation and proportion plots use their stored bounds directly. Components with unavailable intervals retain their points. Intervals are drawn when both bounds are finite. Undefined estimates have no point, but retain their component labels.

Proportions can be negative or greater than one, and the plot preserves those values. Each panel shows only terms belonging to that predictor and its own total. A predictor with zero total variance has undefined proportions and is displayed without points or intervals.

Points use the geom.gratia.effect theme element and intervals use geom.gratia.interval; see gratia_theme_get().

Value

A ggplot2::ggplot() object.

Label layout

Pass `label_control = draw_label_control(...)` to `draw()` to wrap discrete axis labels and legend text, stagger discrete tick labels, or adjust legend and colour-bar layout. Settings apply to each panel, including unwrapped plot lists. See `draw_label_control()` for the available settings.

See Also

`variance_comp()`, `gratia_theme_get()`

Examples

```
load_mgcv()
# Simulate linear and nonlinear effects with Gaussian noise
set.seed(42)
dat <- data.frame(x = runif(120), z = runif(120))
dat$y <- 2 * dat$z + sin(2 * pi * dat$x) + rnorm(120, sd = 0.3)

# Fit a linear effect of z and a smooth effect of x
model <- gam(y ~ z + s(x, k = 8), data = dat, method = "REML")

# Plot the variances used in the random-effect representation of the penalties
components <- variance_comp(model)
draw(components)

# Plot variation of fitted term contributions across the fitting observations
terms <- variance_comp(model, type = "terms", n_draws = 200, seed = 42)
draw(terms)
```

`draw_label_control` *Control label layout in gratia plots*

Description

Configure labels on axes, legends and colour bars across `draw()` and `assemble()` methods. Omitted settings retain the existing appearance.

Usage

```
draw_label_control(axes = list(), legend = list(), colourbar = list())
```

Arguments

`axes`	Named list with optional `wrap`, `n_dodge`, `angle` and `title_wrap` settings. `wrap` and `title_wrap` are positive integer widths in characters. Tick labels wrap at word boundaries, preserving explicit line breaks. Unbroken words are not split. `n_dodge` is the positive integer number of rows for horizontal-axis labels, or columns for vertical-axis labels. `angle` is a finite number of degrees between -90 and 90. Tick settings apply only to discrete axes; `title_wrap` applies to ordinary x and y axis titles, including continuous axes.

legend	Named list with optional wrap, title_wrap, nrow, ncol and byrow settings. Wrapping widths and row/column counts are positive integers. byrow is logical. These settings apply to key-based legends, including colour, fill, shape and linetype legends.
colourbar	Named list with optional title_wrap, angle, width and height settings. title_wrap is a positive integer width in characters; angle is between -90 and 90 degrees. Dimensions are positive, length-one grid::unit() objects and refer to the displayed bar's width and height. These settings apply to continuous colour and fill guides.

Details

Every setting defaults to NULL, retaining the method's existing behaviour. Pass the result as label_control to draw() or assemble(). The same configuration can be reused across plot types; absent guides are ignored. Mathematical expressions and missing labels are preserved. Wrapping changes displayed text after scale labelling, without changing factor levels, breaks, transformations or palettes. Numeric formatting remains controlled by the scale's labels argument.

Explicit axis settings override the existing angle and n_dodge plotting arguments on discrete axes. Hidden guides stay hidden. Native guide types and unrelated settings are retained; later ggplot2 scale and guide additions follow their usual precedence. Settings apply to panels before patchwork collects their guides, including plots returned with wrap = FALSE. Projected coordinate labels in 3D plots retain their renderer's settings; their ordinary legends and colour bars can use this control.

Value

A draw_label_control object.

See Also

draw(), assemble(), gratia_theme_get()

Examples

```
# Wrap categorical tick labels and stagger them across two rows.
layout <- draw_label_control(axes = list(wrap = 12, n_dodge = 2))

# Reuse one configuration for legend text and colour-bar layout.
layout <- draw_label_control(
  axes = list(wrap = 12),
  legend = list(wrap = 20, title_wrap = 20, ncol = 2),
  colourbar = list(title_wrap = 20, height = grid::unit(35, "mm"))
)

# Draw a model with the requested layout.
load_mgcv()
dat <- data_sim("eg4", n = 100, seed = 42)
model <- gam(y ~ fac + s(x2, by = fac, k = 5), data = dat, method = "REML")
draw(model, label_control = layout)
```

edf *Effective degrees of freedom for smooths and GAMs*

Description

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

Usage

```
edf(object, ...)

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

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

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

## S3 method for class 'scam'
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 EDF to extract. If NULL, the default, EDFs for all smooths will be returned. 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.
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

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

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

See Also

gamm4_support

Examples

```
load_mgcv()

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

# extract the EDFs for all smooths
edf(m)

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

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

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

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

`eval_smooth` *S3 methods to evaluate individual smooths*

Description

S3 methods to evaluate individual smooths

Usage

```
eval_smooth(smooth, ...)

## S3 method for class 'mgcv.smooth'
eval_smooth(
  smooth,
  model,
  n = 100,
  n_2d = NULL,
  n_3d = NULL,
  n_4d = NULL,
  data = NULL,
  unconditional = FALSE,
  frequentist = FALSE,
  overall_uncertainty = TRUE,
  dist = NULL,
  envir = NULL,
  back_transform = FALSE,
  ...
)

## S3 method for class 'soap.film'
eval_smooth(
  smooth,
  model,
  n = 100,
  n_2d = NULL,
  n_3d = NULL,
  n_4d = NULL,
  data = NULL,
  unconditional = FALSE,
  frequentist = FALSE,
  overall_uncertainty = TRUE,
  clip = TRUE,
  envir = NULL,
  back_transform = FALSE,
  ...
)

## S3 method for class 'scam_smooth'
eval_smooth(
  smooth,
  model,
  n = 100,
  n_2d = NULL,
  n_3d = NULL,
  n_4d = NULL,
  data = NULL,
```

```
  unconditional = FALSE,
  frequentist = FALSE,
  overall_uncertainty = TRUE,
  dist = NULL,
  envir = NULL,
  back_transform = FALSE,
  ...
)

## S3 method for class 'fs.interaction'
eval_smooth(
  smooth,
  model,
  n = 100,
  n_2d = NULL,
  data = NULL,
  unconditional = FALSE,
  frequentist = FALSE,
  overall_uncertainty = TRUE,
  envir = NULL,
  back_transform = FALSE,
  ...
)

## S3 method for class 'sz.interaction'
eval_smooth(
  smooth,
  model,
  n = 100,
  n_2d = NULL,
  data = NULL,
  unconditional = FALSE,
  frequentist = FALSE,
  overall_uncertainty = TRUE,
  envir = NULL,
  back_transform = FALSE,
  ...
)

## S3 method for class 'random.effect'
eval_smooth(
  smooth,
  model,
  n = 100,
  n_2d = NULL,
  data = NULL,
  unconditional = FALSE,
  frequentist = FALSE,
```

```
  overall_uncertainty = TRUE,
  envir = NULL,
  back_transform = FALSE,
  ...
)

## S3 method for class 'mrf.smooth'
eval_smooth(
  smooth,
  model,
  n = 100,
  n_2d = NULL,
  data = NULL,
  unconditional = FALSE,
  frequentist = FALSE,
  overall_uncertainty = TRUE,
  envir = NULL,
  back_transform = FALSE,
  ...
)

## S3 method for class 't2.smooth'
eval_smooth(
  smooth,
  model,
  n = 100,
  n_2d = NULL,
  n_3d = NULL,
  n_4d = NULL,
  data = NULL,
  unconditional = FALSE,
  frequentist = FALSE,
  overall_uncertainty = TRUE,
  dist = NULL,
  envir = NULL,
  back_transform = FALSE,
  ...
)

## S3 method for class 'tensor.smooth'
eval_smooth(
  smooth,
  model,
  n = 100,
  n_2d = NULL,
  n_3d = NULL,
  n_4d = NULL,
  data = NULL,
```

```
  unconditional = FALSE,
  frequentist = FALSE,
  overall_uncertainty = TRUE,
  dist = NULL,
  envir = NULL,
  back_transform = FALSE,
  ...
)
```

Arguments

smooth	currently an object that inherits from class mgcv.smooth.
...	arguments passed 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 a univariate smooth.
n_2d	numeric; the number of points along each of the first two axes of a smooth surface, including surface panels of higher-dimensional smooths. The default is 50 in plotting and plot-preparation functions. If NULL, use n instead. Ignored when evaluation data are supplied. Factor levels are retained, and curves with only one continuous covariate use n.
n_3d, n_4d	numeric; the number of points along the third axis of a 3D smooth (n_3d, default 16), or each axis after the first two for smooths of dimension four or higher (n_4d, default 4). If NULL, use n for those axes. For factor conditioning dimensions, retain at most that many levels, taking the first in fitted order. The first two surface axes use n_2d. Supplied data are unchanged.
data	an optional data frame of values to evaluate smooth at.
unconditional	logical; if TRUE (and only if frequentist == FALSE) then the bayesian smoothing parameter uncertainty-corrected covariance matrix is returned, if available. Whether it is available depends on which smoothness selection method was used to fit the model.
frequentist	logical; if FALSE, the default, the bayesian covariance matrix is returned, otherwise the frequentist covariance matrix.
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.
envir	an optional environment supplying functions and constants used in model expressions. The available model formula environment is used when NULL. Covariate observations should be supplied in data.

back_transform logical; generate automatic grids on the original covariate scale? The default, FALSE, preserves evenly spaced model-scale grids (e.g. in log(x)). With TRUE, recover raw fitting covariates and generate grids on their original scale, then evaluate the model expressions forwards. Each axis must depend on one distinct original model covariate. Functions such as foo(x) require no inverse. Fitted prediction parameters are retained where provided by the model, including for scale(x). When data are supplied, use those rows without generating a grid; TRUE requires raw coordinates for each axis. Returned model-coordinate columns keep their usual names and raw coordinates are retained in auxiliary .gratia_raw_* columns. Draw these estimates with draw(object, back_transform = TRUE) to display the raw coordinates. Estimates containing only transformed coordinates give an informative error if drawn on the original scale. This option does not change effect values or their intervals at the chosen evaluation points.

clip logical; should evaluation points be clipped to the boundary of a soap film smooth? The default is FALSE, which will return NA for any point that is deemed to lie outside the boundary of the soap film.

---

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, ...)

## S3 method for class 'gamm4'
evaluate_parametric_term(object, ...)
```

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.

See Also

gamm4_support

`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"`.

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 each level once as a factor, preserving its levels and ordered status. all_levels() is an alias of evenly() with a name intended for factors.

Usage

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

all_levels(x, n = 100, by = NULL, lower = NULL, upper = NULL)

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

Arguments

x	numeric or factor; vector over which evenly-spaced values or factor levels 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 by.
lower	numeric; the lower bound of the interval.
upper	numeric; the upper bound of the interval.

Value

A numeric vector of length n, or a factor containing each level of x once, with the same levels and ordered status as x.

See Also

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

Examples

```
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, ...)

## S3 method for class 'gamm4'
factor_combos(object, ...)
```

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.

See Also

gamm4_support

`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 'gamm4'
family(object, ...)
```

Arguments

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

---

family_diagnostics *Extend diagnostics for an mgcv-compatible family*

Description

A family can supply a `gratia_diagnostics` function to describe its response distribution without adding family-name checks to gratia. The function is called with named arguments `mu`, `wt`, `scale`, `y` and `eta`. These contain response parameters, weights, fitted scale, the fitting response and linear predictors, respectively. Diagnostics use fitted parameters and sampling weights; `log_lik()` uses draw-specific parameters and likelihood weights. Missing fitting rows have been removed; grouped families receive only their own rows. `eta` can be `NULL` when a distribution is constructed without a fitted model.

Details

Return a named list. Omitted fields retain gratia's existing family defaults; an explicit `NULL` removes an optional function or moment. Existing `$rd`, `$cdf` and `$qf` functions are preserved and supplied with their usual mgcv arguments. Replacement functions in the returned list instead capture fitted parameters and use the following interface:

- `mean`, `variance`: numeric response moments, one per observation. A matrix is allowed for multiple response columns. Use the full response expectation, rather than a location parameter or a component mean.
- `cdf`, `cdf_left`: functions `(q, log_p = FALSE, lower_tail = TRUE)` for `P(Y <= q)` and `P(Y < q)`, respectively. They must support both tails and logarithms. For an upper tail, `cdf_left` returns `P(Y >= q)`.
- `qf`: a function `(p, log_p = FALSE, lower_tail = TRUE)` for quantiles.

- `rd`: a function with no arguments that draws a complete response vector (or response matrix), retaining any dependence between observations.
- `log_lik`: a function `(y)` returning one normalised log density or log probability mass per observation, including fitting weights. A response vector contributes one value; a multivariate response contributes one joint value per row. `log_lik()` calls the factory at each coefficient draw with likelihood weights, which can differ from diagnostic sampling weights. Parameters represented by `mu` and `eta` vary with that draw; the hook must exclude smoothing penalties and priors. Finite positive log densities and `-Inf` for impossible observations are allowed; missing values and `Inf` are invalid.
- `bounds`: a function `(y)` returning a list with numeric `lower` and `upper` response bounds. Censoring intervals use their endpoints. Exact count observations can use `y - 1` and `y`. Alternatively, supply `cdf_left`: gratia then randomises between `P(Y < y)` and `P(Y <= y)` automatically. A bounds list may also supply `cdf_lower` and `cdf_upper`, functions `(log_p = FALSE, lower_tail = TRUE)` that evaluate its endpoint probabilities, and `left_limit`, a logical vector marking rows to randomise at equal bounds.
- `continuous`: set to `TRUE` to declare that responses have no probability masses. An unfamiliar grouped-family component must supply `bounds`, `cdf_left`, or this declaration before using PIT/quantile residuals.
- `diagnostic_residuals`: a function `(y, type)` for observed and reference deviance, response and Pearson residuals. Otherwise the family's existing residual method is used.
- `observe`: a function `(y)` applying an observation process, such as known censoring thresholds, to generated responses before traditional QQ/worm reference residuals are calculated. Return the response encoding expected by the family's residual method. This function is not applied to fitting observations or to the public response and posterior samplers.
- `reference_available`, `reference_reason`: a single logical value and an explanation. Set availability to `FALSE` when traditional distribution-based references cannot be generated, for example when censoring thresholds are unknown. Analytic PIT/quantile plots remain available.
- `mean_available`, `variance_available`, `moment_reason`: single logical values specifying whether the population moments exist, and an explanation used when a requested residual requires an undefined moment.
- `independent`, `joint_available`, `dependence_reason`: single logical values describing independence and availability of joint response draws, and an explanation of dependence. `joint_available` requires an `rd` function.
- `approximation`: an optional explanation shown in diagnostic subtitles.

Functions must return one value per response in fitting order. Probabilities must be valid and lower bounds must not exceed upper bounds. gratia validates supplied values and reports errors instead of silently using another method. The factory is called again for each prepared distribution; avoid changing the family or drawing random numbers in the factory itself.

Adding this function does not extend a family's supported fitting methods. Posterior predictive sampling still uses the fitting package's parameter and response sampling methods. The likelihood hook also operates at supplied coefficient draws.

See Also

diagnostic_support, `quantile_residuals()`, `appraise()`

Examples

```
load_mgcv()
f <- poisson()
f$gratia_diagnostics <- function(mu, wt, scale, y, eta) {
  list(mean = mu, variance = mu,
    cdf_left = function(q, log_p = FALSE, lower_tail = TRUE) {
      ppois(ceiling(q) - 1, lambda = mu,
        log.p = log_p, lower.tail = lower_tail)
    })
}
dat <- data.frame(x = seq(0, 1, length.out = 40))
dat$y <- rep(c(0, 1, 2, 4), 10)
f$rd <- function(mu, wt, scale) rpois(length(mu), mu)
f$qf <- function(p, mu, wt, scale) qpois(p, mu)
f$cdf <- function(q, mu, wt, scale) ppois(q, mu)
fit <- gam(y ~ x, data = dat, family = f, method = "REML")
qq_plot(fit, type = "quantile", seed = 1)
```

---

family_name *Name of family used to fit model*

Description

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

Usage

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

Arguments

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

Value

A character vector containing the family name.

`family_parameters` *Describe a model family's distribution parameters*

Description

Inspect the meaning and order of distribution parameters independently of the numerical representation used by the fitting package.

Usage

```
family_parameters(object, ...)
```

Arguments

`object`	A fitted mgcv model, an existing GAM wrapper, or an mgcv family object. Other fitting packages require their own specification.
`...`	Arguments reserved for methods.

Details

`fitted_values()` and `fitted_samples()` default to the parameters marked `returned`. On the response scale these are natural distribution parameters, which can differ from the fitting package's native response predictions. For example, Gaussian LSS output contains standard deviation, whereas mgcv predicts reciprocal standard deviation. GEV location is not its expectation. `link` describes the native representation: for example, `ziplss()` uses an identity link for log rate, while natural rate is the exponential of that predictor. The sinh-arcsinh skewness and tailweight parameters control distribution shape; they are not the response's standardized skewness and kurtosis.

`link()` and `inv_link()` retain the fitting package's native conventions and their existing aliases. Use the definitions here to interpret them.

Value

A tibble with `parameter`, `symbol`, `definition`, `native`, `predictor`, `source`, `link`, `component`, `returned`, and `uncertainty`. Predictor indices follow the fitting package's equation order. Global parameters have no predictor and are omitted from fitted parameter output; they remain fixed during coefficient sampling. Derived rows identify constraints such as a multinomial reference category. The `package` and `family` attributes identify the parameterisation provider. The `parameterisation` attribute describes the mathematical distribution. `expectation_condition` records conditions for a finite expected response.

See Also

`fitted_values()`, `fitted_samples()`

Examples

```
family_parameters(mgcv::gevlss())
family_parameters(mgcv::gaulss())
family_parameters(mgcv::ziplss())
```

`family_type` *Extracts the type of family in a consistent way*

Description

Extracts the type of family in a consistent way

Usage

```
family_type(object, ...)

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

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

Arguments

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

`fitted_moments` *Estimate moments of the conditional response distribution*

Description

Calculate population moments at observed or supplied covariate values, with uncertainty propagated jointly through the model's distribution parameters.

Usage

```
fitted_moments(object, ...)

## S3 method for class 'gam'
fitted_moments(
  object,
  data = NULL,
  moments = c("mean", "variance", "cv"),
  uncertainty = c("delta", "simulation", "none"),
```

```
  ci_level = 0.95,
  unconditional = FALSE,
  nuisance = c("estimated", "fixed"),
  n_draws = 10000L,
  seed = NULL,
  scores = NULL,
  method = c("gaussian", "mh", "inla", "user"),
  draws = NULL,
  envir = NULL,
  family_index = NULL,
  ...
)
```

Arguments

object	A fitted mgcv::gam() or mgcv::bam() model.
...	Prediction arguments passed to mgcv::predict.gam(), including exclude and na.action. type, newdata, se.fit, and terms are controlled internally and cannot be supplied.
data	Optional prediction data. With NULL, evaluate fitting rows, retaining fitting offsets and the model's missing-value convention.
moments	Character vector selecting "mean", "variance", "sd", "cv", "skewness", "kurtosis", or "excess_kurtosis". "all" requests all seven. CV is SD divided by a strictly positive mean; kurtosis is the fourth central moment divided by squared variance.
uncertainty	One of "delta", "simulation", or "none".
ci_level	Coverage of pointwise uncertainty intervals, between 0 and 1.
unconditional	Include smoothing parameter uncertainty when available.
nuisance	One of "estimated" or "fixed". The default propagates uncertainty in estimated family parameters outside the coefficient vector, such as Gamma dispersion, negative-binomial shape, or Tweedie power. "fixed" explicitly conditions on their fitted values.
n_draws	Number of joint parameter draws for simulation, at least two. User-supplied coefficient draws determine their own count when omitted.
seed	Optional integer random seed. An explicit seed preserves the caller's random-number state; NULL uses and advances it when generating random draws. Supplied draws do not consume random numbers.
scores	Optional numeric category scores, in category order, for ordinal or multinomial models. Otherwise return moments of each category's zero/one membership indicator. Binomial moments describe a Bernoulli response, independently of fitting trial counts or likelihood weights.
method	Coefficient sampler for simulation; see post_draws(). Joint sampling of estimated global family parameters uses "gaussian".
draws	Optional coefficient matrix for method = "user", in fitted coefficient order. Requires nuisance = "fixed" if global parameters were estimated outside the coefficient vector.

envir	Environment used to recover model covariates.
family_index	Optional integer component-family positions for grouped predictions, one per supplied row. Membership is otherwise recovered as in fitted_samples(). It must agree with membership recoverable from data.

Details

Estimates describe one response at each covariate combination, with unit observation weight. Fitting weights are not transferred to new responses. For ML/REML fits with an estimated dispersion, all uncertainty modes use the likelihood estimate stored in reml.scale. The conditional coefficient covariance is rescaled to this value. This can differ from mgcv's separate Pearson/Fletcher estimate in sig2; the saved outer Hessian describes the likelihood estimate, so combining it with sig2 would mix estimators. Response variance and uncertainty in its estimate are different quantities: .fitted contains the variance when .parameter is "variance", and .se describes uncertainty in that estimate. These are not moments of a posterior predictive mixture or prediction intervals for future observations.

Standard Gaussian, Poisson, binomial, Gamma and inverse-Gaussian families, negative binomial, beta, scaled t, Tweedie, all seven location/scale/shape families, ordinal, multinomial and multivariate-normal models are supported. Censored normal, logistic and Poisson families return moments of the latent uncensored response. Beta moments describe the beta likelihood, without the epsilon clipping used by some response generators. bcg() supports the original lognormal response only when its Box-Cox parameter is fixed at zero: other powers require a distribution outside the inverse transformation's domain that the fitted family does not specify. Quasi and Cox families are unsupported. Grouped (gfam) predictions use each row's component family and the full joint model covariance. Grouped ordinal components require explicit category scores; grouped category-indicator output is unsupported.

The delta method differentiates the complete parameter-to-moment transformation and uses its full joint covariance. Gaussian, Poisson, Gamma, inverse-Gaussian and Tweedie derivatives (including their models with several predictors) are analytic; other families use central numerical differences. Positive variance, SD, CV and kurtosis intervals are calculated on a log scale; .se always remains on the reported quantity's scale. Other delta intervals use the identity scale. Simulation uses joint parameter draws and equal-tailed quantiles. Both methods retain the moment evaluated at fitted parameters as .fitted.

With estimated global family parameters, a first-order joint Gaussian approximation combines the conditional Bayesian coefficient covariance with second derivatives of the ML/REML fitting criterion and derivatives of the fitted coefficients with respect to the family and smoothing parameters. Without unconditional, outer curvature is conditioned on smoothing parameters. With unconditional, the full saved outer Hessian is inverted, including its smoothing uncertainty. The curvature and coefficient sensitivities are evaluated at the fitted parameters, with mgcv's mapping for linked or fixed smoothing parameters. A positive-definite Hessian is required. This does not also add the corrected coefficient covariance, which would count that uncertainty twice. The fitted object must store a matching Hessian and sensitivities; unsupported fitting methods, including fast REML BAM fits, give an error. This is a local approximation about the fitted parameters, not an exact posterior sampler for dispersion or family shape. With fixed global parameters, the existing coefficient covariance or post_draws() machinery is used. Delta intervals and Gaussian simulation with unconditional = TRUE and fixed estimated global parameters use the first-order smoothing correction conditional on those fixed parameters: native corrected covariance can also include their

uncertainty. This requires the same saved outer Hessian and sensitivities when smoothing parameters are present. Legacy `negbin()` fits outside ML/REML require explicit `nuisance = "fixed"`; use `nb()` to estimate and propagate shape uncertainty. The covariance calculation uses a first-order expansion of the coefficient means and does not include mgcv's additional correction for changes in the conditional coefficient covariance. The `moment_info` attribute records which approximation was used.

Nonexistent moments are retained as `NA` with a status and one warning per call. In simulation, `.p_defined` records the fraction of draws with a finite value. Intervals and standard errors are withheld if any draw is undefined; draws are never silently dropped. CV requires a positive mean, and skewness and kurtosis require positive variance. GEV moments of order r require shape $< 1/r$; t moments require degrees of freedom greater than their order. Status values distinguish `"moment_does_not_exist"`, `"nonpositive_mean"`, `"zero_variance"`, `"undefined_in_draws"`, and `"missing_predictors"`. `"numerical_failure"` identifies a value that could not be calculated with finite precision or invalid distribution parameters; `"derivative_unavailable"` identifies a finite estimate whose local derivative could not be calculated.

Value

A tibble of class `"fitted_moments"`, containing `.row`, covariates, `.parameter`, `.fitted`, `.se`, `.lower_ci`, `.upper_ci` and `.status`. Category indicators include `.category`; multivariate responses include `.y`; grouped responses include `.family`. Simulation also includes `.p_defined`. All estimates, standard errors and interval bounds are on the requested quantity's scale.

See Also

`conditional_moments()`, `fitted_values()`, `fitted_samples()`

Examples

```
load_mgcv()
m <- gam(mpg ~ s(wt, k = 5), data = mtcars, method = "REML")
fitted_moments(m, data = mtcars[1:3, ], moments = c("mean", "variance"))
fitted_moments(m, data = mtcars[1:3, ], moments = "variance",
  uncertainty = "simulation", n_draws = 500, seed = 42)
```

---

fitted_samples *Draw fitted values from the posterior distribution*

Description

Distribution parameters or expected responses drawn from the posterior distribution of a fitted model.

Usage

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

fitted_samples(model, ...)

## S3 method for class 'gam'
fitted_samples(
  model,
  ...,
  data = NULL,
  n_draws = NULL,
  chains = NULL,
  seed = NULL,
  method = c("gaussian", "mh", "inla", "user"),
  n_cores = 1L,
  n_threads = 1L,
  burnin = 1000,
  thin = 1L,
  t_df = 40,
  rw_scale = 0.25,
  frequentist = FALSE,
  unconditional = FALSE,
  draws = NULL,
  diagnostics = FALSE,
  mh_backend = c("mgcv", "gratia", "family"),
  mvn_method = c("mvnfast", "mgcv"),
  block_size = 10000L,
  dense_limit = 5e+06,
  scale = c("response", "linear_predictor"),
  family_index = NULL,
  quantity = c("parameters", "expectation"),
  marginalise = NULL,
  integration = c("quadrature", "monte_carlo"),
  n_quad = 40L,
  n_integrate = 5000L
)

## S3 method for class 'scam'
fitted_samples(
  model,
  ...,
  data = NULL,
  n_draws = NULL,
  chains = NULL,
  seed = NULL,
  method = c("gaussian", "mh", "inla", "user"),
  n_cores = 1L,
```

```
  n_threads = 1L,
  burnin = 1000,
  thin = 1L,
  t_df = 40,
  rw_scale = 0.25,
  frequentist = FALSE,
  unconditional = FALSE,
  draws = NULL,
  diagnostics = FALSE,
  mh_backend = c("mgcv", "gratia", "family"),
  mvn_method = c("mvnfast", "mgcv"),
  block_size = 10000L,
  dense_limit = 5e+06,
  scale = c("response", "linear_predictor"),
  quantity = c("parameters", "expectation")
)
```

Arguments

model	a fitted model of the supported types; see diagnostic_support.
...	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.
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.
n_draws	positive integer; total retained draws across all chains. Required when generating draws. With method = "user", the supplied matrix determines the count; an explicit count must agree with its rows.
chains	positive integer or NULL; number of chains. NULL uses one chain, or the chain mapping in supplied draws. n_draws must be divisible by chains. Burn-in is applied separately to each MH chain.
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 as a coefficient matrix.
n_cores	positive integer; maximum concurrent MH chains using mirai, capped at chains. Existing workers are reused; otherwise temporary workers are created and cleaned up. Only used by method = "mh".
n_threads	positive integer; threads within Gaussian sampling using mvnfast. Defaults to one. This is separate from concurrent MH chains.

burnin	numeric; number of samples to discard as the burnin draws. Only used with method = "mh".
thin	positive integer; retain every thin iterations after burn-in.
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".
frequentist	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 frequentist == 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".
diagnostics	logical; calculate coefficient R-hat, effective sample sizes, and Monte Carlo standard errors using the optional posterior package.
mh_backend	character; "mgcv" preserves mgcv::gam.mh(). "gratia" uses compiled common-family likelihoods. "family" uses a compiled MH loop with callbacks to the fitted family object. Both support independent-observation bam fits. Smoothing parameters and parameters outside the fitted coefficient vector remain fixed. LSS scale and shape coefficients are sampled jointly with location. The family backend caches a full design matrix, smoothing penalty, and proposal factorisation once for all chains.
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.
block_size	positive integer; maximum observation rows per block in the gratia MH evaluator. Large designs are evaluated without storing a full matrix.
dense_limit	numeric; maximum number of design-matrix elements to cache for the gratia MH backend. Use zero to force blockwise evaluation.
scale	character; "response" returns natural distribution parameters; "link", "linear_predictor" and "linear predictor" return their predictors. Expectations require "response".
family_index	numeric; optional family-list positions for grouped family (mgcv::gfam()) response predictions, with one integer per requested row. By default membership is recovered from the response/index expression in data, or the fitted membership when data is omitted. This is useful when predicting without responses or with a complex response expression. An explicit index must agree with membership recoverable from data.
quantity	character; "parameters" (the default) returns all distribution parameters represented by predictors, including intercept-only predictors. "expectation" returns the expected response, using every parameter on which it depends, on the response scale only.
marginalise	NULL retains the usual predictions. TRUE selects all supported re and fs terms; character vectors select exact smooth labels. Fitted predictions integrate response means over new zero-mean Gaussian effects. Response samplers draw

new effects before sampling responses. Smoothing parameters and dispersion remain fixed. Only ordinary `gam()` and `bam()` fits with a scalar response mean are supported.

`integration` Integration method when `marginalise` is supplied: `"quadrature"` (the default) or `"monte_carlo"`. Both integrate over the Gaussian distribution of the selected effects at each row.

`n_quad` Initial Gauss-Hermite point count, between 2 and 1000. Successive doubled orders, up to four times this count, and an interlaced rule with one fewer point must agree within `1e-8 + 1e-6 * abs(value)` for means and derivatives. Where the random-effect variance exceeds one, an independent adaptive rule also checks agreement. Failure to agree reports an error. Identity, log and probit links use their closed-form Gaussian integrals; their metadata records zero quadrature points and zero integration error.

`n_integrate` Number of Monte Carlo integration draws, at least two. These are shared across rows and posterior coefficient draws. This count is separate from `n_draws`. Ignored for quadrature.

Details

With `marginalise`, fitted samples contain posterior draws of the integrated mean, rather than the variation among new groups. Selected fitted effects are removed before integration. `exclude` cannot name a selected term and `terms` is unsupported. Only response-scale means and links defined over the whole Gaussian predictor distribution are supported. Grouping columns can be omitted if they occur exclusively in the selected terms. A factor also used as a selected smooth's `by` variable must still be supplied with a fitted level, because it selects the relevant curve distribution. Fits using `min.sp` are unsupported; see `random_effect_samples()`. The `integration_info` attribute records fixed variance estimates, the integration method, point or draw counts, and numerical error estimates. For Monte Carlo these errors are standard errors of integration, separate from uncertainty in model predictions. `unconditional` can adjust the coefficient covariance; it does not draw random-effect variance parameters.

Parameter names and definitions are available through `family_parameters()`. Gaussian LSS returns `mean` and `sd`, GEV returns `location`, `scale` and `shape`, and sinh-arcsinh returns `location`, `scale`, `skewness` and `tailweight`. These shape parameters are not response moments. Natural parameters can differ from native mgcv predictions and `inv_link()` values. Every parameter in one draw uses the same joint coefficient draw. Global parameters outside the coefficient vector remain fixed and are described in the metadata, rather than returned as fitted parameters.

Use `quantity = "expectation"` for expected responses labelled `"expectation"`. This includes Gumbel, GEV and sinh-arcsinh mean adjustments and the zero-truncated Poisson adjustment in `ziplss()`. A GEV expectation does not exist at shape >= 1; parameter and response draws remain available. Categorical expectations are category probabilities, not numeric category scores. `.category` identifies categories, and `.y` identifies multivariate responses. Multinomial link output includes a zero reference predictor. Ordinal parameter output is the latent location predictor; its thresholds are global parameters.

The `prediction_info` attribute records the quantity, scale and resolved parameter descriptions. Unlike `posterior_samples()`, these draws add no response noise. `predicted_samples()` instead simulates observations with coefficients held at their fitted values.

For grouped families, coefficient draws preserve the joint model covariance; each row is then transformed using its own fitted component family. Explicit prediction data must identify each row's family: no training-order fallback is used. A two-column response matrix supplies its second column; a numeric column named for the response supplies indices directly. For a response such as `cbind(y, index)`, the `index` expression can be supplied without `y`. Missing indices behave like missing required predictors. Link-scale draws do not require family membership.

Grouped parameter output includes the latent location for ordinal components. Grouped expectations include scalar fitted means and ordinal category probabilities, with `.category` identifying the ordinal category and `NA` for scalar components. Distributional nuisance parameters are held at their fitted values. This does not add support for general/location-scale families that `gfam()` itself rejects, nor repair upstream fitting failures (including fixed-theta negative-binomial components and some `ziP()` fits in some mgcv versions).

Observation rows use the same missing-value convention as `posterior_samples()`: default results restore `na.exclude` observations in every draw, while explicit `data` are evaluated in their own row order.

Value

A tibble (data frame) in long format, with columns:

- `.row`: row of the evaluation data;
- `.draw`: globally unique draw identifier;
- `.parameter`: the distribution parameter name, or `"expectation"`;
- `.fitted`: the parameter, predictor or expected response requested. Multiple-chain output also includes `.chain` and `.iteration`. Attached coefficient diagnostics are available through `sampling_diagnostics()`.

Note

With `data = NULL`, fitting offsets are retained, including offsets supplied through the fitting function's `offset` argument. Explicit prediction data follow `mgcv::predict.gam()`: offsets in formulas are evaluated from the prediction data, while offsets supplied through the fitting function's `offset` argument are not transferred to new observations. Include offsets in the formula when predicting at new data; the required offset variables must then be present in `data`.

Author(s)

Gavin L. Simpson

References

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

See Also

gamm4_support

Examples

```
load_mgcv()

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

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

fs


# can generate own set of draws and use them
drws <- generate_draws(m1, n_draws = 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

Usage

```
fitted_values(object, ...)

## S3 method for class 'gam'
fitted_values(
  object,
  data = NULL,
  scale = c("response", "link", "linear predictor"),
  ci_level = 0.95,
  envir = NULL,
  ...,
  interval = c("confidence", "simultaneous"),
  n_sim = 10000,
  n_cores = 1,
  seed = NULL,
  unconditional = FALSE,
  interval_method = c("auto", "gaussian", "bootstrap"),
  quantity = c("parameters", "expectation"),
  family_index = NULL,
  marginalise = NULL,
  integration = c("quadrature", "monte_carlo"),
```

```
  n_quad = 40L,
  n_integrate = 5000L
)

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

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

## S3 method for class 'gamm4'
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. "linear_predictor" is also accepted. Expectations require "response"; parameter predictions use natural values or their predictors.
ci_level	numeric; a value between 0 and 1 indicating the coverage of the credible interval.
envir	an optional environment supplying functions and constants used in model expressions. The available model formula environment is used when NULL. Covariate observations should be supplied in data.
interval	character; "confidence" (the default) gives pointwise intervals. "simultaneous" gives simultaneous intervals for predictions at the supplied covariate combinations.
n_sim	positive integer; number of coefficient draws used for simultaneous intervals. Ignored for pointwise intervals.
n_cores	positive integer; number of cores used by mvnfast::rmvn() for simultaneous intervals. Parallel execution requires OpenMP support.
seed	integer or NULL; optional random seed for simultaneous intervals. An explicit seed preserves the caller's random number state. With NULL, the current random number state is used and advanced.
unconditional	logical; include smoothing parameter uncertainty in the Bayesian covariance matrix, if available. Otherwise a warning is issued and the uncorrected covariance is used.
interval_method	character; "auto" uses stored scasm bootstrap draws for pointwise intervals when available and Gaussian intervals otherwise. "gaussian" requests covariance-based limits and "bootstrap" requires stored draws. Simultaneous bootstrap bands are unsupported. Shape-enforced intervals are available for smooth components through smooth_estimates().

quantity — character; "parameters" (the default) returns all distribution parameters represented by predictors, including intercept-only predictors. "expectation" returns the expected response, using every parameter on which it depends, on the response scale only.

family_index — numeric; optional family-list positions for grouped family (mgcv::gfam()) response predictions, with one integer per requested row. By default membership is recovered from the response/index expression in data, or the fitted membership when data is omitted. This is useful when predicting without responses or with a complex response expression. An explicit index must agree with membership recoverable from data.

marginalise — NULL retains the usual predictions. TRUE selects all supported re and fs terms; character vectors select exact smooth labels. Fitted predictions integrate response means over new zero-mean Gaussian effects. Response samplers draw new effects before sampling responses. Smoothing parameters and dispersion remain fixed. Only ordinary gam() and bam() fits with a scalar response mean are supported.

integration — Integration method when marginalise is supplied: "quadrature" (the default) or "monte_carlo". Both integrate over the Gaussian distribution of the selected effects at each row.

n_quad — Initial Gauss-Hermite point count, between 2 and 1000. Successive doubled orders, up to four times this count, and an interlaced rule with one fewer point must agree within 1e-8 + 1e-6 * abs(value) for means and derivatives. Where the random-effect variance exceeds one, an independent adaptive rule also checks agreement. Failure to agree reports an error. Identity, log and probit links use their closed-form Gaussian integrals; their metadata records zero quadrature points and zero integration error.

n_integrate — Number of Monte Carlo integration draws, at least two. These are shared across rows and posterior coefficient draws. This count is separate from n_draws. Ignored for quadrature.

Details

With marginalise, predictions are response means averaged over the selected random-effect distributions, using the methods described in fitted_samples(). The point estimate uses fitted coefficients; pointwise intervals use the derivative of the integrated mean and the coefficient covariance. .se is on the response scale. These normal intervals are not clipped to the response range. Simultaneous and bootstrap marginal intervals are not yet supported. The integration_info attribute records numerical accuracy and fixed variance estimates.

quantity = "parameters" returns all distribution parameters represented by predictors, including intercept-only predictors. See family_parameters() for their definitions. These are natural distribution parameters, which can differ from mgcv's native response predictions: Gaussian LSS returns standard deviation, GEV returns location rather than expectation, and sinh-arcsinh returns shape parameters rather than response moments.

quantity = "expectation" returns the expected response on the response scale, labelled "expectation". Categorical expectations are probabilities of category membership; multivariate models return a mean per response. .category and .y identify the category or response separately from .parameter. Global parameters are described but held fixed when calculating coefficient uncertainty. Coupled

expectations use the full coefficient covariance and a delta-method Jacobian. Their intervals are pointwise. Ordinal and multinomial probability intervals use a normal approximation to log odds, transformed back to probabilities. Log odds and their uncertainty are computed before rounding probabilities, so valid probabilities rounded to zero or one retain defined intervals.

The `prediction_info` attribute records the quantity, scale, standard-error scale and parameter descriptions. Parameter standard errors retain their link-scale meaning; category-probability and expectation standard errors are on the response scale. Unsupported simultaneous or bootstrap transformations are rejected rather than combining marginal parameter intervals.

Native scasm fits with stored bootstrap draws use mgcv's pointwise limits by default, including its `svar.only` correction for smoothing bias. `interval_method = "gaussian"` restores covariance-based limits. `.se` retains its existing covariance-based meaning. No replicates are generated. Bootstrap limits use existing scalar and location-scale parameter transformations. Unsupported nonlinear family outputs give an error for an explicit bootstrap request; automatic selection warns and uses Gaussian limits. Simultaneous bootstrap bands are not supported. Automatic simultaneous intervals warn when stored bootstrap draws are left unused. The `interval_info` attribute records the method and correction.

Formula offsets are included in predictions. Offsets supplied only through the fitting function's `offset` argument are omitted, including when `data = NULL`: this function evaluates explicit prediction rows internally. In contrast, `fitted_samples()` retains fitting-argument offsets when its `data` argument is `NULL`. Put offsets in the formula to evaluate them at new observations.

With `data = NULL`, results follow the model's `na.action`: `na.exclude` restores excluded observations as `NA` fitted values, standard errors and interval bounds; `na.omit` returns only retained observations. Covariates unavailable in the stored model frame are restored as typed `NA` values. `.row` indexes the fitting data after any `subset` was applied.

Supplying `data` requests predictions at those rows, independently of training exclusions. Missing responses do not prevent prediction. Missing required predictors produce `NA` results by default. An explicit `na.action` passed through `...` is honoured; with `na.omit`, `.row` retains the positions in the supplied data, rather than numbering the remaining rows consecutively.

For supplied prediction data, an informative message identifies combinations of categorical predictor levels absent from the retained fitting observations. Predictions are unchanged: an unobserved combination is not necessarily non-estimable, particularly in additive models. Numeric-by-factor range extrapolation is not checked. Missing predictors and genuinely new factor levels retain their usual prediction behavior.

Simultaneous intervals jointly cover the model's underlying expected responses, or linear predictors on the link scale, at all valid covariate combinations evaluated in this call, with approximate posterior probability `ci_level`. The fitted values are estimates of these quantities. The combinations may include factor levels and need not form an ordered sequence, curve, or grid. This interpretation also applies to two predictions. With `data = NULL`, the simultaneous set comprises the retained fitting covariate combinations. Missing predictions do not enter this set. Coverage is not asserted outside this set or between its points. Separate calls define separate simultaneous sets. These are uncertainty intervals for model predictions, not prediction intervals for future observations.

Simultaneous intervals use joint Gaussian coefficient draws and a critical value based on the maximum absolute standardized deviation over the supplied rows. Their limits are computed on the link scale and transformed if `scale = "response"`; `.se` remains on the link scale. The simulation covariance may be positive semidefinite. Supported families are Gaussian, Poisson, binomial, Gamma, inverse Gaussian, quasi families, negative binomial, Tweedie, beta regression, and scaled

t, fitted by gam() or bam() (including the gam component of a gamm() fit), and by scam(). For SCAM models, the Gaussian approximation uses transformed coefficients conditional on the fitted smoothing parameters. Other families support pointwise intervals only.

Response-scale simultaneous intervals support the identity, log, logit, probit, cloglog, cauchit, square-root, inverse, and inverse-squared links. Intervals crossing an inverse-link domain boundary are rejected; use scale = "link" for these intervals or for other links.

terms and exclude in ... retain mgcv::predict.gam() semantics and apply to both prediction and interval calculation. If terms are selected or excluded, intervals concern that selected quantity, not necessarily the full expected response. Use the link scale for sums of smooth contributions, and consider the intercept and offset treatment when selecting terms.

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. The result also includes:

- .parameter: parameter name, or "expectation",
- .fitted: the requested quantity on the specified scale,
- .se: standard error; see prediction_info for its scale,
- .lower_ci, .upper_ci: 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 quantity = "expectation", 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 parameter output, the .se component is on the *link* (*linear predictor*) scale. Multinomial probabilities on the response scale and all expected responses have response-scale standard errors.

See Also

gamm4_support

Examples

```
load_mgcv()

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

# Simultaneous intervals for two arbitrary covariate combinations
fitted_values(m, data = sim_df[c(1, 20), ], interval = "simultaneous",
  n_sim = 1000, seed = 42)
```

```
# A selected sum of smooth contributions on the link scale
fitted_values(m, data = sim_df[c(1, 20), ], scale = "link",
  terms = c("s(x2)", "s(x3)"), interval = "simultaneous",
  n_sim = 1000, seed = 42)
```

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

Description

Removes evaluated offset columns while retaining their raw input variables. Raw inputs are recovered from fitting data if needed; transformed offset values are never renamed as raw values. A scalar `offset_val` overrides the raw input only for a single offset with one unambiguous input variable.

Usage

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

Arguments

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

Value

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

Author(s)

Gavin L. Simpson

Examples

```
load_mgcv()
df <- 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))
```

`fixed_effects` *Extract fixed effects estimates from a fitted model*

Description

Extract fixed effects estimates from a fitted model

Usage

```
fixed_effects(object, ...)

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

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

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

## S3 method for class 'lm'
fixed_effects(object, ...)

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

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

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

Arguments

`object`	a fitted model. Supported classes are models fitted by `mgcv::gam()`, `mgcv::bam()`, `mgcv::gamm()`, `mgcv::gam()`, `gamm4::gamm4()`, `stats::glm()`, and `stats::lm()`.
`...`	arguments passed to other methods

Examples

```
load_mgcv()

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

```
    s(Days, Subject, bs = "re"),
    data = sleepstudy, method = "REML"
  )
  fixed_effects(m)
}
```

---

gamm4_support *Support for models fitted by gamm4*

Description

Models fitted by `gamm4::gamm4()` (version 0.2-7 or later) have class `"gamm4"`. Pass the fitted object directly to gratia methods; historical unclassed lists are not supported.

Details

Smooths, parametric effects, predictions, derivatives, simulations, and posterior draws describe the population GAM component. Terms supplied through `random` are excluded, including their realized effects, predictive variation, and uncertainty. `conditional_values()` and `conditional_differences()` condition on covariate values, not on realized group effects. Group variables occurring only in `random` are not required in prediction data, so new groups receive the same population predictions. Use the `mer` component with lme4 for predictions conditional on group effects and simulations including their variation. Diagnostics and response simulations use the full fitted mixed model by default. With `conditioning = "conditional"`, all fitted smooth and additional random effects are held fixed for response references. `"marginal"` draws new effects from their fitted distribution. `"gam_component"` explicitly selects the extracted GAM calculations used by earlier versions. Marginal GLMM checks average over random effects using lme4's joint response simulations. Residuals then use ranks in a shared reference sample, as described in diagnostic_support. Gaussian identity-link models instead use analytic CDFs of the marginal response distribution.

Posterior response draws also default to the full conditional model. They draw fixed and fitted random effects together, keeping covariance parameters at their fitted values. These Gaussian approximations currently use fitting observations only. To supply user coefficient draws or prediction data to `posterior_samples()`, select `conditioning = "gam_component"` explicitly.

Coefficient draws from `post_draws()`, `fitted_samples()`, `predicted_samples()` and `smooth_samples()` still use the fitted GAM covariance matrix, including its uncertainty due to fitting the mixed model. User draws must have one column per GAM coefficient. Metropolis-Hastings sampling is unsupported: the GAM likelihood does not include the additional random effects. Smoothness uncertainty-corrected covariance matrices and alternative EDF are unavailable from gamm4; requested corrections warn and fall back to the available covariance matrix or EDF. Variance components are extracted from the mixed-model component. Smooth components use the same penalty scale and labels as `gam()` and `bam()`, with a separate row for each penalty of a tensor or factor smooth. Additional random effects retain their mixed-model variances. Confidence intervals use likelihood profiles by default; use `interval_method = "none"` to obtain estimates only. Gaussian profiles use a maximum-likelihood refit even when the reported estimates were fitted by REML. See variance_comp for interval details. With `type = "terms"`, variances describe fitted term contributions, including the additional random effects. Their covariance matrix is returned in the `covariance` attribute. Term intervals use joint posterior coefficient draws, retaining covariance with the additional fitted random effects. Use `n_draws = 0` for term estimates without intervals.

gaussian_draws *Posterior samples using a Gaussian approximation*

Description

Posterior samples using a Gaussian approximation

Usage

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

gaussian_draws(model, ...)

## S3 method for class 'gam'
gaussian_draws(
  model,
  ...,
  n_draws = NULL,
  n_threads = 1L,
  frequentist = FALSE,
  unconditional = FALSE,
  mvn_method = "mvnfast",
  index = NULL,
  seed = NULL,
  chains = NULL,
  diagnostics = FALSE
)

## S3 method for class 'scam'
gaussian_draws(
  model,
  ...,
  n_draws = NULL,
  n_threads = 1L,
  frequentist = FALSE,
  unconditional = FALSE,
  mvn_method = "mvnfast",
  index = NULL,
  parametrized = TRUE,
  seed = NULL,
  chains = NULL,
  diagnostics = FALSE
)
```

Arguments

model	a fitted model from mgcv::gam(), mgcv::bam(), scam::scam(), or gamm4::gamm4(), or a model inheriting from these classes. See gamm4_support for sampling from gamm4 fits.
...	arguments passed to methods.
n_draws	positive integer; total retained draws across all chains. Required when generating draws. With method = "user", the supplied matrix determines the count; an explicit count must agree with its rows.
n_threads	positive integer; threads within Gaussian sampling using mvnfast. Defaults to one. This is separate from concurrent MH chains.
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(). SCAM models do not support this correction; use unconditional = FALSE.
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.
index	numeric; vector of indices of coefficients to use. Can be used to subset the mean vector and covariance matrix extracted from model.
seed	numeric or NULL; seed for the complete sampling operation. An explicit seed restores the caller's random-number state on exit.
chains	positive integer or NULL; number of chains. NULL uses one chain, or the chain mapping in supplied draws. n_draws must be divisible by chains. Burn-in is applied separately to each MH chain.
diagnostics	logical; calculate coefficient R-hat, effective sample sizes, and Monte Carlo standard errors using the optional posterior package.
parametrized	logical; for SCAM fits, transform Gaussian draws from the unrestricted fitting coefficients using the fitted positivity transform. This preserves the smooth's constraints. With FALSE, return Gaussian draws in the unrestricted fitting coordinates. Only for scam::scam() fits.

Details

Sampling controls must be named. Removed arguments produce migration errors; see vignette("sampling-migration"). Explicit controls that the selected method cannot use are rejected.

For SCAM models, Gaussian draws approximate the distribution of the unrestricted fitting coefficients using Vp (or Ve with frequentist = TRUE), then apply scam's fitted exponential or notExp transformation. This is an approximation in fitting coordinates, rather than an exact constrained posterior. Supplied draws must already use the transformed coordinates in coef(model) order. Numerically singular covariance matrices are sampled by eigen decomposition; negative eigenvalues within 100 * .Machine$double.eps * ncol(V) times the largest absolute eigenvalue are treated

as zero. Larger negatives are errors. For weakly identified exponential coefficients, a very large fitting-coordinate variance can yield non-finite transformed draws. Sampling then reports an error because this Gaussian approximation cannot provide usable draws; supply finite transformed draws from a suitable alternative sampler.

For multiple chains, `n_draws` is the total retained count and must be divisible by `chains`. Coefficient matrices carry a chain mapping; use `as_mcmc_draws()` before ordinary matrix operations remove that mapping. Higher-level sampling tables include `.chain` and `.iteration` for multiple chains, with `.draw` unique across chains. Default single-chain table columns are unchanged. Use `sampling_diagnostics()` to extract coefficient diagnostics before response simulation adds further noise.

The gratia MH backend supports Gaussian identity, Poisson log, binomial logit, Gamma log, and negative binomial log fits. It evaluates the fitted bases at original covariates, including for discrete bam fits. GAMM components, AR1 dependence, `paraPen`, `H`, and fits made from pre-fit `G` objects are unsupported.

The family MH backend calls the fitted family's `dev.resids()` and `linkinv()` functions, or `ll()` for families with multiple linear predictors. This retains response encodings and attributes such as censoring indicators and the mappings between coefficients and predictors. It supports compatible standard, extended, and general mgcv families, including multinomial, Gaussian location-scale, and grouped families. Quasi families, Cox proportional hazards models, GAMM components, AR1 dependence, `paraPen`, `H`, and pre-fit `G` objects are unsupported. The fitted coefficient covariance must be nonsingular. It always caches the full design matrix; `block_size` and `dense_limit` apply only to the gratia backend and are rejected for the family backend. The new MH backends reject `bam` fits with nonzero prescribed smooth contributions: mgcv fitting and prediction can use different offsets for those fits. Supplied coefficient draws can still be evaluated with the native prediction conventions.

All MH backends sample the conditional coefficient posterior with fitted smoothing, dispersion, and additional family parameters fixed. Parameters represented in the fitted coefficient vector are sampled, including scale or covariance coefficients in general families. `unconditional` applies to Gaussian draws only.

GAM-class fits with linear inequality constraints, including `mgcv::scasm()` fits, store these as `Ain %*% coefficients >= bin`. Gaussian sampling uses the fitted covariance matrix without enforcing these inequalities and issues a warning. Sampled smooths can therefore violate the fitted shape constraints. Pointwise intervals computed from the same Gaussian covariance approximation also need not preserve those shapes. All MH backends are unavailable for these fits because the samplers do not enforce their inequalities.

If `mgcv::scasm()` was fitted with `bs > 0`, its `bs` component stores coefficient draws from the native constrained bootstrap in columns. These can be supplied explicitly as `draws = t(model$bs)` with `method = "user"`. The result then describes bootstrap variation, rather than Gaussian posterior sampling. Gratia does not automatically select these draws. The caller is responsible for the interpretation and constraint feasibility of user-supplied draws.

See Also

gamm4_support

get_by_smooth *Extract an factor-by smooth by name*

Description

Extract an factor-by smooth by name

Usage

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

Arguments

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

Value

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

get_smooth *Extract an mgcv smooth by name*

Description

Extract an mgcv smooth by name

Usage

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

Arguments

object	a fitted GAM model object.
term	character; the exact 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

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

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

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

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

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

Arguments

object	a fitted GAM model object.
id	numeric; the position of the smooth in the model object.
...	Arguments passed to methods.

get_vcov *Extract the covariance matrix of the model parameters or a subset of these for a particular model term*

Description

Extract the covariance matrix of the model parameters or a subset of these for a particular model term

Usage

```
get_vcov(
  object,
  unconditional = FALSE,
  frequentist = FALSE,
  term = NULL,
```

```
  by_level = NULL
)

## S3 method for class 'scam'
get_vcov(
  object,
  unconditional = FALSE,
  frequentist = FALSE,
  term = NULL,
  by_level = NULL
)
```

Arguments

`object`	an R object from which a covariance matrix shall be extracted.
`unconditional`	logical; if `TRUE` (and only if `frequentist == FALSE`) then the bayesian smoothing parameter uncertainty-corrected covariance matrix is returned, if available. Whether it is available depends on which smoothness selection method was used to fit the model.
`frequentist`	logical; if `FALSE`, the default, the bayesian covariance matrix is returned, otherwise the frequentist covariance matrix.
`term`	character, length 1; return the subset of the covariance matrix for a particular **smooth** term. Only a single term can be selected.
`by_level`	character; currently unsued.

Details

For `mgcv::scasm()` fits with linear inequality constraints, the Gaussian covariance approximation does not enforce the fitted shapes. Pointwise intervals calculated from this matrix can extend outside those constraints. See `post_draws()` for sampling and native-bootstrap details.

Value

a matrix containing the selected covariance matrix.

Author(s)

Gavin L. Simpson

---

gratia_palette *Discrete palettes for gratia plots*

Description

Create a palette function for ggplot2 theme settings. The default preserves gratia's existing nine-colour Okabe-Ito palette and hue fallback.

Usage

```
gratia_palette(name = "default", reverse = FALSE)
```

Arguments

name	Palette name: "default", "tol_bright", "tol_vibrant", "tol_muted", "tol_dark", "tol_banded_12", "tol_banded_16", or "tol_banded_20".
reverse	Logical; reverse the palette order?

Details

Tol's bright and vibrant palettes contain seven colours, muted contains nine, and dark contains eleven. The dark palette favours visibility against white over separation between colours. Banded palettes provide 12, 16 or 20 colours; neighbouring shades become less distinct as the palette grows. These are optional choices and do not change gratia's defaults.

Finite palettes use the published order, returning the first requested colours. They error if their capacity is exceeded; they never recycle or interpolate colours. The default palette uses hue colours above nine levels. Set both colour and fill palettes to keep grouped curves and ribbons paired.

Definitions follow Paul Tol's colour schemes as published in August 2026. The banded schemes are derived from his sequential schemes for use when larger sets of colours are required. See https://sronpersonalpages.nl/~pault/ for the definitions and examples.

Value

A function taking the number of colours and returning a character vector of hexadecimal colours.

Examples

```
pal <- gratia_palette("tol_muted")

pal(5)
ggplot2::theme(
  palette.colour.discrete = pal,
  palette.fill.discrete = pal
)
```

---

gratia_theme_get *Native themes for gratia plots*

Description

Gratia plots use ordinary ggplot2 themes. Component settings are read when the plot is built, so they can also be changed after calling draw().

Usage

```
gratia_theme_get()

gratia_theme_set(theme = NULL)

gratia_theme_update(...)
```

Arguments

theme	A ggplot2 theme, or NULL to clear the package override.
...	Settings passed to ggplot2::theme().

Details

gratia_theme_get() returns the override stored in options("gratia.plot.theme"). It does not include ggplot2's active theme. The override is attached to each plot when it is constructed. Later changes to this option affect new plots. Use withr::with_options() for a temporary override. Unspecified settings inherit ggplot2's active theme as usual.

Component elements accept ggplot2::element_geom(). The main elements are geom.gratia.smooth, geom.gratia.effect, geom.gratia.derivative, geom.gratia.sample, geom.gratia.interval, geom.gratia.partial.residual, geom.gratia.increase, geom.gratia.decrease, geom.gratia.change, geom.gratia.diagnostic.point, geom.gratia.reference, geom.gratia.rug, geom.gratia.contour, geom.gratia.boundary, geom.gratia.observed, geom.gratia.replicated, geom.gratia.chain, geom.gratia.histogram, geom.gratia.rootogram.bar, geom.gratia.rootogram.fitted, and geom.gratia.concurvity.bar. Other elements are geom.gratia.curve (the parent of smooth, effect, derivative, sample and basis), geom.gratia.basis, geom.gratia.surface, geom.gratia.diagnostic.boxplot, geom.gratia.residual.covariate.point, geom.gratia.diagnostic.reference, geom.gratia.chain.reference, geom.gratia.correlation.reference, geom.gratia.histogram.outline, geom.gratia.rootogram.reference, geom.gratia.rootogram.warning, geom.gratia.correlation, and geom.gratia.trend.

Opacity is set separately, using numeric elements between zero and one: gratia.alpha.interval, gratia.alpha.partial.residual, gratia.alpha.diagnostic.point, gratia.alpha.rug, gratia.alpha.sample, gratia.alpha.observed, gratia.alpha.replicated, gratia.alpha.chain, gratia.alpha.histogram, and gratia.alpha.surface.interval. More specific elements, such as geom.gratia.smooth.interval, geom.gratia.diagnostic.interval, geom.gratia.surface.interval, geom.gratia.surface.rug, and their gratia.alpha.* counterparts, override the general component setting.

Standard fields such as colour, fill, linewidth, pointsize and pointshape set component defaults. Fields such as ink and borderwidth follow the parent ggplot2 geom's interpretation.

Default component settings preserve the existing plot appearance. Mapped aesthetics retain their scales. Explicit aesthetic arguments to plotting functions are deprecated but remain fixed overrides; use themes instead. Palette settings use palette.colour.discrete, palette.fill.discrete, palette.colour.continuous, and palette.fill.continuous. Complete scale objects can still be supplied through the existing scale arguments.

For a single ggplot, add a theme with +. For a patchwork returned by draw() or appraise(), use & to affect every panel. Plots delegated to bayesplot retain that package's styling.

Value

`gratia_theme_get()` returns a theme or `NULL`. The setter and updater invisibly return the previous override.

Examples

```
library("ggplot2")
old <- gratia_theme_set(theme(
  geom.gratia.smooth = element_geom(colour = "navy"),
  gratia.alpha.interval = 0.15
))
gratia_theme_update(geom.gratia.contour = element_geom(colour = "grey40"))
gratia_theme_set(old)

pal <- gratia_palette("tol_banded_16")

theme(palette.colour.discrete = pal, palette.fill.discrete = pal)
```

---

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 Wahba test functions*

Description

Gu and Wahba 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

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

---

has_theta *Are additional parameters available for a GAM?*

Description

Are additional parameters available for a GAM?

Usage

```
has_theta(object)
```

Arguments

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

Value

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

Examples

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

`is_by_smooth` *Tests for by variable smooths*

Description

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

Usage

```
is_by_smooth(smooth)

is_factor_by_smooth(smooth)

is_continuous_by_smooth(smooth)

by_variable(smooth)

by_level(smooth)
```

Arguments

`smooth`	an object of class `"mgcv.smooth"`

Value

A logical vector.

Author(s)

Gavin L. Simpson

`is_factor_term` *Is a model term a factor (categorical)?*

Description

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

Usage

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

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`.

See Also

gamm4_support

is_mgcv_family *Is an object one of mgcv's family objects?*

Description

Checks to determine if `object` is a family object of one of the three or so types that *mgcv* produces.

Usage

```
is_mgcv_family(object)
```

Arguments

`object`	the object to test.

Value

A logical vector of length 1, indicating if `object` is one of *mgcv*'s (`TRUE`), or 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.

is_multivariate_y *Is a model multivariate?*

Description

Determines whether a fitted model (GAM) is truly multivariate or not.

Usage

```
is_multivariate_y(model)
```

Arguments

model	a fitted model object; currently only for "gam" objects

Value

A logical vector of length 1, indicating if model is multivariate (TRUE), or otherwise (FALSE).

is_offset *Is a model term an offset?*

Description

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

Usage

```
is_offset(terms)
```

Arguments

terms	character vector of model terms.

Value

A logical vector of the same length as terms.

Author(s)

Gavin L. Simpson

Examples

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

---

link *Extract link and inverse link functions from models*

Description

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

Usage

```
link(object, ...)

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

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

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

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

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

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

inv_link(object, ...)

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

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

## S3 method for class 'bam'
```

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

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

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

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

extract_link(family, ...)

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

## S3 method for class 'general.family'
extract_link(family, parameter = NULL, 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()`. `NULL` selects the first parameter. Only parameters and aliases supported by the supplied family are accepted. For `mgcv::twlss()`, `"power"` selects the second linear predictor and `"scale"`, `"sigma"`, and `"phi"` select the third.
`which_eta`	numeric; the integer index of the linear predictor to extract for families `mgcv::mvn()` and `mgcv::multinom()`. Defaults to the first linear predictor. If multiple indices are supplied, the first is used with a warning.
`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

```
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

```
load_mgcv()
```

Value

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

---

log_lik *Posterior draws of pointwise log likelihoods*

Description

Evaluate each fitting observation at draws from the coefficient posterior. This leaves the scalar `stats::logLik()` method supplied by mgcv unchanged.

Usage

```
log_lik(model, ...)

## S3 method for class 'gam'
log_lik(
  model,
  ...,
  n_draws = NULL,
  chains = NULL,
  diagnostics = FALSE,
  seed = NULL,
  method = c("gaussian", "mh", "user"),
  draws = NULL,
  unconditional = FALSE
)

## S3 method for class 'bam'
log_lik(
  model,
  ...,
  n_draws = NULL,
  chains = NULL,
  diagnostics = FALSE,
  seed = NULL,
  method = c("gaussian", "mh", "user"),
  draws = NULL,
  unconditional = FALSE
)

## S3 method for class 'scam'
log_lik(model, ..., unconditional = FALSE)
```

Arguments

model	A fitted independent-response gam, bam, or scam model.
...	Additional arguments to post_draws(), such as burnin, thin or mvn_method. Prediction at new observations and frequentist covariance matrices are not supported.
n_draws	Positive integer; total coefficient draws across chains. Defaults to 4000 when generating draws. Supplied draws determine their own count; an explicit count must agree.
chains	positive integer or NULL; number of chains. NULL uses one chain, or the chain mapping in supplied draws. n_draws must be divisible by chains. Burn-in is applied separately to each MH chain.
diagnostics	logical; calculate coefficient R-hat, effective sample sizes, and Monte Carlo standard errors using the optional posterior package.
seed	Optional random seed. The caller's random-number state is retained.

method	One of "gaussian", "mh" or "user"; see post_draws().
draws	A numeric matrix with one row per draw and one column per coefficient, in fitted coefficient order. Used with method = "user".
unconditional	Include the smoothing-parameter covariance correction when available, using mgcv::vcov.gam(). Defaults to FALSE.

Details

The result contains log densities or log probability masses, including their normalising constants and fitting weights. Smoothing penalties and priors are excluded. Likelihood weights need not have the same interpretation as the weights used by posterior_samples() to generate responses.

Ordinary and extended response families, distributional GAMs, multinomial, multivariate-normal and grouped families are supported. A multivariate-normal observation contributes one joint log density; its precision coefficients vary with each draw. Grouped observations use their fitted component family. Custom families can supply the log_lik hook in family_diagnostics. Quasi, censored and Cox families, residual-correlated BAMs and full mixed models are not supported. Only retained fitting observations are evaluated.

Gaussian draws approximate the coefficient posterior. Parameters represented by coefficients vary jointly, including distributional-model scale and shape predictors. Other scale, shape and smoothing parameters remain at fitted values. Even with unconditional = TRUE, those additional parameters are not jointly sampled. Ordinary PSIS-LOO uses these draws without correcting their Gaussian approximation. Pareto diagnostics assess reweighting, not the accuracy of the Gaussian approximation. See loo.gam for optional loo methods.

Independent-response SCAM fits are supported using the coefficient approximation selected by post_draws(). The prediction matrix multiplies transformed coefficients. Smoothing parameters and response scale remain at their fitted values; smoothing-parameter uncertainty is unavailable.

Value

A numeric matrix with draws in rows and retained fitting observations in columns. Attribute observation gives their original fitting-row indices after any fitting subset; gratia describes the sampling assumptions. The sampling attribute retains chain identities and requested coefficient diagnostics; as_mcmc_draws() converts the likelihood matrix by chain. Impossible observations may have log likelihood -Inf; invalid numerical calculations produce an error. LOO and WAIC require finite log likelihoods.

See Also

posterior_samples(), family_diagnostics

Examples

```
load_mgcv()
m <- gam(mpg ~ s(wt, k = 5), data = mtcars, method = "REML")
ll <- log_lik(m, n_draws = 500, seed = 42)
dim(ll)
if (requireNamespace("loo", quietly = TRUE)) {
  loo::loo(ll, r_eff = 1)
}
```

loo.gam *Leave-one-out validation and WAIC for GAMs*

Description

These methods evaluate the likelihood of each fitting observation at posterior coefficient draws and return the usual objects from **loo**. Neither method changes `stats::logLik()` for fitted GAMs.

Usage

```
loo.gam(
  x,
  ...,
  n_draws = NULL,
  seed = NULL,
  sampling_args = list(),
  r_eff = NULL,
  chain_id = NULL,
  save_psis = FALSE,
  cores = 1
)

loo.bam(
  x,
  ...,
  n_draws = NULL,
  seed = NULL,
  sampling_args = list(),
  r_eff = NULL,
  chain_id = NULL,
  save_psis = FALSE,
  cores = 1
)

waic.gam(x, ..., n_draws = NULL, seed = NULL, sampling_args = list())

waic.bam(x, ..., n_draws = NULL, seed = NULL, sampling_args = list())

loo.scam(x, ...)

waic.scam(x, ...)
```

Arguments

x	a GAM or BAM fitted by **mgcv**, or an independent-response SCAM fitted by **scam**.
...	additional arguments passed to `loo::loo()` or `loo::waic()`.

n_draws	total posterior coefficient draws across all chains. Defaults to 4000 when generating draws; supplied draws determine their own count.
seed	random seed for posterior sampling.
sampling_args	named list of arguments passed to the coefficient sampler, including chains, diagnostics, method, mh_backend and method-specific controls; see log_lik().
r_eff	optional relative effective sample sizes, either one value or one per observation. Independent Gaussian draws use 1 by default. For Metropolis–Hastings draws this is estimated from the likelihoods. Supplied draws use their stored chain mapping when available; otherwise supply r_eff or chain_id.
chain_id	integer chain identifier for each coefficient draw, in draw order. Identifiers must be consecutive starting at one, and chains must contain equal numbers of draws. A supplied mapping must agree with the coefficient draws' stored mapping. Stored iterations determine temporal order for relative efficiency without reordering response or likelihood rows.
save_psis	retain the PSIS object in the returned LOO result.
cores	number of cores used by **loo**.

Details

The Gaussian posterior approximation and fitted values of additional family parameters are described in log_lik(). PSIS diagnostics assess the importance weights; they do not measure the accuracy of that approximation. With random-effect smooths, leaving out individual observations predicts further observations in the fitted groups, rather than new groups.

Likelihoods are evaluated one observation at a time using the function interface in **loo**. Saving PSIS weights still requires storage proportional to the number of observations times the number of draws.

Value

A native loo or waic object, with attributes gratia describing posterior sampling assumptions and observation identifying fitting rows.

Examples

```
if (requireNamespace("loo", quietly = TRUE)) {
  load_mgcv()
  m <- gam(mpg ~ s(wt, k = 5), data = mtcars, method = "REML")
  loo::loo(m, n_draws = 500, seed = 42)
}
```

---

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

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

lp_matrix(model, ...)

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

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

## S3 method for class 'scam'
model.matrix(object, ...)
```

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.
`object`	a fitted SCAM model, for the `model.matrix()` method.

Details

The linear prediction matrix $\mathbf{X}_p$ is a matrix that maps values of parameters $\hat{\beta}_p$ to values on the linear predictor of the model $\hat{\eta}_p = \mathbf{X}_p\hat{\beta}_p + \mathbf{o}_p$, where $\mathbf{o}_p$ contains known additive contributions such as formula offsets and prescribed smooth boundary values. These contributions are not coefficient columns of $\mathbf{X}_p$. Native prediction may attach them as `model.offset` (formula offsets, by linear predictor) or `offset` (fixed smooth contributions, by smooth) attributes. Matrix multiplication alone does not add these attributes. $\mathbf{X}_p$ is the model matrix where spline covariates have been replaced by the values of the basis functions evaluated at the respective covariates. Parametric covariates are also included.

Value

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

See Also

gamm4_support

Examples

```
load_mgcv()
```

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

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

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

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

---

mh_draws *Posterior samples using Metropolis Hastings*

Description

Posterior samples using Metropolis Hastings

Usage

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

mh_draws(model, ...)

## S3 method for class 'gam'
mh_draws(
  model,
  ...,
  n_draws = NULL,
  chains = NULL,
  n_cores = 1L,
  burnin = 1000,
  thin = 1L,
  t_df = 40,
  rw_scale = 0.25,
  diagnostics = FALSE,
  mh_backend = c("mgcv", "gratia", "family"),
  block_size = 10000L,
```

```
  dense_limit = 5e+06,
  index = NULL,
  seed = NULL
)

## S3 method for class 'scam'
mh_draws(model, ...)
```

Arguments

model	a fitted model from mgcv::gam(), mgcv::bam(), scam::scam(), or gamm4::gamm4(), or a model inheriting from these classes. See gamm4_support for sampling from gamm4 fits.
...	arguments passed to methods.
n_draws	positive integer; total retained draws across all chains. Required when generating draws. With method = "user", the supplied matrix determines the count; an explicit count must agree with its rows.
chains	positive integer or NULL; number of chains. NULL uses one chain, or the chain mapping in supplied draws. n_draws must be divisible by chains. Burn-in is applied separately to each MH chain.
n_cores	positive integer; maximum concurrent MH chains using mirai, capped at chains. Existing workers are reused; otherwise temporary workers are created and cleaned up. Only used by method = "mh".
burnin	numeric; the length of any initial burn in period to discard. See mgcv::gam.mh().
thin	positive integer; retain every thin iterations after burn-in.
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().
diagnostics	logical; calculate coefficient R-hat, effective sample sizes, and Monte Carlo standard errors using the optional posterior package.
mh_backend	character; "mgcv" preserves mgcv::gam.mh(). "gratia" uses compiled common-family likelihoods. "family" uses a compiled MH loop with callbacks to the fitted family object. Both support independent-observation bam fits. Smoothing parameters and parameters outside the fitted coefficient vector remain fixed. LSS scale and shape coefficients are sampled jointly with location. The family backend caches a full design matrix, smoothing penalty, and proposal factorisation once for all chains.
block_size	positive integer; maximum observation rows per block in the gratia MH evaluator. Large designs are evaluated without storing a full matrix.
dense_limit	numeric; maximum number of design-matrix elements to cache for the gratia MH backend. Use zero to force blockwise evaluation.
index	numeric; vector of indices of coefficients to use. Can be used to subset the mean vector and covariance matrix extracted from model.
seed	numeric or NULL; seed for the complete sampling operation. An explicit seed restores the caller's random-number state on exit.

Details

Sampling controls must be named. Removed arguments produce migration errors; see `vignette("sampling-migration")`. Explicit controls that the selected method cannot use are rejected.

For SCAM models, Gaussian draws approximate the distribution of the unrestricted fitting coefficients using `Vp` (or `Ve` with `frequentist = TRUE`), then apply scam's fitted exponential or `notExp` transformation. This is an approximation in fitting coordinates, rather than an exact constrained posterior. Supplied draws must already use the transformed coordinates in `coef(model)` order. Numerically singular covariance matrices are sampled by eigen decomposition; negative eigenvalues within `100 * .Machine$double.eps * ncol(V)` times the largest absolute eigenvalue are treated as zero. Larger negatives are errors. For weakly identified exponential coefficients, a very large fitting-coordinate variance can yield non-finite transformed draws. Sampling then reports an error because this Gaussian approximation cannot provide usable draws; supply finite transformed draws from a suitable alternative sampler.

For multiple chains, `n_draws` is the total retained count and must be divisible by `chains`. Coefficient matrices carry a chain mapping; use `as_mcmc_draws()` before ordinary matrix operations remove that mapping. Higher-level sampling tables include `.chain` and `.iteration` for multiple chains, with `.draw` unique across chains. Default single-chain table columns are unchanged. Use `sampling_diagnostics()` to extract coefficient diagnostics before response simulation adds further noise.

The gratia MH backend supports Gaussian identity, Poisson log, binomial logit, Gamma log, and negative binomial log fits. It evaluates the fitted bases at original covariates, including for discrete bam fits. GAMM components, AR1 dependence, `paraPen`, `H`, and fits made from pre-fit `G` objects are unsupported.

The family MH backend calls the fitted family's `dev.resids()` and `linkinv()` functions, or `ll()` for families with multiple linear predictors. This retains response encodings and attributes such as censoring indicators and the mappings between coefficients and predictors. It supports compatible standard, extended, and general mgcv families, including multinomial, Gaussian location-scale, and grouped families. Quasi families, Cox proportional hazards models, GAMM components, AR1 dependence, `paraPen`, `H`, and pre-fit `G` objects are unsupported. The fitted coefficient covariance must be nonsingular. It always caches the full design matrix; `block_size` and `dense_limit` apply only to the gratia backend and are rejected for the family backend. The new MH backends reject `bam` fits with nonzero prescribed smooth contributions: mgcv fitting and prediction can use different offsets for those fits. Supplied coefficient draws can still be evaluated with the native prediction conventions.

All MH backends sample the conditional coefficient posterior with fitted smoothing, dispersion, and additional family parameters fixed. Parameters represented in the fitted coefficient vector are sampled, including scale or covariance coefficients in general families. `unconditional` applies to Gaussian draws only.

GAM-class fits with linear inequality constraints, including `mgcv::scasm()` fits, store these as `Ain %*% coefficients >= bin`. Gaussian sampling uses the fitted covariance matrix without enforcing these inequalities and issues a warning. Sampled smooths can therefore violate the fitted shape constraints. Pointwise intervals computed from the same Gaussian covariance approximation also need not preserve those shapes. All MH backends are unavailable for these fits because the samplers do not enforce their inequalities.

If `mgcv::scasm()` was fitted with `bs > 0`, its `bs` component stores coefficient draws from the native constrained bootstrap in columns. These can be supplied explicitly as `draws = t(model$bs)`

with `method = "user"`. The result then describes bootstrap variation, rather than Gaussian posterior sampling. Gratia does not automatically select these draws. The caller is responsible for the interpretation and constraint feasibility of user-supplied draws.

See Also

gamm4_support

`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,
  ...
)

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

## S3 method for class 'scam'
model_concurvity(model, ...)
```

Arguments

`model`	a fitted GAM. Currently only objects of class `"gam"` are supported
`...`	arguments passed to other methods.
`terms`	currently ignored

type	character;
pairwise	logical; extract pairwise concurvity of model terms?

See Also

gamm4_support

Examples

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

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

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

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

---

model_constant *Extract the model constant term*

Description

[Experimental] Extracts the model constant term(s), the model intercept, from a fitted model object. Predictors without an intercept return zero.

Usage

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

model_constant(model, ...)

## S3 method for class 'gam'
model_constant(model, lp = NULL, ...)
```

```
## S3 method for class 'gamlss'
model_constant(model, ...)

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

## S3 method for class 'scam'
model_constant(model, lp = NULL, ...)
```

Arguments

model	a fitted model for which a coef() method exists.
...	arguments passed to other methods.
lp	numeric; which linear predictors to extract constant terms for.

See Also

gamm4_support

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_terms *Find the names of model terms*

Description

Returns the names of any terms in a model, without needing to call summary(). The list of model terms is especially useful when predicting from a mgcv::gam() model using the exclude or terms argument of mgcv::predict.gam() or mgcv::predict.bam().

Usage

```
model_terms(object, ...)

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

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

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

## S3 method for class 'lm'
model_terms(object, ...)

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

Arguments

`object`	a fitted model.
`...`	arguments to be passed to other methods; not currently used.

Details

From the point of view of *gratia*, models contain two types of term:

1. parametric terms, and
2. smooth terms.

If we consider the formula `y ~ fac + s(x2, by = fac) + s(x0)`, for a factor `fac` with three levels, there are seven terms in the model:

1. the model constant term, with name `"(Intercept)"`,
2. the parametric factor term, with names
   - `fac2`,
   - `fac3`,
3. the univariate smooth of `x0`, named `"s(x0)"`, and
4. the three factor-by smooths with names
   - `"s(x2):fac1"`,
   - `"s(x2):fac2"`, and
   - `"s(x2):fac3".‘`

`model_terms()` will return a vector of those names.

Value

A character vector of model terms.

Examples

```
load_mgcv()

# simulate data
df <- data_sim("eg4", n = 400, dist = "normal", scale = 2, seed = 1)

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

# return the names of terms in this model
model_terms(m)
```

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

Description

List the variables involved in a model fitted with a formula

Usage

```
model_vars(model, ...)

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

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

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

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

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

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

Arguments

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

Examples

```
load_mgcv()

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

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

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

---

n_eta *The Number of linear predictors in model*

Description

[Experimental] Extracts the number of linear predictors from the fitted model.

Usage

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

## S3 method for class 'scam'
n_eta(model, ...)

n_eta(model, ...)

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

Arguments

model	a fitted model. Currently, only models inheriting from class "gam" are supported.
...	arguments passed to methods.

Value

An integer vector of length 1 containing the number of linear predictors in the model.

See Also

gamm4_support

`n_smooths` *How many smooths in a fitted model*

Description

How many smooths in a fitted model

Usage

```
## S3 method for class 'gamm4'
n_smooths(object)

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

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

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()`.

Details

For `glmmTMB` models, `n_smooths()` counts smooth terms across the conditional, zero-inflation and dispersion predictors. Each fitted smooth is counted once; native random effects are excluded. Models without smooths return integer zero.

See Also

gamm4_support

nb_theta *Negative binomial parameter theta*

Description

Negative binomial parameter theta

Usage

```
nb_theta(model)

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

## S3 method for class 'gamm4'
nb_theta(model)
```

Arguments

model	a fitted model.

Value

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

Methods (by class)

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

See Also

gamm4_support

Examples

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

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

null_deviance *Extract the null deviance of a fitted model*

Description

Extract the null deviance of a fitted model

Usage

```
null_deviance(model, ...)

## Default S3 method:
null_deviance(model, ...)
```

Arguments

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

observed_fitted_plot *Plot of fitted against observed response values*

Description

Plot of fitted against observed response values

Usage

```
observed_fitted_plot(
  model,
  ylab = NULL,
  xlab = NULL,
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  point_col = "grey20",
  point_alpha = 1,
  by_family = NULL,
  conditioning = NULL,
  ...
)
```

Arguments

model	a fitted model supported by diagnostic_support.
ylab	character or expression; the label for the y axis. If not supplied, a suitable label will be generated.
xlab	character or expression; the label for the x axis. If not supplied, a suitable label will be generated.
title	character or expression; the title for the plot. See ggplot2::labs().
subtitle	character or expression; the subtitle for the plot. See ggplot2::labs().
caption	character or expression; the plot caption. See ggplot2::labs().
point_col	Deprecated; use gratia_theme_get() theme settings instead. Constant point colour when components are not mapped to colour. Pooled grouped diagnostics instead colour points by component.
point_alpha	Deprecated; use gratia_theme_get() theme settings instead. Numeric; alpha transparency for points in plots.
by_family	logical or NULL; separate indexed grouped-family components into facets. The default pools quantile residuals with component colours and facets traditional residuals. TRUE and FALSE explicitly select the layout. In appraise(), observed-versus-fitted values always use component facets. Standalone observed_fitted_plot() facets by default and permits explicit pooling. Ignored for ordinary models.
conditioning	character or NULL; "conditional" holds fitted random effects fixed, "marginal" averages over new effects, and "gam_component" uses the extracted GAM. Availability and defaults depend on the model; see diagnostic_support.
...	optional arguments passed to model preparation, such as the sdmTMB field_method. For grouped-family GAMs and joint GJRM models, response = NULL uses every response and an integer selects one; see appraise(). Selection leaves by_family independent.

---

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

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

overview(model, ...)

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

```
  model,
  parametric = TRUE,
  random_effects = TRUE,
  dispersion = NULL,
  frequentist = FALSE,
  accuracy = 0.001,
  digits = 3,
  stars = FALSE,
  ...
)

## S3 method for class 'scam'
overview(model, ...)
```

Arguments

model	a fitted model object to overview.
...	arguments passed to other methods.
parametric	logical; include the model parametric terms in the overview?
random_effects	tests of fully penalized smooth terms (those with a zero-dimensional null space, e.g. random effects) are computationally expensive and for large data sets producing these p values can take a very long time. If 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".
digits	numeric; the number of significant digits to be used.
stars	logical; should significance stars be added to the output?

See Also

gamm4_support

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

```
## S3 method for class 'gamm4'
parametric_effects(object, ...)

parametric_effects(object, ...)

## S3 method for class 'gam'
parametric_effects(
  object,
  terms = NULL,
  data = NULL,
  unconditional = FALSE,
  unnest = TRUE,
  ci_level = 0.95,
  envir = NULL,
  transform = lifecycle::deprecated(),
  n = 100,
  n_2d = 50,
  n_3d = 16,
  n_4d = 4,
  dist = 0.1,
  overall_uncertainty = TRUE,
  back_transform = FALSE,
  ...
)

## S3 method for class 'scam'
parametric_effects(object, ..., unconditional = 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`	an optional data frame containing raw model covariates at which to evaluate parametric effects. By default, stored raw columns are used; missing raw inputs are recovered from the fitting data when possible.

`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 optional environment for local functions and constants, and for recovering fitting data when required raw columns are not stored in the model. Defaults to the model's evaluation environment.

`transform` **[Deprecated]** Use `back_transform = !transform` instead.

`n, n_2d, n_3d, n_4d`
Grid resolutions for multivariate parametric terms. Curves use `n = 100`; the first two numeric surface axes use `n_2d = 50`. The third dimension of a three-variable term uses `n_3d = 16`; dimensions beyond the second of higher-dimensional terms use `n_4d = 4`. `NULL` uses `n` instead. Factors retain their levels. Ignored when `data` is supplied. Single-variable terms retain their observed evaluation values.

`dist` Non-negative distance for masking surface plots far from observed covariates, as in `draw.gam()`. Applied to the first two numeric axes when drawing; returned estimates are not masked. Use zero to disable masking.

`overall_uncertainty`
logical; include uncertainty in the intercept of the term's linear predictor, including its covariance with the term? Defaults to `TRUE`. Partial effect estimates are unchanged. With no intercept, this has no effect. Set to `FALSE` for term-only uncertainty.

`back_transform` logical; plot covariates on their original scale? Defaults to `FALSE`, using model coordinates such as `log(x)` for a term containing `log(x)`. With `TRUE`, recover the raw covariates, generate a grid on their original scale and evaluate the model expressions forwards. Applies to axes, rugs, partial-residual locations and conditioning facets, including tensor products and parametric interactions. Each smooth axis must depend on one distinct original model covariate. Functions such as `foo(x)` are supported when their raw inputs and functions can be recovered; no inverse function is needed. Missing raw data or unavailable functions produce an error; supply `data` or `envir` as appropriate. With supplied prediction `data`, unavailable raw fitting observations omit rugs and partial residuals; prediction rows are never used as observations. For precomputed smooth estimates, prepare raw grids with `smooth_estimates(model, back_transform = TRUE)` before drawing with `back_transform = TRUE`. Raw inputs supplied through `data` also retain their coordinates. Transformed-only estimates cannot be relabelled on the raw scale, including grids carrying typical values from `include_all = TRUE`. Transformed soap-film boundaries lack retained raw coordinates and must be drawn with `back_transform = FALSE`. Parametric estimates retain both scales; multi-column basis expansions retain raw axes on either setting. Does not transform estimated effects or their intervals; use `fun` for that.

Details

Each estimate is the contribution of one formula term to its linear predictor. Intercepts, main effects and other terms are not added to an interaction. Components follow the model's contrast coding:

with treatment contrasts, a numeric-by-factor interaction represents a departure from the reference-level slope, rather than a complete slope for each level.

For sdmTMB time-varying coefficients, `.type = "time_varying"` identifies coefficient trajectories. In these rows, `.value` contains time, `.partial` contains the full coefficient, and `.se` includes the joint uncertainty of its ordinary coefficient and temporal deviation. Default selection omits ordinary terms already included in these trajectories; their usual selectors remain available for explicit requests. See sdmTMB-methods for details.

A multi-column term such as `poly(x, 3)` is one component, whereas `x` and `I(x^2)` remain separate components. Scalar numeric transformations use model coordinates by default, including each axis of an interaction. Basis expansions such as `poly(x, 3)`, terms combining several inputs in one expression, and interactions containing several transformations of the same input retain raw axes because they have no single model coordinate per input. Other covariates in generated prediction data are held at typical values solely to evaluate the model matrix; their contributions are not included. Rugs show fitting observations, including when prediction `data` are supplied. If fitting observations cannot be recovered, these rugs are omitted.

The drawing method uses the first two numeric covariates for surface axes, grouping curves by the first factor when there is only one numeric covariate. Factor-only terms use grouped points and intervals. Remaining covariates define facets: one wraps, two or more use the first as rows and the rest as columns. Formula order is retained within numeric and discrete covariates. Logical covariates are discrete. Surface plots show estimates; uncertainty is retained in the returned data rather than drawn as additional surfaces.

For models with multiple linear predictors, plot subtitles identify each term's distributional parameter or model component, as for smooth plots. Shared terms name every predictor to which they contribute. The optional `.model` column retains these labels, including with `unnest = FALSE`. Single-predictor models omit automatic subtitles. Older saved estimates remain drawable when predictor labels are absent.

Value

A tibble of class `parametric_effects`, with `.term`, `.type`, `.partial` and `.se`. Single-variable terms retain `.value` or `.level`; multivariate terms use columns named for their raw covariates, containing coordinates on the selected scale. Names conflicting with reserved columns are repaired with numeric suffixes. The `term_info` attribute records multivariate column mappings, plotting order, factor levels and observation data. With `unnest = FALSE`, estimates and covariates are nested in a `data` list column. The `axis_info` attribute identifies retained raw and model coordinate columns and their labels so the drawing method can select either scale. The `back_transform` attribute records the scale of the returned coordinate columns. Auxiliary `.gratia_raw_*` and `.gratia_model_*` columns retain both scales for transformed axes.

See Also

gamm4_support

Examples

```
load_mgcv()

# Simulate numeric and factor effects
```

```
set.seed(42)
d <- data.frame(
  x = runif(100),
  group = factor(rep(c("A", "B"), each = 50))
)
d$y <- with(d, 2 * x + (group == "B") + rnorm(100, sd = 0.25))
m <- gam(y ~ x + group, data = d, method = "REML")

# Estimate and draw the contribution of each parametric term
pe <- parametric_effects(m)
pe
draw(pe)
```

`parametric_terms` *Names of any parametric terms in a GAM*

Description

Names of any parametric terms in a GAM

Usage

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

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

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.

See Also

gamm4_support

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 = NULL,
  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,
  envir = NULL,
  wrt = c("smooth", "covariate"),
  interval_method = c("auto", "gaussian", "bootstrap"),
  simultaneous_method = c("simulation", "tube"),
  simultaneous_domain = NULL,
  tube_control = list()
)
```

```
## S3 method for class 'gamm4'
partial_derivatives(object, ...)

## S3 method for class 'scam'
partial_derivatives(object, ..., unconditional = FALSE)
```

Arguments

object	an R object to compute derivatives for.
...	arguments passed to other methods.
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.
focal	character; name of the focal variable. The partial derivative of the estimated smooth with respect to this variable will be returned. All other variables involved in the smooth will be held at constant. This can be missing if supplying data, in which case, the focal variable will be identified as the one variable that is not constant.
data	a data frame containing the values of the model covariates at which to evaluate the first derivatives of the smooths. If supplied, all but one variable must be held at a constant value.
order	numeric; the order of derivative.
type	character; the type of finite difference used. One of "forward", "backward", or "central".
n	numeric; the number of points to evaluate the derivative at.
eps	a positive finite number giving the absolute finite-difference step, or NULL (the default) to choose it automatically. The automatic step is the fitted range of the differentiation coordinate multiplied by .Machine$double.eps^(1 / (order + p)), where p = 2 for central differences and p = 1 for forward or backward differences. A constant coordinate uses its absolute value (or one if zero) instead of its range. The step is rounded upwards and bounded below so that adding it to the coordinates produces a representable change. Stored transformed values or raw covariate summaries supply the range; supplied data are used if neither is available. This is a scale-aware heuristic, not an error bound; unusual function scales or domain boundaries may require explicit eps. For first central differences the two points are separated by eps; for second central differences they are each eps from the target.
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
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.
envir	optional environment for functions and constants in model expressions; see smooth_estimates().
wrt	differentiation coordinate: "smooth" (the default) uses the smooth expression, such as log(x); "covariate" uses a raw covariate, such as x, and reevaluates expressions at each finite-difference point.
interval_method	character; use stored scasm bootstrap draws for pointwise intervals with "auto" (the default), or explicitly select "gaussian" or "bootstrap". Bootstrap simultaneous bands are unavailable.
simultaneous_method	character; calibration for simultaneous bands. The default, "simulation", uses coefficient draws on the evaluation grid. "tube" uses a conservative Gaussian threshold-crossing bound over a continuous interval for supported one-dimensional cubic splines. See simultaneous_bands.
simultaneous_domain	numeric pair of increasing finite endpoints, or a named list of such pairs keyed by spline coordinate. For tube bands, NULL uses the range of returned evaluation coordinates, including gaps. All returned points must lie inside the interval and the interval must lie inside the fitted spline support. Requires tube calibration.
tube_control	named list controlling numerical tube calibration: rel_tol (default 1e-6), abs_tol (1e-8), subdivisions (1000L), and root_tol (1e-8). Requires tube calibration. Simulation controls n_sim, n_cores (or ncores), and seed are ignored by tube calibration.

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 ± (crit * se) gives the upper and lower bounds of the requested confidence or simultaneous interval (given level); NA for bootstrap limits,
- .lower_ci: the lower bound of the confidence or simultaneous interval,
- .upper_ci: the upper bound of the confidence or simultaneous interval.

Parallel calculation

With mirai and carrier installed, configure workers with `mirai::daemons()` to distribute calculations across selected smooths. `difference_smooths()` distributes factor-level comparisons instead. Without configured workers, or with only one calculation, evaluation is serial. These functions leave the caller's worker pool running.

Model selection, evaluation of model expressions and random draws are prepared in the caller. Workers receive the coordinates and coefficient columns needed for their calculation. Custom prediction methods are evaluated in the caller, including custom tensor margins and factor-smooth bases; workers receive their prediction matrices. Simultaneous intervals use successive draws in selection order, so `set.seed()` gives the same results with or without workers, independently of their number or completion order. `derivatives()` uses `set.seed()`; the other functions also accept `seed`, which restores the caller's random number state on exit.

`smooth_samples()` draws coefficients jointly once, then reuses those draws for every smooth, preserving dependence and chain identities. Sampling finishes before smooth calculations start, so MH chains and smooth workers are not nested. Sampling thread controls keep their existing meanings. Workers use their configured BLAS thread settings; when using a threaded BLAS, limit its threads before starting daemons to avoid competing threads.

Parallel work is most useful for expensive bases or interval calculations. Small grids can be slower because of transfer and scheduling costs. Large grids and many draws also require memory for worker inputs and returned results, in addition to the final output.

Note

`partial_derivatives()` will ignore any random effect smooths it encounters in `object`. For soap-film smooths with known boundary values, derivatives include the fixed boundary contribution. This shifts the estimates and interval limits without adding uncertainty from fitted coefficients.

Author(s)

Gavin L. Simpson

See Also

`difference_smooths()` with `order > 0` and `focal` to compute partial derivatives of differences across full smooth surfaces.

gamm4_support

Examples

```
load_mgcv()
# Simulate a smooth surface with Gaussian noise and fit a tensor-product smooth
df <- data_sim("eg2", n = 400, scale = 0.1, seed = 42)
m <- gam(y ~ te(x, z, k = c(7, 7)), data = df, method = "REML")

# Estimate change along x while holding z at 0.4
ds <- data_slice(m, x = evenly(x, n = 100), z = 0.4)
pd_x <- partial_derivatives(m, data = ds, type = "central", focal = "x")
pd_x
draw(pd_x)
```

partial_residuals *Partial residuals*

Description

Partial residuals

Usage

```
partial_residuals(object, ...)

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

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

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.

Details

With na.exclude, excluded observations are restored as NA partial residuals, in fitting order after any subset. With na.omit, only retained observations are returned. Computation uses aligned working residuals, working weights and term predictions before restoring excluded positions.

Partial residuals require working residuals and working weights from the fitted model. Families that do not provide these, including the location-scale families in mgcv, are not supported, even for terms in the location predictor.

Examples

```
## load mgcv
load_mgcv()

## example data - Gu & Wahba four term model
```

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

## extract partial residuals
partial_residuals(m)

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

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

---

penalty *Extract and tidy penalty matrices*

Description

Extract and tidy penalty matrices

Usage

```
## S3 method for class 'gamm4'
penalty(object, ...)

penalty(object, ...)

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

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

```
  ...
)

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

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

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

## S3 method for class 're.smooth.spec'
penalty(object, data, rescale = FALSE, margins = FALSE, ...)

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

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().
diagonalize	logical; if TRUE, reparameterises the smooth such that the associated penalty is an identity matrix. This has the effect of turning the last diagonal elements of the penalty to zero, which highlights the penalty null space.
margins	logical; if FALSE, extract the stored tensor product penalty matrices. If TRUE, extract the stored marginal penalties instead, with labels identifying the parent tensor and each margin. Marginal penalties need not use the parameterisation of the fitted tensor product.
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.

Details

For SCAM fits, the returned penalties act on the unrestricted fitting parameters, before scam transforms selected coefficients to impose positivity. The `parameterisation` attribute records this distinction.

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

See Also

gamm4_support

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")
```

`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

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

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

post_draws(model, ...)

## Default S3 method:
post_draws(
  model,
  ...,
  n_draws = NULL,
  chains = NULL,
  seed = NULL,
  method = c("gaussian", "mh", "inla", "user"),
  n_cores = 1L,
  n_threads = 1L,
  burnin = 1000,
  thin = 1L,
  t_df = 40,
  rw_scale = 0.25,
  frequentist = FALSE,
  unconditional = FALSE,
  draws = NULL,
  diagnostics = FALSE,
  mh_backend = c("mgcv", "gratia", "family"),
  mvn_method = c("mvnfast", "mgcv"),
```

```
  block_size = 10000L,
  dense_limit = 5e+06,
  index = NULL,
  parametrized = TRUE
)

generate_draws(model, ...)

## S3 method for class 'gam'
generate_draws(
  model,
  ...,
  n_draws = NULL,
  chains = NULL,
  seed = NULL,
  method = c("gaussian", "mh", "inla", "user"),
  n_cores = 1L,
  n_threads = 1L,
  burnin = 1000,
  thin = 1L,
  t_df = 40,
  rw_scale = 0.25,
  frequentist = FALSE,
  unconditional = FALSE,
  draws = NULL,
  diagnostics = FALSE,
  mh_backend = c("mgcv", "gratia", "family"),
  mvn_method = c("mvnfast", "mgcv"),
  block_size = 10000L,
  dense_limit = 5e+06,
  index = NULL
)

## S3 method for class 'scam'
generate_draws(model, ...)
```

Arguments

model	a fitted model from mgcv::gam(), mgcv::bam(), scam::scam(), or gamm4::gamm4(), or a model inheriting from these classes. See gamm4_support for sampling from gamm4 fits.
...	arguments passed to methods.
n_draws	positive integer; total retained draws across all chains. Required when generating draws. With method = "user", the supplied matrix determines the count; an explicit count must agree with its rows.
chains	positive integer or NULL; number of chains. NULL uses one chain, or the chain mapping in supplied draws. n_draws must be divisible by chains. Burn-in is applied separately to each MH chain.

seed	numeric or NULL; seed for the complete sampling operation. An explicit seed restores the caller's random-number state on exit.
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 Hastings 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.
n_cores	positive integer; maximum concurrent MH chains using mirai, capped at chains. Existing workers are reused; otherwise temporary workers are created and cleaned up. Only used by method = "mh".
n_threads	positive integer; threads within Gaussian sampling using mvnfast. Defaults to one. This is separate from concurrent MH chains.
burnin	numeric; the length of any initial burn in period to discard. See mgcv::gam.mh().
thin	positive integer; retain every thin iterations after burn-in.
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().
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(). SCAM models do not support this correction; use unconditional = FALSE.
draws	matrix; user supplied posterior draws to be used when method = "user".
diagnostics	logical; calculate coefficient R-hat, effective sample sizes, and Monte Carlo standard errors using the optional posterior package.
mh_backend	character; "mgcv" preserves mgcv::gam.mh(). "gratia" uses compiled common-family likelihoods. "family" uses a compiled MH loop with callbacks to the fitted family object. Both support independent-observation bam fits. Smoothing parameters and parameters outside the fitted coefficient vector remain fixed. LSS scale and shape coefficients are sampled jointly with location. The family backend caches a full design matrix, smoothing penalty, and proposal factorisation once for all chains.
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.
block_size	positive integer; maximum observation rows per block in the gratia MH evaluator. Large designs are evaluated without storing a full matrix.
dense_limit	numeric; maximum number of design-matrix elements to cache for the gratia MH backend. Use zero to force blockwise evaluation.

index	numeric; vector of indices of coefficients to use. Can be used to subset the mean vector and covariance matrix extracted from model.
parametrized	logical; for SCAM fits, transform Gaussian draws from the unrestricted fitting coefficients using the fitted positivity transform. This preserves the smooth's constraints. With FALSE, return Gaussian draws in the unrestricted fitting coordinates. Only for scam::scam() fits.

Details

Sampling controls must be named. Removed arguments produce migration errors; see vignette("sampling-migration"). Explicit controls that the selected method cannot use are rejected.

For SCAM models, Gaussian draws approximate the distribution of the unrestricted fitting coefficients using Vp (or Ve with frequentist = TRUE), then apply scam's fitted exponential or notExp transformation. This is an approximation in fitting coordinates, rather than an exact constrained posterior. Supplied draws must already use the transformed coordinates in coef(model) order. Numerically singular covariance matrices are sampled by eigen decomposition; negative eigenvalues within 100 * .Machine$double.eps * ncol(V) times the largest absolute eigenvalue are treated as zero. Larger negatives are errors. For weakly identified exponential coefficients, a very large fitting-coordinate variance can yield non-finite transformed draws. Sampling then reports an error because this Gaussian approximation cannot provide usable draws; supply finite transformed draws from a suitable alternative sampler.

For multiple chains, n_draws is the total retained count and must be divisible by chains. Coefficient matrices carry a chain mapping; use as_mcmc_draws() before ordinary matrix operations remove that mapping. Higher-level sampling tables include .chain and .iteration for multiple chains, with .draw unique across chains. Default single-chain table columns are unchanged. Use sampling_diagnostics() to extract coefficient diagnostics before response simulation adds further noise.

The gratia MH backend supports Gaussian identity, Poisson log, binomial logit, Gamma log, and negative binomial log fits. It evaluates the fitted bases at original covariates, including for discrete bam fits. GAMM components, AR1 dependence, paraPen, H, and fits made from pre-fit G objects are unsupported.

The family MH backend calls the fitted family's dev.resids() and linkinv() functions, or ll() for families with multiple linear predictors. This retains response encodings and attributes such as censoring indicators and the mappings between coefficients and predictors. It supports compatible standard, extended, and general mgcv families, including multinomial, Gaussian location-scale, and grouped families. Quasi families, Cox proportional hazards models, GAMM components, AR1 dependence, paraPen, H, and pre-fit G objects are unsupported. The fitted coefficient covariance must be nonsingular. It always caches the full design matrix; block_size and dense_limit apply only to the gratia backend and are rejected for the family backend. The new MH backends reject bam fits with nonzero prescribed smooth contributions: mgcv fitting and prediction can use different offsets for those fits. Supplied coefficient draws can still be evaluated with the native prediction conventions.

All MH backends sample the conditional coefficient posterior with fitted smoothing, dispersion, and additional family parameters fixed. Parameters represented in the fitted coefficient vector are sampled, including scale or covariance coefficients in general families. unconditional applies to Gaussian draws only.

GAM-class fits with linear inequality constraints, including `mgcv::scasm()` fits, store these as `Ain %*% coefficients >= bin`. Gaussian sampling uses the fitted covariance matrix without enforcing these inequalities and issues a warning. Sampled smooths can therefore violate the fitted shape constraints. Pointwise intervals computed from the same Gaussian covariance approximation also need not preserve those shapes. All MH backends are unavailable for these fits because the samplers do not enforce their inequalities.

If `mgcv::scasm()` was fitted with `bs > 0`, its `bs` component stores coefficient draws from the native constrained bootstrap in columns. These can be supplied explicitly as `draws = t(model$bs)` with `method = "user"`. The result then describes bootstrap variation, rather than Gaussian posterior sampling. Gratia does not automatically select these draws. The caller is responsible for the interpretation and constraint feasibility of user-supplied draws.

See Also

gamm4_support

`posterior_predictive_plot` *Posterior predictive check of the response distribution*

Description

Compare the observed response distribution with replicated data sets drawn from the approximate posterior predictive distribution of a fitted model.

Usage

```
posterior_predictive_plot(
  model,
  n = 50,
  seed = NULL,
  observed_col = "#D55E00",
  replicated_col = "#0072B2",
  replicated_alpha = NULL,
  xlab = NULL,
  ylab = NULL,
  title = "Posterior predictive check",
  subtitle = NULL,
  caption = NULL,
  level = 0.9,
  conditioning = NULL,
  scalable = NULL,
  scalable_threshold = 10000L,
  batch_size = 10L,
  ...,
  response = NULL
)
```

Arguments

model	a fitted model with a posterior response sampler.
n	positive integer; number of replicated data sets to draw.
seed	numeric; optional random seed. With an explicit seed, the caller's random-number state is preserved.
observed_col, replicated_col	Deprecated; use gratia_theme_get() theme settings instead. Colours for the observed and replicated distributions, respectively. Standalone plots default to orange-red ("#D55E00") and blue ("#0072B2"). When called by appraise(), the check uses that function's line_col and ci_col instead.
replicated_alpha	Deprecated; use gratia_theme_get() theme settings instead. Numeric or NULL; opacity of replicated distributions. The default, NULL, uses 0.8 for discrete posterior median points and interval bars, and 0.2 for continuous density curves.
xlab	character or expression; label for the response axis.
ylab	character or expression; the label for the y axis. If not supplied, a suitable label will be generated.
title	character or expression; the title for the plot. See ggplot2::labs().
subtitle	character or expression; the subtitle for the plot. See ggplot2::labs().
caption	character or expression; the plot caption. See ggplot2::labs().
level	numeric; coverage of the equal-tailed posterior predictive intervals for discrete responses. Must be strictly between zero and one. Defaults to 0.9. Ignored for continuous density overlays.
conditioning	character or NULL; mixed-model conditioning mode.
scalable	logical or NULL; summarize posterior response draws on a common grid, using temporary disk storage instead of retaining all responses in memory. NULL enables this at scalable_threshold observations.
scalable_threshold	positive integer; observation count for automatic scalable plotting.
batch_size	positive integer; number of draws per processing batch. Batch boundaries do not alter seeded draws. Scalable discrete comparisons with more than 512 distinct response values aggregate probability into 512 shared equal-width bins to bound memory and plot size.
...	arguments passed to posterior_samples(), such as unconditional or mvn_method. For delta sdmTMB fits, component = "response", "binomial", or "positive" selects the complete response or a fitted component. Prediction at new data is not supported.
response	For mgcv::gfam() models, NULL uses all components; a single integer selects observations assigned to that family-list position. Repeated families retain distinct indices. Each selected component is displayed in its own facet.

Details

Mixed-model captions name the conditioning mode. Supply `caption` to replace this text, or `caption = ""` to omit it.

Each replicated data set includes uncertainty in the coefficients and random variation in the responses, at the covariate values used to fit the model. Linear models and GLMs draw coefficients from a Gaussian distribution with the estimated coefficient covariance, keeping dispersion at its fitted value. For GAMs that model several distribution parameters, all their coefficients are drawn together. Distribution parameters outside this coefficient vector remain at fitted values, including additional parameters of grouped-family components.

glmmTMB draws mean, zero-inflation and dispersion coefficients together with fitted random effects, using a Gaussian (Laplace) approximation to their posterior. Their covariance includes uncertainty from estimating the parameters that control random-effect covariance. For conditional checks, responses are generated using the drawn values of the fitted random effects. Given those values, their prior covariance is not needed to generate responses. Additional family parameters, such as Tweedie power, remain at fitted values. For marginal checks, coefficient draws include the same estimation uncertainty, but new random effects are generated using the fitted prior covariance. Thus these checks keep the covariance of new effects fixed while retaining its estimation uncertainty in the coefficient draws; they do not condition the coefficient posterior on fixed covariance parameters. sdmTMB draws parameters together from a Gaussian approximation, including fixed coefficients and random fields, and uses sdmTMB's response simulator to generate the complete response. These plots do not require an analytic residual distribution, so native simulated families such as Student-t and beta-binomial are available. SCAM models support ordinary and scalable predictive checks using transformed coefficient draws. Gaussian identity-link AR1 fits retain the fitted response correlation and the series boundaries stored in `AR.start`. Gaussian `gamm` and `gamm4` models draw fixed and random effects together, keeping covariance parameters at fitted values; marginal checks draw new effects. For gamm fits estimated by penalized quasi-likelihood (PQL), the sampler uses the Gaussian approximation used in fitting and then generates responses from the fitted response family. This is supported only when the Gaussian working residuals are uncorrelated. Correlation in those working residuals does not specify a joint distribution for responses from the fitted family. `gamm4` defaults to `conditioning = "conditional"`, checking the full fitted mixed model while including uncertainty in fitted effects. Use the explicit `"gam_component"` option for the extracted GAM calculations. GJRM draws the final distributional coefficients jointly for supported uncensored univariate families, including BE, LO, WEI, IG, GU and rGU. Both distributional equations retain their coefficient covariance; beta scale draws use its logit link. See diagnostic_support for the family list. These are approximate predictive checks; no refitting is performed.

Continuous responses use kernel density estimates with a common bandwidth derived from the observed responses and a shared evaluation grid. Discrete responses show observed empirical probability masses as points, alongside posterior medians and equal-tailed intervals across replicated data sets. Observed points, predicted points and intervals are placed at the actual response values on the horizontal axis. Binomial responses are compared as proportions, including models fitted to success and failure counts. Each fitting observation contributes equally to the empirical distribution. Response sampling follows the fitted family's interpretation of weights; see `posterior_samples()`. Factor-valued binomial observations retain one Bernoulli trial per observation, and native gaulss fitting weights do not change its response precision. Explicit GAM sampling weights override the defaults. Multivariate normal responses are shown in separate facets.

Excluded observations are omitted from both observed and replicated data. Responses remain on the scale used in fitting, including transformations in the response formula. Grouped families use

separate component facets. Censored responses require a model of how censoring changes the generated observations and are not supported. Unsupported models or samplers produce an error; no alternative simulation method is substituted. Additional sampler limitations are described in posterior_samples().

Value

A ggplot object. Its data contain the observed responses (.draw = 0) and posterior replicates (.draw > 0), indexed by .row and .parameter. Scalable plots instead retain density or probability summaries evaluated on the same grid for all draws.

See Also

appraise(), posterior_samples()

Examples

```
load_mgcv()
dat <- data_sim("eg1", n = 100, seed = 2)
mod <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3),
  data = dat,
  method = "REML"
)
posterior_predictive_plot(mod, n = 20, seed = 42)
appraise(mod, ppcheck = TRUE, n_ppc = 20, seed = 42)
```

---

posterior_samples.lm *Draw samples from the posterior distribution of an estimated model*

Description

Draw samples from the posterior distribution of an estimated model

Usage

```
## S3 method for class 'lm'
posterior_samples(model, ..., n_draws = NULL, seed = NULL)

## S3 method for class 'glm'
posterior_samples(model, ..., n_draws = NULL, seed = NULL)

## S3 method for class 'glmmTMB'
posterior_samples(model, ..., n_draws = NULL, seed = NULL, conditioning = NULL)

## S3 method for class 'sdmTMB'
posterior_samples(model, ..., n_draws = NULL, seed = NULL, conditioning = NULL)

## S3 method for class 'gamlss'
```

```
posterior_samples(model, ..., n_draws = NULL, seed = NULL)

## S3 method for class 'gamm'
posterior_samples(model, ..., n_draws = NULL, seed = NULL, conditioning = NULL)

## S3 method for class 'lme'
posterior_samples(model, ..., n_draws = NULL, seed = NULL, conditioning = NULL)

## S3 method for class 'merMod'
posterior_samples(model, ..., n_draws = NULL, seed = NULL, conditioning = NULL)

## S3 method for class 'gamm4'
posterior_samples(model, ..., n_draws = NULL, seed = NULL, conditioning = NULL)

posterior_samples(model, ...)

## S3 method for class 'gam'
posterior_samples(
  model,
  ...,
  data = NULL,
  n_draws = NULL,
  chains = NULL,
  seed = NULL,
  method = c("gaussian", "mh", "inla", "user"),
  n_cores = 1L,
  n_threads = 1L,
  burnin = 1000,
  thin = 1L,
  t_df = 40,
  rw_scale = 0.25,
  frequentist = FALSE,
  unconditional = FALSE,
  draws = NULL,
  diagnostics = FALSE,
  mh_backend = c("mgcv", "gratia", "family"),
  mvn_method = c("mvnfast", "mgcv"),
  block_size = 10000L,
  dense_limit = 5e+06,
  weights = NULL,
  family_index = NULL,
  marginalise = NULL
)

## S3 method for class 'scam'
posterior_samples(
  model,
  ...,
```

```
  data = NULL,
  n_draws = NULL,
  chains = NULL,
  seed = NULL,
  method = c("gaussian", "mh", "inla", "user"),
  n_cores = 1L,
  n_threads = 1L,
  burnin = 1000,
  thin = 1L,
  t_df = 40,
  rw_scale = 0.25,
  frequentist = FALSE,
  unconditional = FALSE,
  draws = NULL,
  diagnostics = FALSE,
  mh_backend = c("mgcv", "gratia", "family"),
  mvn_method = c("mvnfast", "mgcv"),
  block_size = 10000L,
  dense_limit = 5e+06,
  weights = NULL,
  family_index = NULL
)
```

Arguments

model	a fitted model of the supported types; see diagnostic_support.
...	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_draws	positive integer; total retained draws across all chains. Required when generating draws. With method = "user", the supplied matrix determines the count; an explicit count must agree with its rows.
seed	numeric; a random seed for the simulations.
conditioning	character or NULL; mixed-model conditioning mode; see diagnostic_support.
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.
chains	positive integer or NULL; number of chains. NULL uses one chain, or the chain mapping in supplied draws. n_draws must be divisible by chains. Burn-in is applied separately to each MH chain.
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 as a coefficient matrix.

`n_cores` positive integer; maximum concurrent MH chains using mirai, capped at `chains`. Existing workers are reused; otherwise temporary workers are created and cleaned up. Only used by `method = "mh"`.

`n_threads` positive integer; threads within Gaussian sampling using mvnfast. Defaults to one. This is separate from concurrent MH chains.

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

`thin` positive integer; retain every `thin` iterations after burn-in.

`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"`.

`frequentist` 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 `frequentist == 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"`.

`diagnostics` logical; calculate coefficient R-hat, effective sample sizes, and Monte Carlo standard errors using the optional posterior package.

`mh_backend` character; `"mgcv"` preserves `mgcv::gam.mh()`. `"gratia"` uses compiled common-family likelihoods. `"family"` uses a compiled MH loop with callbacks to the fitted family object. Both support independent-observation bam fits. Smoothing parameters and parameters outside the fitted coefficient vector remain fixed. LSS scale and shape coefficients are sampled jointly with location. The family backend caches a full design matrix, smoothing penalty, and proposal factorisation once for all chains.

`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.

`block_size` positive integer; maximum observation rows per block in the gratia MH evaluator. Large designs are evaluated without storing a full matrix.

`dense_limit` numeric; maximum number of design-matrix elements to cache for the gratia MH backend. Use zero to force blockwise evaluation.

`weights` numeric; response sampling weights for GAM methods. An explicit vector overrides the fitting weights; a single value applies to every observation. With `data = NULL`, the default is the fitted prior weights, except factor-valued binomial responses retain one Bernoulli trial per observation. Native `gaulss()` ignores fitting weights in its likelihood, so its default response weights are also ones; explicitly supplied sampling weights still override this default. With prediction

data, weights default to a vector of ones. For ordinary Gaussian and Gamma responses, larger weights reduce response variance (precision weights). For binomial responses, weights give integer trial counts. They do not alter the fitted coefficient covariance. Include positions restored by `na.exclude` when supplying weights for fitting rows.

`family_index` numeric; optional family-list positions for grouped family (`mgcv::gfam()`) response draws, one integer per evaluation row. Membership can instead be recovered from the response-index expression in `data`; observed response values are not required.

`marginalise` `NULL` retains the usual predictions. `TRUE` selects all supported `re` and `fs` terms; character vectors select exact smooth labels. Fitted predictions integrate response means over new zero-mean Gaussian effects. Response samplers draw new effects before sampling responses. Smoothing parameters and dispersion remain fixed. Only ordinary `gam()` and `bam()` fits with a scalar response mean are supported.

Details

The `lm`, `glm`, `glmmTMB`, `GJRM::gamlss`, `sdmTMB`, and full mixed-model diagnostic samplers currently use the fitting observations only. They draw coefficients together using Gaussian approximations. See `posterior_predictive_plot()` for which additional distribution parameters remain fixed and how random effects are treated for each fitter.

With no explicit `data`, observation rows follow the fitted model's `na.action`. `na.exclude` restores excluded rows as `NA` in every draw; `na.omit` returns retained rows only. `.row` indexes the fitting data after any `subset`. With explicit `data`, `.row` indexes that data, missing responses do not prevent prediction, and missing required predictors give `NA` draws. An explicit prediction `na.action` in `...` is honoured without renumbering retained rows. Restoration does not consume random numbers.

For Gaussian identity-link `bam` and `scam` fits, first-order autoregressive (AR1) errors make consecutive responses correlated. Responses are drawn together to retain this fitted correlation, restarting at each `AR.start` segment. These draws currently use the fitting observations only. For other families or links, the AR1 correction applies to working residuals used during fitting. It does not specify a joint distribution of actual responses, so sampling produces an error.

Location-scale-shape families `mgcv::gaulss()`, `mgcv::gammals()`, `mgcv::gumbls()`, `mgcv::gevlss()`, `mgcv::shash()`, `mgcv::twlss()` and `mgcv::ziplss()` support fitting observations, subsets and new covariates. Every distribution parameter is computed from the same joint coefficient draw. Formula offsets and prescribed smooth contributions are retained. Evaluation data need predictors and formula-offset variables, but no observed response. mgcv natively represents the second `gaulss()` parameter as reciprocal standard deviation: response sampling uses SD, divided by the square root of the sampling weight. Other LSS generators currently ignore sampling weights, following their mgcv implementations. Parameters represented by coefficients vary jointly; other family parameters remain fixed.

Cox survival models require evaluation times and an observation process and do not have a response generator for this interface. Censored-family draws are latent, uncensored responses; reproducing observed censoring requires an observation process and remains unsupported in posterior predictive checks. Grouped-family posterior response draws support fitting observations and supplied data with known family membership. Component nuisance parameters remain fixed. `.parameter`

identifies the component used for each row. Censored grouped components require an observation process; use `simulate()` for their latent, uncensored responses.

With `marginalise`, selected random effects are drawn for new groups in each response replicate. Group labels must be supplied for new data. See `random_effect_samples()` for the estimated Gaussian distributions.

Value

A tibble (data frame) in long format, with columns:

- `.row`: row of the evaluation data;
- `.draw`: globally unique draw identifier;
- `.response`: simulated response. Where needed, `.parameter` identifies the response or grouped component. Multiple-chain output also includes `.chain` and `.iteration`. Attached coefficient diagnostics are available through `sampling_diagnostics()`.

Note

With `data = NULL`, fitting offsets are retained, including offsets supplied through the fitting function's `offset` argument. Explicit prediction data follow `mgcv::predict.gam()`: offsets in formulas are evaluated from the prediction data, while offsets supplied through the fitting function's `offset` argument are not transferred to new observations. Include offsets in the formula when predicting at new data; the required offset variables must then be present in `data`.

Author(s)

Gavin L. Simpson

References

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

See Also

gamm4_support

---

pp_check *Posterior checks with bayesplot*

Description

Compare observations with replicated responses using an optional installation of bayesplot. Call `bayesplot::pp_check()` after loading gratia.

Usage

```
pp_check.gam(
  object,
  ...,
  type = "dens_overlay",
  prefix = c("ppc", "ppd"),
  n_draws = NULL,
  seed = NULL,
  group = NULL,
  x = NULL,
  response = NULL,
  conditioning = NULL,
  sampling_args = list()
)

pp_check.bam(
  object,
  ...,
  type = "dens_overlay",
  prefix = c("ppc", "ppd"),
  n_draws = NULL,
  seed = NULL,
  group = NULL,
  x = NULL,
  response = NULL,
  conditioning = NULL,
  sampling_args = list()
)

pp_check.gamm(
  object,
  ...,
  type = "dens_overlay",
  prefix = c("ppc", "ppd"),
  n_draws = NULL,
  seed = NULL,
  group = NULL,
  x = NULL,
  response = NULL,
  conditioning = NULL,
  sampling_args = list()
)

pp_check.gamm4(
  object,
  ...,
  type = "dens_overlay",
  prefix = c("ppc", "ppd"),
```

```
  n_draws = NULL,
  seed = NULL,
  group = NULL,
  x = NULL,
  response = NULL,
  conditioning = NULL,
  sampling_args = list()
)

pp_check.scam(object, ...)
```

Arguments

object	a fitted GAM, BAM, gamm, or gamm4 model.
...	arguments passed only to the selected bayesplot plotting function.
type	character; the bayesplot plot name without its ppc_ or ppd_ prefix. Available names come from the installed version of bayesplot.
prefix	character; "ppc" compares responses with observations; "ppd" shows replicated responses alone.
n_draws	positive integer; total coefficient and response draws. Defaults to 50 for ordinary plots and 4000 for leave-one-out plots. With supplied coefficient draws, defaults to their number of rows.
seed	integer or NULL; seed for reproducible sampling.
group, x	a variable name in the fitting model frame, or a vector in fitting-row order. Missing fitting rows are removed with the response.
response	character or integer; response column or grouped-family component to check, for example "response2" or "component1". Required when the model contains several responses or grouped components.
conditioning	character or NULL; conditioning passed to the existing posterior predictive sampler. See posterior_predictive_plot().
sampling_args	named list; arguments for the posterior sampler, including method, draws, unconditional, and mvn_method. Prediction at new data is not supported. Fitted binomial trials must be retained. Use chains for multiple coefficient chains and diagnostics = TRUE for coefficient diagnostics. Leave-one-out plots use stored chain identities; for unlabelled supplied draws, include r_eff or chain_id here.

Details

These methods require bayesplot only when called. Leave-one-out plots also require loo and a likelihood supported by log_lik(). Their response and likelihood calculations use exactly the same coefficient draws. For a multivariate response, the weights leave out the entire response vector. Grouped-family checks use only the selected component's observations.

Some bayesplot plotting functions, including error_binned and LOO-PIT plots, also require bayesplot's optional rstantools package. Install it with install.packages("rstantools") to use those plots.

Binomial observations and replicated responses are shown as numbers of successes with fitted trial counts. Other gratia predictive plots use proportions. The `error_binned` plot uses expected responses rather than response simulations and requires an available finite response mean. For gamm and gamm4 it currently requires `conditioning = "gam_component"`. Plotting functions requiring additional inputs, such as censoring indicators, receive those inputs through ....

Coefficient draws use a Gaussian approximation by default; additional family parameters remain at fitted values. See `log_lik()` for the assumptions of approximate leave-one-out calculations. Existing conditioning and response dependence restrictions of the diagnostic samplers also apply here.

Value

The object returned by the selected bayesplot function, usually a ggplot object.

See Also

`posterior_predictive_plot()`, `log_lik()`

Examples

```
if (requireNamespace("bayesplot", quietly = TRUE)) {
  load_mgcv()
  dat <- data_sim("eg1", n = 100, seed = 2)
  fit <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = dat)
  bayesplot::pp_check(fit, n_draws = 10, seed = 42)
}
```

---

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

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

predicted_samples(model, ...)

## Default S3 method:
predicted_samples(model, ...)
```

```
## S3 method for class 'gam'
predicted_samples(
  model,
  ...,
  data = NULL,
  n_draws = NULL,
  seed = NULL,
  weights = NULL,
  marginalise = NULL
)

## S3 method for class 'scam'
predicted_samples(
  model,
  ...,
  data = NULL,
  n_draws = NULL,
  seed = NULL,
  weights = NULL
)
```

Arguments

model	a fitted model of the supported types; see diagnostic_support.
...	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.
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.
n_draws	positive integer; number of response replicates to generate at the fitted parameters. Required; coefficient chains are not used.
seed	numeric; a random seed for the simulations.
weights	numeric; response sampling weights for GAM methods. An explicit vector overrides the fitting weights; a single value applies to every observation. With data = NULL, the default is the fitted prior weights, except factor-valued binomial responses retain one Bernoulli trial per observation. Native gaulss() ignores fitting weights in its likelihood, so its default response weights are also ones; explicitly supplied sampling weights still override this default. With prediction data, weights default to a vector of ones. For ordinary Gaussian and Gamma responses, larger weights reduce response variance (precision weights). For binomial responses, weights give integer trial counts. They do not alter the fitted coefficient covariance. Include positions restored by na.exclude when supplying weights for fitting rows.
marginalise	NULL retains the usual predictions. TRUE selects all supported re and fs terms; character vectors select exact smooth labels. Fitted predictions integrate response means over new zero-mean Gaussian effects. Response samplers draw

new effects before sampling responses. Smoothing parameters and dispersion remain fixed. Only ordinary `gam()` and `bam()` fits with a scalar response mean are supported.

Details

LSS response simulation uses all fitted distribution parameters at the requested data, with the same parameter transformations and offset rules as `posterior_samples()`. It holds coefficients fixed, so it has no coefficient chains or attached coefficient diagnostics.

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_draws`, indicating which draw each row relates to,
- `.response` (numeric) the predicted response for the indicated row of `data`.

Author(s)

Gavin L. Simpson

See Also

gamm4_support

Examples

```
load_mgcv()

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

predicted_samples(m, n_draws = 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_draws = 5, data = newd, exclude = "s(x2)", seed = 25)

## Exclude s(x1)
predicted_samples(m, n_draws = 5, data = 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_draws = 5, data = newd, seed = 25,
  terms = c("Intercept", "s(x0)", "s(x2)", "s(x3)")
)
```

---

qq_plot *Quantile-quantile plot of model residuals*

Description

Quantile-quantile plots (QQ plots) for fitted models using the reference quantiles of Augustin *et al* (2012), or fixed normal references for randomised quantile residuals following Dunn and Smyth (1996).

Usage

```
qq_plot(model, ...)

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

## S3 method for class 'gam'
qq_plot(
  model,
  method = NULL,
  type = NULL,
  n_uniform = 10,
  n_simulate = 50,
  seed = NULL,
  level = 0.9,
  ylab = NULL,
  xlab = NULL,
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  ci_col = "black",
  ci_alpha = 0.2,
  point_col = "grey20",
  point_alpha = 1,
  line_col = "red",
  by_family = NULL,
  conditioning = NULL,
  ...,
  n_dharma = 250L
)
```

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

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

## S3 method for class 'gamm4'
qq_plot(model, conditioning = NULL, ...)

## S3 method for class 'gjrm'
qq_plot(model, ..., type = "quantile", seed = NULL, response = NULL)

## S3 method for class 'gamm'
qq_plot(model, conditioning = NULL, ...)

## S3 method for class 'gratia_diagnostic_proxy'
qq_plot(model, type = "quantile", ...)

## S3 method for class 'lme'
qq_plot(model, conditioning = NULL, ...)

## S3 method for class 'merMod'
qq_plot(model, conditioning = NULL, ...)

## S3 method for class 'gamlss'
qq_plot(model, ..., type = "quantile", seed = NULL)

## S3 method for class 'sdmTMB'
qq_plot(
  model,
  conditioning = NULL,
  component = "response",
  field_method = c("mle-mvn", "mle-eb", "mle-mvn-integrated"),
  ...
)
```

Arguments

`model`	a fitted model supported by diagnostic_support.
`...`	arguments passed to other methods.
`method`	character or `NULL`; how to calculate reference quantiles. Probability integral transform (PIT) and quantile residuals use fixed uniform and standard-normal references, respectively. For quantile residuals, `"normal"` also selects that fixed reference; other named methods require traditional residuals. For deviance, response and Pearson residuals:

- `"uniform"` applies the fitted inverse cumulative distribution function (CDF)

to randomised uniform quantiles and averages the resulting ordered residuals over `n_uniform` draws.

- `"simulate"` generates `n_simulate` response data sets at the fitting covariates. Responses are generated together when needed to preserve dependence, and their ordered residuals supply reference values and bands.
- `"marginal"` approximates the pooled distribution of individual residuals, without bands for dependence. This is selected for BAM GEE and PQL correlations that do not define a joint response distribution. Explicit joint simulation is unavailable in those cases.
- `"normal"` uses a normal approximation fitted to the residual location and scale. For dependent responses it omits bands based on independence.
- `"direct"` is deprecated in favour of `"uniform"`.

For ordinary GAMs, `NULL` selects `"simulate"` for Tweedie and `"uniform"` otherwise. Missing inverse CDFs cause a switch to simulation; missing response generators can cause a switch to a normal approximation for independent responses. Dependent responses use joint simulation when available, or marginal checks otherwise. For grouped families, `NULL` selects simulation if every component has a generator, or inverse-CDF sampling with a message if every component has an inverse CDF. If neither is available, it gives an error; a normal approximation must be requested explicitly.

`type` character or `NULL`; residual type: `"deviance"`, `"response"`, `"pearson"`, `"pit"`, `"quantile"`, `"dharma"`, or `"surrogate"` (for `ocat()` or `multinom()` GAMs). `NULL` uses the model-specific default: deviance for ordinary lm/glm/GAM fits, and quantile for supported grouped families, GJRM, sdmTMB and full mixed-model checks. glmmTMB uses Pearson residuals conditionally and response residuals marginally. Extracted GAM components follow the ordinary GAM defaults. See diagnostic_support.

`n_uniform` numeric; number of inverse-CDF draws to average when `method = "uniform"`.

`n_simulate` numeric; number of data sets to simulate from the estimated model when using the simulation method (`method = "simulate"`).

`seed` numeric; the random number seed to use for `method = "simulate"` and `method = "uniform"`, and for randomised quantile residuals.

`level` numeric; the coverage level for reference intervals. Must be strictly `0 < level < 1`. Used for quantile reference bands and traditional `method = "simulate"` or `"normal"` bands.

`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	Deprecated; use gratia_theme_get() theme settings instead. Fill colour for reference intervals, including bands for quantile residuals.
ci_alpha	Deprecated; use gratia_theme_get() theme settings instead. Alpha transparency for reference intervals.
point_col	Deprecated; use gratia_theme_get() theme settings instead. Constant point colour when family components are not mapped to colour. Pooled grouped diagnostics instead colour points by component.
point_alpha	Deprecated; use gratia_theme_get() theme settings instead. Alpha transparency of points on the QQ plot.
line_col	Deprecated; use gratia_theme_get() theme settings instead. Colour used to draw the reference line.
by_family	logical or NULL; separate indexed grouped-family components into facets. The default pools quantile residuals with component colours and facets traditional residuals. TRUE and FALSE explicitly select the layout. In appraise(), observed-versus-fitted values always use component facets. Standalone observed_fitted_plot() facets by default and permits explicit pooling. Ignored for ordinary models.
conditioning	character or NULL; "conditional" holds fitted random effects fixed, "marginal" averages over new effects, and "gam_component" uses the extracted GAM. Availability and defaults depend on the model; see diagnostic_support.
n_dharma	positive integer; number of response simulations for type = "dharma". This is separate from n_simulate, which controls traditional QQ reference calculations. Defaults to 250.
response	NULL uses all responses; a single integer selects one. For mgcv::gfam() models, the integer indexes the family list and selects observations assigned to that component, including when families repeat. For joint GJRM models, it selects a response in model order. Selection is independent of by_family.
component	character; for delta sdmTMB models, select the complete "response", occurrence ("binomial"), or "positive" component. Positive-component checks use only observations with positive responses.
field_method	how to treat fitted sdmTMB fields: hold one posterior draw fixed ("mle-mvn"), hold estimates fixed ("mle-eb"), or average over fitted-field uncertainty ("mle-mvn-integrated"). See quantile_residuals() and diagnostic_support.

Details

See appraise() for grouped-family defaults, supported distributions, pooling and separate component panels. PIT and quantile residuals use fixed uniform and standard-normal reference quantiles. For ideal independent responses, bands are pointwise intervals for ordered residuals. For dependent responses, bands use simulations that preserve dependence when available; otherwise they are omitted. Residuals calculated from simulation ranks also use simulated bands. See diagnostic_support for these choices and limits. Bands condition on fitted parameters and exclude fitting uncertainty; they do not provide simultaneous coverage. Traditional residuals use the fitted response distributions or a normal approximation.

For mgcv::ocat() fits, type = "surrogate" uses logistic reference quantiles with location zero and scale one. Set method = NULL. Independent observations have approximate pointwise bands

obtained by applying `qlogis()` to beta order-statistic limits. Worm plots subtract the same logistic reference quantiles from residuals and bands. These references treat fitted parameters as fixed. For `mgcv::multinom()` GAMs, surrogate residuals retain a utility component for every category, including 0. Components have separate facets and centred Gumbel reference quantiles, including when `by_family = FALSE`. The same reference is subtracted in worm plots, and pointwise beta order-statistic bands use the centred Gumbel quantile function separately for each component. Ordinary and scalable displays share their draw. See `surrogate_residuals()`.

Subtitles name the method. Captions identify conditioning and field choices, joint simulation bands or omitted bands. Routine pointwise bands are unlabelled. Supply `caption` to replace this text, or `caption = ""` to omit it. Ordinary and scalable QQ and worm plots use the same labels.

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"`.

References

The underlying methodology used when `method` is `"simulate"` or `"uniform"` is described in Augustin *et al* (2012):

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

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

See Also

mgcv::qq.gam for more details on the methods used.

gamm4_support

Examples

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

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

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

```
qq_plot(m,
  method = "simulate",
  seed = 42,
  point_col = "steelblue",
  point_alpha = 0.4
)

## ... or request a normal approximation for the residuals.
qq_plot(m, method = "normal")
```

`qq_plot.glmmTMB` *Effects and diagnostics for glmmTMB models*

Description

Evaluate and plot the stored mgcv smooths in a glmmTMB fit. Coefficients and their joint covariance are transformed back to the constrained mgcv basis. Intervals use TMB's joint covariance, including fixed–random covariance; they are not mgcv frequentist or smoothness-corrected intervals.

Usage

```
## S3 method for class 'glmmTMB'
qq_plot(model, ..., method = NULL, type = NULL, conditioning = NULL)

## S3 method for class 'glmmTMB'
worm_plot(model, ..., method = NULL, type = NULL, conditioning = NULL)

## S3 method for class 'glmmTMB'
appraise(model, ..., method = NULL, type = NULL, conditioning = NULL)

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

## S3 method for class 'glmmTMB'
parametric_effects(
  object,
  terms = NULL,
  ...,
  unconditional = FALSE,
  frequentist = FALSE
)

## S3 method for class 'glmmTMB'
smooths(object, ...)

## S3 method for class 'glmmTMB'
get_smooths_by_id(object, id, ...)
```

```
## S3 method for class 'glmmTMB'
smooth_estimates(object, ..., unconditional = FALSE, frequentist = FALSE)

## S3 method for class 'glmmTMB'
smooth_coefs(object, select, ...)

## S3 method for class 'glmmTMB'
assemble(
  object,
  ...,
  parametric = NULL,
  residuals = FALSE,
  unconditional = FALSE,
  frequentist = FALSE,
  ranef_lines = FALSE,
  envir = NULL
)

## S3 method for class 'glmmTMB'
draw(
  object,
  ...,
  parametric = NULL,
  residuals = FALSE,
  unconditional = FALSE,
  frequentist = FALSE,
  ranef_lines = FALSE,
  envir = NULL
)
```

Arguments

`...`	Arguments passed to the corresponding gratia method.
`method`	QQ method; `NULL` selects `"simulate"` for traditional residuals.
`type`	Residual type: `"pearson"`, `"response"`, `"deviance"`, `"pit"`, `"quantile"`, or `"dharma"`, when available for the family. The default is Pearson for conditional checks and response residuals for marginal checks.
`conditioning`	character or `NULL`; `"conditional"` (default) or `"marginal"`; see diagnostic_support.
`object, model`	A fitted `glmmTMB` model.
`terms`	Character vector selecting parametric terms, as returned by `parametric_terms()`. Later predictors append `.1`, `.2`, etc.
`unconditional, frequentist`	Must be `FALSE`; mgcv-specific covariance alternatives are not available.
`id`	Integer positions of smooths across all linear predictors.
`select`	Smooth or native random-effect coefficient selection, across all linear predictors. Numeric indices select smooths first, then native random coefficients. Native

	labels have the form `(1 | g)` and `(x | g)`; later predictors append `.1`, `.2`, etc. `smooth_estimates()` and `smooth_coefs()` select smooths only.
`parametric`	Include parametric effect plots? `NULL` follows `draw.gam()`: include them unless `select` is supplied. Use `terms` to choose parametric terms, and `parametric = FALSE` to omit them.
`residuals`	Must be `FALSE`; partial residual overlays are not supported.
`ranef_lines`	Logical; replace QQ plots for a simple native random intercept and numeric slope term, `(1 + x | g)`, with one panel of lines showing `b[0,g] + x * b[1,g]`? Defaults to `FALSE`. Both coefficients must be selected. Multiple slopes, interactions, factor slopes, transformed slopes and structured covariance terms retain separate QQ plots.
`envir`	Optional environment for evaluating model covariates.

Details

Smooth support requires stored smooth and random-effect metadata. Unsupported multi-penalty structures, mapped joint parameters and rank-deficient smooth models give errors. Factor-by and tensor smooth availability is determined by glmmTMB; gratia does not expand the smooth types glmmTMB can fit. Use `draw(model, surface = "3d", ci_3d = TRUE)` for interval surfaces on smooths and numeric parametric interactions in any linear predictor. `ci_3d_alpha` and `ci_3d_col` control their opacity and fill colour. Pass `cyclic = "polar"` and optional `polar_args` to `draw()` or `assemble()` for polar plots of supported one-dimensional cyclic smooths. See `draw.gam()` for the coordinate options.

Parametric effects use fitted contrast coding and exclude smooth null-space coefficients. Their intervals include uncertainty in the intercept of their own linear predictor by default, as for GAMs. `overall_uncertainty = FALSE` gives term-only intervals. Models without smooths are also supported.

Native random effects use one Gaussian QQ panel per coefficient type and grouping factor, matching mgcv random-effect smooth plots. The internal random effects representing smooths are excluded from these panels. With `ranef_lines = TRUE`, eligible intercept-slope pairs use a common grid over the fitting range of the numeric covariate, or its supplied `data` values. Lines show random contributions only, without population-level terms or uncertainty bands. Different formula terms are never combined.

Diagnostics use native Pearson residuals by default. QQ simulations hold fitted smooths and grouping effects fixed and draw new responses; no refitting is performed. Supported response families are Gaussian, Poisson, binomial, nbinom1/2, beta, Gamma/ziGamma, Tweedie, lognormal, and zero-truncated Poisson/nbinom1/2, including applicable zero-inflation or hurdle mixtures and varying dispersion. Truncated nbinom1 currently requires `type = "response"` because glmmTMB does not supply its Pearson residuals. Other unavailable residual types fail explicitly. Supported fitted response distributions also provide randomized PIT/quantile residuals and inverse-CDF QQ references, including their zero-inflation and truncation probabilities. Posterior predictive checks draw coefficients and fitted random effects together from their approximate posterior. See `posterior_predictive_plot()` for how covariance and other family parameters are treated. Marginal checks simulate new effects using glmmTMB's response generator. If marginal CDFs are not analytic, residual checks use a shared sample of simulations to calculate moments and ranks. See diagnostic_support for conditioning choices and remaining limitations.

Value

Smooth estimates, coefficients, or plots, as for the GAM methods. `assemble()` returns a named list of ggplot objects, of class `"assembled_plots"`. `draw()` combines these plots into a patchwork unless `wrap = FALSE`, which returns the individual plots instead. All smooths, native random effects and parametric terms are plotted by default, as for `draw.gam()`. Smooths in later linear predictors use mgcv-style labels such as `s.1(x)` and `s.2(x)`. Predictor order is conditional, zero-inflation (if present), then dispersion (if present). Use `smooths(object)` and `select` to select individual smooths. For models with multiple linear predictors, smooth and parametric subtitles identify the "Conditional model", "Zero-inflation model" or "Dispersion model", consistently with native random-effect panels. Constant dispersion alone does not trigger these subtitles. Saved `parametric_effects()` retain the labels in an optional `.model` column. `names(assemble(object, parametric = FALSE))` also lists native random-effect coefficient panels.

Label layout

Pass `label_control = draw_label_control(...)` to `draw()` to wrap discrete axis labels and legend text, stagger discrete tick labels, or adjust legend and colour-bar layout. Settings apply to each panel, including unwrapped plot lists. See `draw_label_control()` for the available settings.

Examples

```
if (requireNamespace("glmmTMB", quietly = TRUE)) {
  load_mgcv()
  # Simulate a nonlinear mean with Gaussian noise
  set.seed(42)
  d <- data.frame(x = runif(150))
  d$y <- sin(6 * d$x) + rnorm(150, sd = 0.2)

  # Fit and draw a smooth using glmmTMB
  m <- glmmTMB::glmmTMB(y ~ s(x, k = 8), data = d, REML = TRUE)
  n_smooths(m)
  draw(m)
}
```

`quantile_residuals` *PIT and randomised quantile residuals*

Description

Map observations to probabilities under the fitted response distribution. These probability integral transform (PIT) residuals have a uniform reference distribution under a correctly specified continuous model with known parameters. Quantile residuals transform them to standard-normal scores. Discrete observations require randomisation over their probability mass.

Usage

```
## S3 method for class 'gamm4'
quantile_residuals(
  model,
  type = c("pit", "quantile"),
  seed = NULL,
  conditioning = NULL,
  ...
)

## S3 method for class 'gamlss'
quantile_residuals(model, type = c("pit", "quantile"), seed = NULL, ...)

## S3 method for class 'gjrm'
quantile_residuals(
  model,
  type = c("pit", "quantile"),
  seed = NULL,
  response = NULL,
  ...
)

## S3 method for class 'gamm'
quantile_residuals(
  model,
  type = c("pit", "quantile"),
  seed = NULL,
  conditioning = NULL,
  ...
)

## S3 method for class 'gratia_diagnostic_proxy'
quantile_residuals(
  model,
  type = c("pit", "quantile"),
  seed = NULL,
  conditioning = NULL,
  ...
)

## S3 method for class 'lme'
quantile_residuals(
  model,
  type = c("pit", "quantile"),
  seed = NULL,
  conditioning = NULL,
  ...
)
```

```
## S3 method for class 'merMod'
quantile_residuals(
  model,
  type = c("pit", "quantile"),
  seed = NULL,
  conditioning = NULL,
  ...
)

quantile_residuals(
  model,
  type = c("pit", "quantile"),
  seed = NULL,
  ...,
  response = NULL
)

## S3 method for class 'gam'
quantile_residuals(
  model,
  type = c("pit", "quantile"),
  seed = NULL,
  conditioning = NULL,
  ...,
  response = NULL
)

## S3 method for class 'glm'
quantile_residuals(model, type = c("pit", "quantile"), seed = NULL, ...)

## S3 method for class 'lm'
quantile_residuals(model, type = c("pit", "quantile"), seed = NULL, ...)

## Default S3 method:
quantile_residuals(
  model,
  type = c("pit", "quantile"),
  seed = NULL,
  conditioning = NULL,
  ...
)

## S3 method for class 'glmmTMB'
quantile_residuals(
  model,
  type = c("pit", "quantile"),
  seed = NULL,
```

```
  conditioning = NULL,
  ...
)

## S3 method for class 'gratia_glmmtmb_diagnostic'
quantile_residuals(
  model,
  type = c("pit", "quantile"),
  seed = NULL,
  conditioning = NULL,
  ...
)

## S3 method for class 'sdmTMB'
quantile_residuals(
  model,
  type = c("pit", "quantile"),
  seed = NULL,
  conditioning = NULL,
  component = "response",
  field_method = c("mle-mvn", "mle-eb", "mle-mvn-integrated"),
  ...
)
```

Arguments

`model`	a fitted model object.
`type`	character; `"pit"` returns probabilities and `"quantile"` returns standard-normal scores.
`seed`	integer or `NULL`; random seed for randomised residuals. With `NULL`, calculations use the current random-number state. An explicit seed preserves the caller's state, including when none was previously initialised.
`conditioning`	character or `NULL`; hold fitted random effects fixed (`"conditional"`), average over new effects (`"marginal"`), or use the extracted GAM (`"gam_component"`), where supported. See diagnostic_support.
`...`	arguments passed to other methods.
`response`	`NULL` uses all responses; a single integer selects one. For `mgcv::gfam()` models, the integer indexes the family list and selects observations assigned to that component. For joint GJRM models, it selects a response in model order. All-response GJRM residuals form a matrix with one column per response; selecting one returns a vector. Grouped-family residuals are always vectors.
`component`	character; which delta sdmTMB component to check; see `appraise()`.
`field_method`	For sdmTMB diagnostics, `"mle-mvn"` (the default) follows sdmTMB's recommended residual calculation. Draw the fitted random effects together once from their approximate posterior, then use that same draw to evaluate the response CDF and generate every reference response. Fixed coefficients and covariance

parameters remain at their maximum likelihood estimates. The independent seed in `gratia.diagnostics.field_seed` (default

1. keeps different diagnostic panels on the same draw of the effects. `"mle-eb"` uses the fitted empirical-Bayes effects without drawing their uncertainty; sdmTMB does not recommend this mode for goodness-of-fit testing. `"mle-mvn-integrated"` draws new values of the fitted effects for each simulation, from their approximate posterior. Residuals use the rank of each response among the simulated reference responses. This averages over uncertainty in the fitted effects, but reuses the data used in fitting and can make goodness-of-fit checks less likely to detect departures. These residuals differ from sdmTMB's calculation at a single draw of the effects. With `conditioning = "marginal"`, new spatial, temporal and group effects are drawn from their fitted prior distributions, while smooth effects remain at fitted values; `field_method` does not add another draw of the fields. These approximations do not guarantee that goodness-of-fit tests have their intended false-positive rate after parameter estimation. Posterior predictive checks separately include coefficient and field uncertainty; see `posterior_predictive_plot()`. Student-t responses use the fitted location, scale and fixed or estimated degrees of freedom. The scale is not the response standard deviation. Pearson residuals require degrees of freedom greater than two, because their response variance is otherwise infinite. PIT, quantile residuals and posterior predictive checks remain available when this variance is infinite.

Details

With `na.exclude`, excluded observations are restored as `NA` residuals; with `na.omit`, only observations used for fitting are returned. Selecting a grouped-family component with `response` returns only its retained fitting observations, without placeholders for excluded rows whose component membership is unavailable.

For `mgcv::cnorm()`, `mgcv::clog()` and `mgcv::cpois()` models, censored observations have PIT residuals sampled uniformly between `F(l)` and `F(u)`. Here `l` and `u` bound the censoring interval and `F` is the fitted cumulative distribution function (CDF) of the underlying response. Left and right censoring use probabilities zero and one for the unbounded end, respectively. Quantile residuals apply `qnorm()` to these PIT values. Uncensored continuous observations use `F(y)`; uncensored Poisson observations are randomised between `F(y - 1)` and `F(y)`. Use non-integer censoring limits for `cpois()`, as recommended by mgcv.

For grouped-family GAMs fitted with `mgcv::gfam()`, each observation uses its component's CDF, fitted nuisance parameters, dispersion and weights. Residuals are returned in fitting order. `response = NULL` uses all components; an integer selects observations assigned to that family-list position. Repeated families retain distinct indices. With an explicit seed, selected residuals match the corresponding rows of an all-component call whenever all components support the calculation. All selected active components must supply supported CDFs and probability limits for discrete or censored observations. Unsupported components give an error; `appraise()` can instead select deviance residuals automatically. See `appraise()` for supported families. Tweedie zeros have positive probability mass: their PITs are randomised between `F(0-) = 0` and `F(0)`. Positive observations use `F(y)`.

When a CDF can calculate log probabilities in both tails, quantile residuals use the log probability in the smaller tail directly. This avoids infinities caused by rounding probabilities to zero or one.

Probabilities are not clipped, so true zero-probability tails still give infinite residuals for ordinary models; grouped diagnostics report non-finite residuals as errors. PITs are returned as ordinary probabilities and may still round to zero or one in extreme tails. Applying `qnorm()` to returned PITs can therefore be less accurate than requesting quantile residuals directly. CDFs without log probabilities retain ordinary-probability calculations.

For `mgcv::multinom()` models, probabilities accumulate in the existing category coding order `0, ..., K`. If category `j` has probability `p_j`, its PIT is `F(j - 1) + V * p_j`, where `V` is uniform on `(0, 1)`. This uses one categorical response, rather than a vector of trial counts. The coding does not change the multinomial likelihood into an ordinal model. Each fixed enumeration gives a uniform randomized transform under the true model, but changing the enumeration can change individual residuals and which departures from the model are easiest to detect.

Marginal checks may calculate residuals from ranks in joint response simulations when no analytic marginal CDF is available. See diagnostic_support for the rank definition, conditioning choices and limitations after fitting.

See Also

gamm4_support

`random_effect_samples` *Draw new random effects from an estimated GAM*

Description

Sample zero-mean Gaussian random effects using the fitted smoothing parameters and dispersion. These are new effects, rather than posterior draws of the effects estimated for the fitting groups.

Usage

```
random_effect_samples(model, data = NULL, select = NULL, n_draws, seed = NULL)
```

Arguments

`model`	An ordinary `mgcv::gam()` or `mgcv::bam()` fit with one linear predictor.
`data`	Evaluation data. With `NULL`, use the fitting observations. Supplied data need only contain predictors of the selected smooths. Group labels may be new; repeated labels share an effect within a draw.
`select`	Exact smooth labels, or `NULL` for all `re` and `fs` terms.
`n_draws`	Positive integer giving the number of draws.
`seed`	Optional integer seed. An explicit seed preserves the caller's random-number state; `NULL` advances the current state.

Details

Standard `re` random intercepts, slopes and factor interactions, and `fs` random curves are supported. Each curve is drawn jointly over its evaluation points, including the components penalising its constant and linear parts. Custom `re` penalties supplied through `xt$S` are unsupported. Fits using `min.sp` are also unsupported: mgcv does not retain the fitted lower bounds needed to reconstruct their random-effect covariance.

For stored penalties S_j, smoothing parameters λ_j, and dispersion ϕ, coefficient covariance is

$$\phi(\sum_j \lambda_j S_j)^{-1}.$$

Smoothing parameters and dispersion remain fixed. As in `mgcv::gam.vcomp()`, random-effect covariance uses `reml.scale` when it is finite and `sig2` otherwise. Response generators retain their usual `sig2` dispersion. Different smooths and different groups have independent new coefficients, as specified by these model terms. Covariance between fitted coefficients is not used here.

Use `marginalise` in `fitted_values()` or `fitted_samples()` to integrate response means over these distributions. Use it in `predicted_samples()`, `posterior_samples()` or `simulate.gam()` to simulate new-group responses.

Value

A tibble of class `random_effect_samples`, with `.row`, `.draw`, `.smooth`, `.value` (the link-scale contribution), and covariates. The `random_effect_info` attribute describes the fixed covariance estimates.

Examples

```
load_mgcv()
m <- gam(travel ~ s(Rail, bs = "re"), data = nlme::Rail,
  method = "REML")
random_effect_samples(m, n_draws = 3, seed = 42)

# Compare zero random effects with response means averaged over groups.
set.seed(2)
d <- expand.grid(x = seq(0, 1, length.out = 15), id = factor(1:12))
d$y <- rpois(nrow(d), exp(0.3 + d$x + rnorm(12, sd = 0.6)[d$id]))
p <- gam(y ~ x + s(id, bs = "re"), data = d, family = poisson(),
  method = "REML")
nd <- data.frame(x = c(0, 0.5, 1), id = d$id[1])
fitted_values(p, data = nd, exclude = "s(id)")
fitted_values(p, data = nd, marginalise = TRUE)
fitted_values(p, data = nd, marginalise = TRUE,
  integration = "monte_carlo", seed = 42)
nd$id <- c("new", "new", "other")
predicted_samples(p, data = nd, marginalise = TRUE, n_draws = 3, seed = 42)
```

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 factor with the selected values and the same levels, level order and ordered status as fct. ref_level() returns the first level.

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.

residual_correlation *Ordered residual correlation diagnostics*

Description

Compute lag correlations within independent series. These are descriptive checks of ordered residual structure, with no independence-based confidence bands or significance tests.

Usage

```
residual_correlation(
  model,
  type = NULL,
  lag_max = 20L,
  series = NULL,
  seed = NULL,
  ...,
  n_dharma = 250L
)

residual_correlation_plot(
  model,
  type = NULL,
```

```
  lag_max = 20L,
  series = NULL,
  seed = NULL,
  ...,
  n_dharma = 250L
)
```

Arguments

model	a supported fitted model.
type	residual type. For an AR1 BAM the default is "standardized", using mgcv's stored std.rsd. Otherwise the default is "quantile". BAM also supports "working", using weighted residuals from the final iteratively reweighted least-squares fit, and "working.standardized", adjusting those residuals for the fitted AR1 correlation (whitening). Extended families are supported if they supply the expected deviance derivatives required for these working residuals.
lag_max	positive integer; maximum observation lag.
series	optional vector identifying series, in the order of observations used for fitting or of the original observations. BAM's AR.start boundaries are always retained, even if series is supplied. Other fitters need an explicit series and appropriately ordered fitting data when separate time series are present.
seed	optional seed for randomized residuals.
...	arguments used to prepare the model for diagnostics, including conditioning and, for sdmTMB, field_method.
n_dharma	positive integer; number of response simulations for type = "dharma". This is separate from n_simulate, which controls traditional QQ reference calculations. Defaults to 250.

Details

Use type = "dharma" and n_dharma for simulation-based PIT residuals on the uniform scale; see dharma_residuals(). Use type = "surrogate" for ocat() models to calculate correlations of residuals on the latent logistic scale; see surrogate_residuals().

BAM's generalized estimating equations (GEE) AR1 correction acts on the weighted working response used for fitting. Whitening these working residuals can reveal remaining serial structure. The stored std.rsd in current mgcv adjusts its final residuals for AR1 correlation, which need not be the weighted working residuals for non-Gaussian models. Both checks are approximate. They can reveal remaining serial structure, but do not specify a joint response distribution, establish normality or provide calibrated goodness-of-fit tests. Fitted parameters, leverage and discrete likelihoods affect correlations.

Lags use observation positions within series, not elapsed time or spatial distance. Means are removed within each series, and correlations use only finite pairs at each lag; missing residuals and omitted fitting rows do not close gaps. If series are interleaved, supply series for all original observations so omitted rows can be assigned to the right series. If series covers only observations used for fitting, an omitted position is counted as a gap in every series with observations on both

sides of it. Observations assigned to another series never create gaps. Significance bands that assume independent residuals are omitted. Check spatial residual patterns with their coordinates and a suitable distance diagnostic.

Value

`residual_correlation()` returns a tibble with lag, correlation and the number of finite observation pairs used at each lag. `residual_correlation_plot()` returns a ggplot.

See Also

diagnostic_support, `qq_plot()`

`residuals_covariate_plot` *Plot residuals against covariates*

Description

Inspect residuals from the complete fitted model against individual covariates or pairs of numeric covariates.

Usage

```
residuals_covariate_plot(
  model,
  covariates = NULL,
  terms = NULL,
  pairs = NULL,
  data = NULL,
  by = NULL,
  type = NULL,
  smooth = NULL,
  bins = c(20L, 20L),
  min_n = 5L,
  statistic = c("mean", "sd"),
  seed = NULL,
  conditioning = NULL,
  n_dharma = 250L,
  point_col = "grey20",
  point_alpha = 0.5,
  line_col = "red",
  xlab = NULL,
  ylab = NULL,
  title = NULL,
  subtitle = NULL,
  caption = NULL,
```

```
  ncol = NULL,
  ...
)

scalable_residuals_covariate_plot(
  model,
  covariates = NULL,
  terms = NULL,
  pairs = NULL,
  data = NULL,
  by = NULL,
  type = NULL,
  smooth = NULL,
  bins = c(20L, 20L),
  min_n = 5L,
  statistic = c("mean", "sd"),
  seed = NULL,
  conditioning = NULL,
  n_dharma = 250L,
  density_bins = c(50L, 50L),
  density = c("conditional", "count"),
  density_method = c("kde", "histogram"),
  bandwidth = NULL,
  line_col = "red",
  xlab = NULL,
  ylab = NULL,
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  ncol = NULL,
  ...
)
```

Arguments

`model`	a fitted model supported by diagnostic_support, or prepared diagnostic data. The original model must retain its fitting observations.
`covariates`	character vector of exact covariate names, in plotting order. `NULL` selects model covariates, excluding variables used only as offsets or random-effect groups. `character()` selects no individual panels, for use with `pairs`.
`terms`	character vector of exact mgcv smooth labels, as returned by `smooths()`. For example, use `"te(x,z)"`, not `"te(x, z)"`. Parametric terms, partial matches and numeric or logical selectors are not accepted. Selected smooths supply their original covariates and by variables. Cannot be combined with a non-`NULL` `covariates`.
`pairs`	list of character vectors, each naming two distinct numeric covariates. Adds a joint residual map for each pair, in the supplied order.

data	original fitting data or retained fitting observations, including model predictors needed to verify alignment. Can contain additional covariates absent from the model. Rows are matched to stored observations and their evaluated predictors checked. Row counts alone do not establish alignment. NULL uses stored data, recovering raw covariates from the fitting call only when needed and verifying them in the same way.
by	optional name of a categorical covariate defining separate facets.
type	character or NULL; residual type: "deviance", "pearson", "response", "pit", "quantile", "dharma", or "surrogate" (for ocat() or multinom() GAMs). Grouped models automatically use quantile residuals when all active components are supported, otherwise all use deviance residuals with a message. Ordinary models retain deviance residuals by default. See appraise() for distributional support.
smooth	logical or NULL; add a descriptive penalised smooth of the mean residual to numeric panels. The default, NULL, fits a trend only for groups with at most 20,000 eligible observations. TRUE fits a trend regardless of group size; FALSE omits all trends. No confidence bands or tests are computed.
bins	positive integer or pair of integers giving the numbers of cells along the two covariate axes of joint maps.
min_n	positive integer; joint-map cells with fewer observations are blank. Counts for occupied cells are retained in the plotting data.
statistic	joint-map summary: "mean" or sample standard deviation, "sd". Cells containing one observation have undefined standard deviation.
seed	integer; random seed to use for PIT, quantile, surrogate or DHARMa-style residuals.
conditioning	character or NULL; "conditional" holds fitted random effects fixed, "marginal" averages over new effects, and "gam_component" uses the extracted GAM. Availability and defaults depend on the model; see diagnostic_support.
n_dharma	positive integer; number of simulations for DHARMa residuals.
point_col	Deprecated; use gratia_theme_get() theme settings instead. Colour for individual residual points or observation locations.
point_alpha	Deprecated; use gratia_theme_get() theme settings instead. Numeric; alpha transparency for points in plots.
line_col	Deprecated; use gratia_theme_get() theme settings instead. Colour for reference lines and categorical group means.
xlab	character or expression; the label for the x axis. If not supplied, a suitable label will be generated.
ylab	character or expression; the label for the y axis. If not supplied, a suitable label will be generated.
title	character or expression; the title for the plot. See ggplot2::labs().
subtitle	character or expression; the subtitle for the plot. See ggplot2::labs().
caption	character or expression; the plot caption. See ggplot2::labs().
ncol	optional number of columns when composing several panels.

`...`	arguments used to prepare model diagnostics, such as sdmTMB `component` and `field_method`. For grouped-family GAMs and joint GJRM models, `response = NULL` uses every response and an integer selects one; see `appraise()`. Selection leaves `by_family` independent.
`density_bins`	Positive integer or pair of integers giving the number of horizontal and vertical heatmap bins.
`density`	One of `"conditional"` or `"count"`. Conditional density is estimated using `density_method`. With `"histogram"`, it is the count divided by the horizontal-bin total and vertical-bin width. Empty horizontal bins are blank. Counts are always unsmoothed.
`density_method`	Conditional-density estimator: `"kde"` (the default) uses `KernSmooth::bkde2D()` with Gaussian kernels; `"histogram"` uses unsmoothed rectangular bins. Only affects the residual heatmap.
`bandwidth`	Optional pair of positive bandwidths for the predictor and residual axes, in their plotted units. `NULL` uses normal-reference bandwidths, adjusted for two dimensions, with a fallback for constant axes. Used only for `density_method = "kde"` and conditional density.

Details

Covariates are plotted in their original units. A term such as `s(log(x))` selects `x`; stored transformed coordinates are never inverted to recover original values. Supply `data` if those values are unavailable. Term selection does not plot fitted term contributions or partial residuals. A factor-specific smooth selects covariates without restricting observations; use by to inspect patterns separately among groups.

Pearson residuals are useful for checking the mean function because they account for the specified mean–variance relationship. Response residuals retain the response units. A remaining curve can suggest an incorrectly specified mean function, including insufficient flexibility in a smooth. Quantile and PIT residuals are useful for checking the conditional response distribution, including changing spread and tails. Non-Gaussian deviance residuals need not have zero conditional mean, so their mean curve need not follow the zero reference line even under the fitted model.

Numeric panels show points, or conditional-density heatmaps in the scalable version. Categorical panels show boxplots and group means; the ordinary version also shows points. Joint maps show means or standard deviations within rectangular cells. The ordinary version also shows observation locations. Mean maps use a diverging colour scale centred on zero, or 0.5 for PIT and DHARMa residuals. Standard-deviation maps use a sequential scale. These are descriptive summaries, not tests against simulated reference maps. An omitted interaction can produce a joint pattern even when both individual covariate panels appear flat. Correlated covariates can share residual patterns; use the plots to guide model revisions and subsequent checks.

All eligible observations are used. Mean trends use a shrinkage cubic spline with up to 40 basis functions, fitted with GCV below 10,000 observations and discrete fREML via `mgcv::bam()` otherwise. The same calculation is used by both plotting functions. Trends are omitted with fewer than four distinct covariate values. Summaries and trends are calculated separately within each response, family component and by group. Missing plotting values are reported and omitted separately for each requested plot.

Value

A ggplot for one panel, or a patchwork composition for several. Scalable plots retain only summaries and evaluated trend curves.

See Also

`appraise()`, `residuals_linpred_plot()`, `partial_residuals()`

Examples

```
load_mgcv()
dat <- data_sim("eg1", n = 400, seed = 2)
m <- gam(y ~ x2 + s(x0) + s(x1), data = dat, method = "REML")
# A curve against x2 suggests reconsidering its linear effect.
residuals_covariate_plot(m, covariates = "x2", type = "pearson")
# Check a covariate omitted from the model.
residuals_covariate_plot(m, covariates = "x3", data = dat)
# Inspect individual covariates and possible remaining interaction structure.
scalable_residuals_covariate_plot(m, covariates = c("x0", "x2"),
  pairs = list(c("x0", "x2")), bins = 6, type = "quantile", seed = 42)
```

---

residuals_hist_plot *Histogram of model residuals*

Description

Histogram of model residuals

Usage

```
residuals_hist_plot(
  model,
  type = NULL,
  n_bins = c("sturges", "scott", "fd"),
  ylab = NULL,
  xlab = NULL,
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  seed = NULL,
  by_family = NULL,
  conditioning = NULL,
  ...,
  n_dharma = 250L
)
```

Arguments

model	a fitted model supported by diagnostic_support.
type	character or NULL; residual type: "deviance", "pearson", "response", "pit", "quantile", "dharma", or "surrogate" (for ocat() or multinom() GAMs). Grouped models automatically use quantile residuals when all active components are supported, otherwise all use deviance residuals with a message. Ordinary models retain deviance residuals by default. See appraise() for distributional support.
n_bins	character or numeric; either the number of bins or a string indicating how to calculate the number of bins.
ylab	character or expression; the label for the y axis. If not supplied, a suitable label will be generated.
xlab	character or expression; the label for the x axis. If not supplied, a suitable label will be generated.
title	character or expression; the title for the plot. See ggplot2::labs().
subtitle	character or expression; the subtitle for the plot. See ggplot2::labs().
caption	character or expression; the plot caption. See ggplot2::labs().
seed	integer; random seed to use for PIT, quantile, surrogate or DHARMa-style residuals.
by_family	logical or NULL; separate indexed grouped-family components into facets. The default pools quantile residuals with component colours and facets traditional residuals. TRUE and FALSE explicitly select the layout. In appraise(), observed-versus-fitted values always use component facets. Standalone observed_fitted_plot() facets by default and permits explicit pooling. Ignored for ordinary models.
conditioning	character or NULL; "conditional" holds fitted random effects fixed, "marginal" averages over new effects, and "gam_component" uses the extracted GAM. Availability and defaults depend on the model; see diagnostic_support.
...	optional arguments passed to model preparation, such as the sdmTMB field_method. For grouped-family GAMs and joint GJRM models, response = NULL uses every response and an integer selects one; see appraise(). Selection leaves by_family independent.
n_dharma	positive integer; number of response simulations for type = "dharma". This is separate from n_simulate, which controls traditional QQ reference calculations. Defaults to 250.

Details

Pooled grouped-family histograms use common bin edges and component-specific densities, with faint group-coloured fills and opaque stepped upper boundaries. Set by_family = TRUE for separate panels.

`residuals_linpred_plot` *Plot of residuals versus linear predictor values*

Description

Plot of residuals versus linear predictor values

Usage

```
residuals_linpred_plot(
  model,
  type = NULL,
  ylab = NULL,
  xlab = NULL,
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  point_col = "grey20",
  point_alpha = 1,
  line_col = "red",
  seed = NULL,
  by_family = NULL,
  rank = NULL,
  conditioning = NULL,
  ...,
  n_dharma = 250L
)
```

Arguments

`model`	a fitted model supported by diagnostic_support.
`type`	character or `NULL`; residual type: `"deviance"`, `"pearson"`, `"response"`, `"pit"`, `"quantile"`, `"dharma"`, or `"surrogate"` (for `ocat()` or `multinom()` GAMs). Grouped models automatically use quantile residuals when all active components are supported, otherwise all use deviance residuals with a message. Ordinary models retain deviance residuals by default. See `appraise()` for distributional support.
`ylab`	character or expression; the label for the y axis. If not supplied, a suitable label will be generated.
`xlab`	character or expression; the label for the x axis. If not supplied, a suitable label will be generated.
`title`	character or expression; the title for the plot. See `ggplot2::labs()`.
`subtitle`	character or expression; the subtitle for the plot. See `ggplot2::labs()`.
`caption`	character or expression; the plot caption. See `ggplot2::labs()`.

point_col	Deprecated; use gratia_theme_get() theme settings instead. Constant point colour when components are not mapped to colour. Pooled grouped diagnostics instead colour points by component.
point_alpha	Deprecated; use gratia_theme_get() theme settings instead. Numeric; alpha transparency for points in plots.
line_col	Deprecated; use gratia_theme_get() theme settings instead. Colour specification for 1:1 line.
seed	integer; random seed to use for PIT, quantile, surrogate or DHARMa-style residuals.
by_family	logical or NULL; separate indexed grouped-family components into facets. The default pools quantile residuals with component colours and facets traditional residuals. TRUE and FALSE explicitly select the layout. In appraise(), observed-versus-fitted values always use component facets. Standalone observed_fitted_plot() facets by default and permits explicit pooling. Ignored for ordinary models.
rank	logical or NULL; use within-component ranks on the horizontal axis of residual-versus-predictor plots. The default ranks pooled grouped plots and retains original predictors in facets. TRUE or FALSE overrides this independently of the residual type and layout. Only finite observations included in each component determine its ranks; average ties are used.
conditioning	character or NULL; "conditional" holds fitted random effects fixed, "marginal" averages over new effects, and "gam_component" uses the extracted GAM. Availability and defaults depend on the model; see diagnostic_support.
...	optional arguments passed to model preparation, such as the sdmTMB field_method. For grouped-family GAMs and joint GJRM models, response = NULL uses every response and an integer selects one; see appraise(). Selection leaves by_family independent.
n_dharma	positive integer; number of response simulations for type = "dharma". This is separate from n_simulate, which controls traditional QQ reference calculations. Defaults to 250.

Details

The default subtitle identifies the response family. Conditioning and field choices appear in the caption where relevant. Supply caption to replace this text, or caption = "" to omit it. The scalable version uses the same labels.

---

response_derivatives *Derivatives on the response scale from an estimated GAM*

Description

Derivatives on the response scale from an estimated GAM

Usage

```
response_derivatives(object, ...)

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

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

## S3 method for class '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 = NULL,
  n_sim = 10000,
  level = 0.95,
  seed = NULL,
  mvn_method = c("mvnfast", "mgcv"),
  ...,
  uncertainty = c("simulation", "delta"),
  unconditional = FALSE
)

## S3 method for class 'scam'
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 = NULL,
  n_sim = 10000,
  level = 0.95,
  seed = NULL,
  mvn_method = c("mvnfast", "mgcv"),
  ...,
  uncertainty = c("simulation", "delta"),
  unconditional = FALSE
)
```

```
## S3 method for class 'gamm4'
response_derivatives(object, ...)
```

Arguments

object	an R object to compute derivatives for.
...	arguments passed to posterior sampling or prediction, such as draws, envir, and exclude. freq = TRUE selects frequentist coefficient covariance for delta uncertainty or Gaussian draws. For delta uncertainty, newdata, se.fit, and terms are not supported; use data and exclude.
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 must be supplied.
data	a data frame containing the values of the model covariates at which to evaluate derivatives of fitted values. 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 via draws in ....
n	numeric; the number of points to evaluate the derivative at (if data is not supplied).
eps	a positive finite number giving the absolute finite-difference step, or NULL (the default) to choose it automatically. The automatic step is the fitted range of the differentiation coordinate multiplied by .Machine$double.eps^(1 / (order + p)), where p = 2 for central differences and p = 1 for forward or backward differences. A constant coordinate uses its absolute value (or one if zero) instead of its range. The step is rounded upwards and bounded below so that adding it to the coordinates produces a representable change. Stored transformed values or raw covariate summaries supply the range; supplied data are used if neither is available. This is a scale-aware heuristic, not an error bound; unusual function scales or domain boundaries may require explicit eps. For first central differences the two points are separated by eps; for second central differences they are each eps from the target.
n_sim	integer; number of posterior draws. Ignored when using user-supplied draws.

level	numeric; 0 < level < 1; the coverage level of the credible interval. The default is 0.95 for a 95% interval. Supply one finite numeric value. Use level for both uncertainty methods; ci_level and coverage are not accepted, even when level is also supplied.
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.
uncertainty	character; "simulation" (default) uses posterior draws and equal-tailed intervals; "delta" uses the delta method and normal intervals. Both methods return pointwise intervals. Sampler controls, including n_sim, method, and seed, are ignored for delta uncertainty.
unconditional	logical; include smoothing-parameter uncertainty in the Bayesian covariance, if available, for delta intervals and Gaussian draws. This does not modify MH or user-supplied draws.

Details

Delta uncertainty uses the same shifted prediction rows and finite-difference formulas as simulation, for both first and second derivatives. It applies those formulas to the coefficient gradients of fitted means and propagates their full joint covariance. Formula offsets are included in predictions. Intervals concern derivatives of the conditional mean, not future observations. With Bayesian coefficient covariance, delta intervals have an approximate posterior interpretation. They are symmetric and may differ from simulation intervals when posterior transformations are strongly nonlinear; the two methods' point estimates can also differ.

Delta uncertainty supports single-linear-predictor gam and bam models with an ordinary scalar inverse-link mean (including negative binomial, Tweedie, beta regression, and scaled t families), and gamm via its GAM component. SCAM models use transformed-coefficient covariance conditional on the fitted smoothing parameters. General multi-predictor families do not support delta uncertainty; their existing simulation support is unchanged. The uncertainty attribute records the selected method.

By default, all supplied prediction rows are returned in their original order. Rows with missing or non-finite required covariates have NA derivatives, standard errors and interval limits. Category, response and family identifiers are retained. An explicit na.action follows the prediction convention: na.omit drops unavailable rows with missing covariates, na.exclude and na.pass retain them, and na.fail reports missing covariates as an error. Undefined calculations at available rows, including in only some coefficient draws, raise an error; draws are never removed to form a summary.

Response-scale derivatives require the expected response to exist. For a scaled t family, fitted degrees of freedom must exceed one, whether fixed or estimated. A finite response variance is not required. Linear-predictor derivatives remain available when the response mean does not exist.

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 posterior median for simulation uncertainty, or the finite difference of fitted predictions for delta uncertainty,
- `.se`: posterior standard deviation for simulation uncertainty (`NA` with only one draw), or the delta-method standard error,
- `.lower_ci`: the lower bound of the pointwise interval,
- `.upper_ci`: the upper bound of the pointwise interval,
- additional columns containing the covariate values at which the derivative was evaluated.

Author(s)

Gavin L. Simpson

See Also

gamm4_support

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"
  )

# pointwise normal intervals from the delta method
y_delta <- response_derivatives(m, data = ds, focal = "x2",
  type = "central", uncertainty = "delta")

# 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
p1 + p2 + plot_layout(nrow = 2)
```

---

rootogram *Rootograms to assess goodness of model fit*

Description

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

Usage

```
## S3 method for class 'gamm4'
rootogram(object, ..., conditioning = NULL)

rootogram(object, ...)

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

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

Arguments

object	an R object
...	arguments passed to other methods
conditioning	for gamm4, "conditional" (default) checks the response distribution at all fitted effects, "marginal" averages over new effects, and "gam_component" checks the extracted GAM.
max_count	integer; the largest count to consider
breaks	for continuous responses, how to group the response. Can be anything that is acceptable as the breaks argument of graphics::hist.default()

References

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

See Also

gamm4_support

Examples

```
load_mgcv()

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

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

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

---

rtw *Simulator for tweedie LSS models*

Description

Simulate random deviates from a Tweedie distribution with given parameters μ, p, and ϕ. Works with vector values for all parameters, unlike the version on *mgcv*.

Usage

```
rtw(mu, p, phi)
```

Arguments

mu	numeric vector of mean values of Tweedie distribution.
p	numeric vector of values for the power parameter of the Tweedie distribution.
phi	numeric vector of values for the scale parameter ϕ of the Tweedie distribution.

Details

Parameters must be finite numeric vectors of length one or a common nonzero length. Length-one parameters are expanded to the common length; partial recycling is not supported. If all parameters are empty, an empty numeric vector is returned; mixing empty and nonempty parameters is an error.

---

sampling_diagnostics *Extract coefficient-sampler diagnostics*

Description

Diagnostics attached to fitted values or response samples describe the coefficient chains before transformation or response simulation. Response noise can conceal poor mixing of coefficient chains. Use `as_mcmc_draws()` to examine particular derived quantities as well.

Usage

```
sampling_diagnostics(x)

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

Arguments

x	Output from a gratia sampling function; for the print method, a `sampling_diagnostics` object.
...	Further arguments passed to the print methods for the summaries.

Details

Coefficient draws are retained when sampling with `diagnostics = TRUE`, allowing `draw.sampling_diagnostics()` to plot the original coefficient chains even when `x` contains response samples, fitted values, smooths or derivatives. These plots do not describe the transformed quantities. Numerical summaries require posterior, but plotting does not.

Value

A `sampling_diagnostics` list with `chains` (acceptance rates and elapsed seconds), `coefficients` (posterior summaries, or `NULL` if not requested), and `draws` (a coefficient matrix with chain identities, or `NULL` if unavailable). Coefficient matrices provide their own draws; other sampling outputs need `diagnostics = TRUE` to retain them. Likelihood matrices from `log_lik()` also require `diagnostics = TRUE` for coefficient plots; use `as_mcmc_draws()` for their log-density draws.

See Also

`draw.sampling_diagnostics()`, `as_mcmc_draws()`

`scalable_qq_plot` *Scalable model diagnostic plots*

Description

Aggregate all observations before drawing model diagnostic plots with ggplot. QQ points are grouped into bins of equal distance along the curve, keeping both endpoints. Heatmaps and histograms contain precomputed counts, so redrawing does not bin the original data again. Uniform references average draws as they are generated. Simulated references retain exact pooled quantiles, using temporary files above the configured memory threshold; see diagnostic_support.

Usage

```
scalable_qq_plot(
  model,
  method = NULL,
  type = NULL,
  n_uniform = 10,
  n_simulate = 50,
  seed = NULL,
  level = 0.9,
  qq_bins = 1000L,
  xlab = NULL,
  ylab = NULL,
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  ci_col = "black",
  ci_alpha = 0.2,
  point_col = "grey20",
  point_alpha = 1,
  line_col = "red",
  by_family = NULL,
  conditioning = NULL,
  ...,
  n_dharma = 250L
```

```
)

scalable_worm_plot(model, ..., n_dharma = 250L)

scalable_residuals_linpred_plot(
  model,
  type = NULL,
  density_bins = c(50L, 50L),
  density = c("conditional", "count"),
  seed = NULL,
  xlab = NULL,
  ylab = NULL,
  title = "Residuals vs linear predictor",
  subtitle = NULL,
  caption = NULL,
  line_col = "red",
  by_family = NULL,
  rank = NULL,
  density_method = c("kde", "histogram"),
  bandwidth = NULL,
  conditioning = NULL,
  ...,
  n_dharma = 250L
)

scalable_observed_fitted_plot(
  model,
  density_bins = c(50L, 50L),
  density = c("count", "conditional"),
  xlab = "Fitted values",
  ylab = "Response",
  title = "Observed vs fitted values",
  subtitle = NULL,
  caption = NULL,
  by_family = NULL,
  conditioning = NULL,
  ...
)

scalable_residuals_hist_plot(
  model,
  type = NULL,
  n_bins = c("sturges", "scott", "fd"),
  seed = NULL,
  xlab = NULL,
  ylab = "Frequency",
  title = "Histogram of residuals",
  subtitle = NULL,
```

```
  caption = NULL,
  by_family = NULL,
  conditioning = NULL,
  ...,
  n_dharma = 250L
)
```

Arguments

model	A fitted model supported by diagnostic_support.
method	character or NULL; how to calculate reference quantiles. Probability integral transform (PIT) and quantile residuals use fixed uniform and standard-normal references, respectively. For quantile residuals, "normal" also selects that fixed reference; other named methods require traditional residuals. For deviance, response and Pearson residuals: • "uniform" applies the fitted inverse cumulative distribution function (CDF) to randomised uniform quantiles and averages the resulting ordered residuals over n_uniform draws. • "simulate" generates n_simulate response data sets at the fitting covariates. Responses are generated together when needed to preserve dependence, and their ordered residuals supply reference values and bands. • "marginal" approximates the pooled distribution of individual residuals, without bands for dependence. This is selected for BAM GEE and PQL correlations that do not define a joint response distribution. Explicit joint simulation is unavailable in those cases. • "normal" uses a normal approximation fitted to the residual location and scale. For dependent responses it omits bands based on independence. • "direct" is deprecated in favour of "uniform". For ordinary GAMs, NULL selects "simulate" for Tweedie and "uniform" otherwise. Missing inverse CDFs cause a switch to simulation; missing response generators can cause a switch to a normal approximation for independent responses. Dependent responses use joint simulation when available, or marginal checks otherwise. For grouped families, NULL selects simulation if every component has a generator, or inverse-CDF sampling with a message if every component has an inverse CDF. If neither is available, it gives an error; a normal approximation must be requested explicitly.
type	Residual type; NULL follows the model-specific defaults in appraise(). PIT and quantile residuals are available for supported response distributions. "surrogate" is available for ocat() and multinom() GAMs with method = NULL: logistic references for ordinal residuals and centred Gumbel references for each nominal category utility. Nominal components always have separate panels. Their residuals are drawn once per observation and shared with ordinary diagnostics. Components are summarised one at a time; QQ bin limits apply to each category. Residuals use category probabilities on the predictor axis, including category 0. See surrogate_residuals() and diagnostic_support.
n_uniform	numeric; number of inverse-CDF draws to average when method = "uniform".

`n_simulate` numeric; number of data sets to simulate from the estimated model when using the simulation method (`method = "simulate"`).

`seed` Random seed for QQ calculations or randomized residuals. Explicit seeds preserve the caller's RNG state, including an absent state.

`level` Coverage of pointwise QQ reference intervals, strictly between zero and one. Used by simulation and normal-reference methods.

`qq_bins` Maximum number of QQ points per family component, including both endpoints; an integer of at least three.

`xlab, ylab` Axis labels; `NULL` selects suitable defaults.

`title, subtitle, caption`
Plot title, subtitle and caption, passed to `ggplot2::labs()`.

`ci_col` Deprecated; use `gratia_theme_get()` theme settings instead. Fill colour for reference intervals, including bands for quantile residuals.

`ci_alpha` Deprecated; use `gratia_theme_get()` theme settings instead. Alpha transparency for reference intervals.

`point_col` Deprecated; use `gratia_theme_get()` theme settings instead. Constant point colour when family components are not mapped to colour. Pooled grouped diagnostics instead colour points by component.

`point_alpha` Deprecated; use `gratia_theme_get()` theme settings instead. Alpha transparency of points on the QQ plot.

`line_col` Deprecated; use `gratia_theme_get()` theme settings instead. Colour used to draw the reference line.

`by_family` logical or `NULL`; separate indexed grouped-family components into facets. The default pools quantile residuals with component colours and facets traditional residuals. `TRUE` and `FALSE` explicitly select the layout. In `appraise()`, observed-versus-fitted values always use component facets. Standalone `observed_fitted_plot()` facets by default and permits explicit pooling. Ignored for ordinary models.

`conditioning` character or `NULL`; `"conditional"` holds fitted random effects fixed, `"marginal"` averages over new effects, and `"gam_component"` uses the extracted GAM. Availability and defaults depend on the model; see diagnostic_support.

`...` optional arguments used to prepare the model for diagnostics, including sdmTMB `field_method`. For grouped-family GAMs and joint GJRM models, `response = NULL` uses every response and an integer selects one; see `appraise()`. Selection leaves `by_family` independent. For `scalable_worm_plot()`, other arguments pass to `scalable_qq_plot()` except its internal worm setting.

`n_dharma` positive integer; number of response simulations for `type = "dharma"`. This is separate from `n_simulate`, which controls traditional QQ reference calculations. Defaults to 250.

`density_bins` Positive integer or pair of integers giving the number of horizontal and vertical heatmap bins.

`density` One of `"conditional"` or `"count"`. Conditional density is estimated using `density_method`. With `"histogram"`, it is the count divided by the horizontal-bin total and vertical-bin width. Empty horizontal bins are blank. Counts are always unsmoothed.

rank	logical or NULL; use within-component ranks on the horizontal axis of residual-versus-predictor plots. The default ranks pooled grouped plots and retains original predictors in facets. TRUE or FALSE overrides this independently of the residual type and layout. Only finite observations included in each component determine its ranks; average ties are used.
density_method	Conditional-density estimator: "kde" (the default) uses KernSmooth::bkde2D() with Gaussian kernels; "histogram" uses unsmoothed rectangular bins. Only affects the residual heatmap.
bandwidth	Optional pair of positive bandwidths for the predictor and residual axes, in their plotted units. NULL uses normal-reference bandwidths, adjusted for two dimensions, with a fallback for constant axes. Used only for density_method = "kde" and conditional density.
n_bins	character or numeric; either the number of bins or a string indicating how to calculate the number of bins.

Details

GJRM support covers uncensored univariate N, LN, GA, GAi, P, NBI, NBII, BE, LO, WEI, IG, GU and rGU models. PIT/quantile references are uniform/normal; traditional response/Pearson references use inverse-CDF or simulated responses. Probabilities are recomputed from final likelihood parameters. Quantile QQ bands give pointwise intervals for ordered standard-normal residuals, conditional on the fitted distribution, without accounting for parameter estimation. Observed-versus-fitted plots use response means, including the lognormal, Weibull and Gumbel mean corrections. Residuals are plotted against the first distributional linear predictor.

Grouped-family models retain the residual defaults, component reference distributions, predictor ranks and layout controls of appraise(). QQ bin limits apply separately to each component. Pooled residual heatmaps summarize the component mixture rather than colouring individual observations; observed-versus-fitted appraisals always separate components. Pooled histograms use faint component-coloured fills with opaque stepped upper boundaries, retaining common bin edges and component densities.

Residual-versus-predictor plots identify the response family in the subtitle and show conditioning and field choices in the caption where relevant, matching residuals_linpred_plot(). Supply caption = "" to omit the caption.

Kernel density estimates are calculated on a fixed grid before plotting. Each predictor slice is normalized to integrate to one over the displayed residual grid. Bandwidths, rather than density_bins, control smoothing; increase grid resolution when using small bandwidths. A square-root fill transformation reveals low-density regions while retaining density units in the legend. Gaussian kernels smooth the bands that discrete residuals can form even under a correctly specified count model. Smoothing does not give traditional residuals a common reference distribution.

Point styling affects QQ and worm points. Heatmap fill scales can be replaced with ggplot scales. Non-finite plotting observations are reported and omitted; fitting exclusions do not count as dropped observations.

Value

A ggplot object containing aggregated plotting data.

References

Fasiolo, M., Nedellec, R., Goude, Y. and Wood, S. N. (2020). Scalable Visualization Methods for Modern Generalized Additive Models. Journal of Computational and Graphical Statistics, 29, 78-86. doi:10.1080/10618600.2019.1629942.

See Also

`appraise()`, `qq_plot()`, `worm_plot()`

Examples

```
load_mgcv()
dat <- data_sim("eg1", n = 400, seed = 2)
m <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data = dat)
scalable_qq_plot(m, seed = 42)
scalable_residuals_linpred_plot(m)
scalable_observed_fitted_plot(m, density_bins = c(30, 40))
appraise(m, scalable = TRUE, seed = 42)
```

---

scale_fill_partial_effect

Default diverging red-blue colour palette for partial effects

---

Description

Default diverging red-blue colour palette for partial effects

Usage

```
scale_fill_partial_effect(
  name = "Partial effect",
  ...,
  na.value = "grey50",
  guide = "colourbar",
  direction = -1
)
```

Arguments

`name`	The name of the scale. Used as the legend title. If `NULL` the legend title will be omitted.
`...`	arguments passed to `ggplot2::continuous_scale()`.
`na.value`	Missing values will be replaced with this value (colour).
`guide`	A function used to create a guide or its name. See `ggplot2::guides()` for more information.
`direction`	Sets the order of colours in the scale. If `1`, the default, colours are as output by `RColorBrewer::brewer.pal()`. If `-1`, the order of colours is reversed.

sdmTMB-methods *Formula effects for sdmTMB models*

Description

Evaluate and plot formula smooths and parametric effects from a fitted sdmTMB model using the same plotting methods as mgcv smooths. Penalized and unpenalized coefficients and their joint covariance are transformed back to the constrained mgcv basis.

Usage

```
## S3 method for class 'sdmTMB'
smooths(object, ...)

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

## S3 method for class 'sdmTMB'
get_smooths_by_id(object, id, ...)

## S3 method for class 'sdmTMB'
smooth_coefs(object, select, ...)

## S3 method for class 'sdmTMB'
smooth_estimates(
  object,
  ...,
  unconditional = FALSE,
  frequentist = FALSE,
  envir = NULL
)

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

## S3 method for class 'sdmTMB'
parametric_effects(
  object,
  terms = NULL,
  ...,
  unconditional = FALSE,
  frequentist = FALSE,
  envir = NULL
)

## S3 method for class 'sdmTMB'
assemble(
```

```
  object,
  ...,
  parametric = NULL,
  residuals = FALSE,
  unconditional = FALSE,
  frequentist = FALSE,
  envir = NULL,
  time_varying_style = c("ribbon", "pointrange")
)

## S3 method for class 'sdmTMB'
draw(
  object,
  ...,
  parametric = NULL,
  residuals = FALSE,
  unconditional = FALSE,
  frequentist = FALSE,
  envir = NULL,
  time_varying_style = c("ribbon", "pointrange")
)
```

Arguments

object, model	A fitted sdmTMB model.
...	Arguments passed to the corresponding gratia methods, including grid, interval, transformation, surface, styling and layout controls. For threshold terms, n sets the number of evenly spaced evaluation points over the fitted covariate range; a fitted breakpoint within that range is also included. Supplied data determines evaluation locations.
id	Integer positions of smooths across model components.
select	Smooth selection across both predictors of delta models. Numeric and logical indices follow stored smooth order, first component then second. Character selections use smooths() labels, such as s(x) and s.1(x) for the second component.
unconditional, frequentist	Must be FALSE. Intervals use the joint RTMB covariance; mgcv-specific alternatives are unavailable.
envir	Optional environment for evaluating model covariates.
terms	Parametric terms to evaluate or draw, as returned by parametric_terms(), including breakpt(x) and logistic(x) threshold terms and time_varying(temperature) coefficient trajectories. Second-component selectors have a .1 suffix.
parametric	Include parametric effects? NULL includes them unless smooths are selected explicitly, as for draw.gam().
residuals	Must be FALSE; partial residuals are not supported.

`time_varying_style`
: Display time-varying coefficients with a line, points and uncertainty ribbon (`"ribbon"`, the default), or with points, interval bars and a faint connecting line (`"pointrange"`).

Details

Use `smooths()` to list smooth names, `get_smooth()` to extract an exact name, `get_smooths_by_id()` to extract by position, and `n_smooths()` to count smooths, including both predictors of a delta model. These utilities do not calculate coefficient covariances.

Smooth, ordinary parametric and threshold estimates are partial contributions on each component's link scale. Spatial, spatiotemporal, IID random effects, offsets and varying coefficients are not added to these effects. Their influence on coefficient uncertainty is retained through the joint covariance. Smooth intervals include uncertainty in their component's overall mean by default; parametric intervals include the component's intercept. Use `overall_uncertainty = FALSE` for term-only intervals. `fun` transforms displayed partial effects, not full predictions. Use `surface = "3d", ci_3d = TRUE` in `draw()` or `assemble()` to add interval surfaces to smooths and numeric parametric interactions in either component. `ci_3d_alpha` and `ci_3d_col` control their opacity and fill colour. Pass `cyclic = "polar"` and optional `polar_args` to `draw()` or `assemble()` for polar plots of supported one-dimensional cyclic smooths. See `draw.gam()` for the coordinate options.

Delta components have separate smooth and parametric coefficients and are drawn separately, with component subtitles. Upstream sdmTMB restrictions on supported bases and delta formulas still apply. Tensor smooths with multiple penalties are supported. Fits require stored smooth metadata; mapping or profiling required coefficients and non-positive-definite Hessians give errors. Parametric subtitles also identify the component for threshold curves and time-varying coefficients. Saved `parametric_effects()` retain the labels in an optional `.model` column.

Support requires sdmTMB >= 1.2.0 models fitted with the RTMB backend. Refit older models or TMB fits with `control = sdmTMB::sdmTMBcontrol(backend = "rtmb")`. Diagnostics use complete fitted predictors, including spatial, spatiotemporal and independent group effects and offsets. Supported single-response and standard delta families provide PIT/quantile residuals, ordinary and scalable diagnostic plots, and joint approximate posterior predictive checks. Delta diagnostics can select the complete response, occurrence or positive component. Raw posterior response draws are available through `posterior_samples()`. By default, fields are drawn together once from their approximate posterior and held fixed (`mle-mvn`), following sdmTMB's recommended residual calculation. Checks can also hold estimates fixed, average over fitted-field uncertainty, or simulate new fields. Marginal simulations preserve dependence and supply residual ranks, including for nonlinear and coupled delta responses. See diagnostic_support for treatment of coefficients, fields and covariance, and what each check assesses.

Threshold curves are partial contributions without centring. They follow parametric selection rules: `terms` selects them and `parametric = FALSE` excludes them. Both delta components have separate threshold parameters. Threshold intervals use the delta method with the joint covariance of the fitted parameters, including uncertainty in the breakpoint or logistic width. By default they include intercept uncertainty and its covariance with the curve, without adding the intercept to the estimated contribution. Use `overall_uncertainty = FALSE` for term-only uncertainty. At the breakpoint, the derivative is taken from the constant side, matching native sdmTMB prediction. Threshold intervals remain pointwise when simultaneous smooth bands are requested. Rugs show fitting observations.

Time-varying coefficients are plotted at their stored time steps, including `extra_time`; `n` does not add time steps. Select each fitted design column with a label such as `time_varying(temperature)`

or `time_varying(Intercept)`. A covariate named `Intercept` is distinguished from the intercept by backticks: `time_varying(`Intercept`)`. Factor contrasts, transformations and interactions retain the fitted coefficient coding. For `rw`, estimates are the temporal coefficients. For `rw0` and `ar1`, the matching ordinary coefficient is added to its temporal deviation. Their joint covariance includes the variance of both parts and their covariance. An absent ordinary coefficient contributes zero. Matching columns must have compatible coding in the main and temporal designs. Default effects omit ordinary terms whose coefficients are already included in trajectories; their ordinary selectors remain available for explicit requests. `overall_uncertainty` governs ordinary partial effects; temporal panels always retain their complete coefficient covariance. Temporal intervals remain pointwise when simultaneous smooth bands are requested. Supplied `data` selects stored time steps if it contains the model's time column; otherwise all stored times are retained. Missing or unknown requested times give an error. sdmTMB 1.2.0 requires numeric time values for time-varying processes.

Time-varying estimates from `parametric_effects()` use `.type = "time_varying"`, `.value` for time, `.partial` for the full coefficient, and `.se` for its standard error. Both display styles are also available through `draw.parametric_effects()`, including nested saved estimates.

Smooth extraction and effect plots do not support dispersion formulas, other multi-predictor families, or nonlocal effects. Spatial fields and group random effects are not plotted.

Value

Smooth labels, smooth objects, coefficients, effect estimates, or plots, as for the corresponding GAM methods. `draw()` returns a patchwork, or named panels with `wrap = FALSE`; `assemble()` returns an `assembled_plots` list. Models without smooths can draw parametric effects.

Label layout

Pass `label_control = draw_label_control(...)` to `draw()` to wrap discrete axis labels and legend text, stagger discrete tick labels, or adjust legend and colour-bar layout. Settings apply to each panel, including unwrapped plot lists. See `draw_label_control()` for the available settings.

Examples

```
if (requireNamespace("sdmTMB", quietly = TRUE)) {
  load_mgcv()
  # Simulate nonlinear and linear effects with Gaussian noise
  set.seed(42)
  d <- data.frame(x = runif(150), z = runif(150))
  d$y <- 1 + sin(6 * d$x) + 0.5 * d$z + rnorm(150, sd = 0.3)

  # Fit without a spatial field and draw the linear and smooth effects
  m <- sdmTMB::sdmTMB(y ~ z + s(x, k = 6), data = d, spatial = "off")
  smooths(m)
  draw(m)
}
```

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 eps. This is particularly useful in computing derivatives via finite differences where without this adjustment we may be predicting for values outside the range of the data and hence the constraints of the penalty.

Usage

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

Arguments

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.

shape_confidence_set *Intersect a band with linear restrictions on coefficients*

Description

Represent the set of coefficient vectors whose evaluated functions are inside an initial band and satisfy supplied linear inequalities and equalities.

Usage

```
shape_confidence_set(
  X,
  lower,
  upper,
  Ain = NULL,
  bin = NULL,
  C = NULL,
  d = NULL,
  level = 0.95,
  interpretation = c("confidence", "credible"),
  band_type = c("unspecified", "simultaneous", "pointwise")
)

in_shape_set(object, coefficients, tolerance = 1e-08)
```

Arguments

`X`	A finite numeric matrix that maps coefficients to function values at the evaluation points. Each column corresponds to a coefficient.
`lower`, `upper`	Numeric vectors of lower and upper limits, one per row of `X`. Infinite limits are allowed. Missing values are not allowed.
`Ain`	A finite numeric matrix defining inequalities `Ain %*% b >= bin`. It must have the same number of columns as `X`. `NULL` means no inequalities.
`bin`	A finite numeric vector, one entry per row of `Ain`. If `NULL`, zero is used for every inequality.
`C`	A finite numeric matrix defining equalities `C %*% b = d`. It must have the same number of columns as `X`. `NULL` means no equalities.
`d`	A finite numeric vector, one entry per row of `C`. If `NULL`, zero is used for every equality.
`level`	A number strictly between zero and one recording the nominal level of the initial band. This does not calibrate the band.
`interpretation`	Is the initial band a confidence or credible band?
`band_type`	Record whether the initial limits were calibrated jointly (`"simultaneous"`), separately at each evaluation point (`"pointwise"`), or whether their calibration is unspecified (`"unspecified"`, the default).
`object`	A set created by `shape_confidence_set()`.
`coefficients`	A finite numeric vector, or a finite numeric matrix with one candidate coefficient vector in each column. Rows must correspond to columns of `object$X` in the same order; names are not used for matching.
`tolerance`	A finite non-negative number used as an absolute tolerance in function values and constraint residuals. The default is `1e-8`. Setting it to zero uses exact floating-point comparisons. Multiplying a constraint row by a constant changes the effect of an absolute tolerance.

Details

The returned object represents the set

$$\{b : \text{lower} \leq Xb \leq \text{upper}, \quad A_{in}b \geq b_{in}, \quad Cb = d\}.$$

The matrices are retained explicitly; the function does not optimise over the set, test whether it is empty, or draw samples from it. Redundant and inconsistent restrictions are allowed. Inconsistent restrictions represent an empty set.

The initial band restricts function values only at the supplied evaluation points. The extent of the shape restriction depends on the supplied coefficient inequalities. For example, appropriate spline coefficient inequalities can guarantee monotonicity throughout a spline's domain, whereas differences between evaluated function values guarantee it only on that grid. This function does not infer restrictions from a fitted model or extend restrictions between evaluation points.

If the true coefficient vector satisfies the supplied restrictions, it is in the intersection exactly when its evaluated function is in the initial band. Thus intersection preserves the coverage of an initially valid simultaneous confidence band, for truths satisfying those restrictions. Pointwise limits alone do not supply a simultaneous confidence set. For credible bands, the same posterior probability statement requires the posterior to be supported entirely on the restricted coefficient space. Intersection does not fix smoothing bias, invalid initial calibration, or an incorrect restriction.

Coordinatewise minima and maxima of functions in this set need not themselves belong to the set. This matters for coupled restrictions such as an integral equal to one: two ordered functions with that integral must coincide almost everywhere. Retaining the whole set avoids assigning such equalities to separate endpoint curves.

Value

`shape_confidence_set()` returns a list of class `"shape_confidence_set"` containing `X`, `lower`, `upper`, `Ain`, `bin`, `C`, `d`, `level`, `interpretation`, and `band_type`. Missing constraint matrices are stored as matrices with zero rows. `in_shape_set()` returns a logical vector, one value per candidate coefficient vector.

Examples

```
# Two non-negative masses whose sum is one.
region <- shape_confidence_set(
  X = diag(2), lower = c(0.1, 0.2), upper = c(0.8, 0.9),
  Ain = diag(2), C = matrix(c(1, 1), nrow = 1), d = 1,
  band_type = "simultaneous"
)
in_shape_set(region, c(0.4, 0.6))
in_shape_set(region, c(0.4, 0.7))
```

`shape_intervals` *Enforce named shapes on estimates and interval endpoints*

Description

Apply shape-enforcing operators to an existing estimate and its lower and upper endpoints. Construction and calibration of the initial uncertainty remain the caller's responsibility.

Usage

```
shape_intervals(
  estimate,
  lower,
  upper,
  data,
  constraints,
  interval = c("simultaneous", "pointwise"),
  level = 0.95
)
```

Arguments

`estimate`, `lower`, `upper`	finite numeric vectors, one value per row of `data`. The lower endpoints must not exceed the upper endpoints.
`data`	a data frame of finite numeric coordinates. Rows may be in any order but must form a complete Cartesian product without duplicates. Coordinates may be irregularly spaced. A single coordinate gives a curve.
`constraints`	a named list, with coordinate names as names and character vectors of restrictions as elements. Codes are `"m+"` (nondecreasing), `"m-"` (nonincreasing), `"c+"` (convex), `"c-"` (concave), and `"+"` (nonnegative). Curvature restrictions may occur in at most one coordinate. Monotonicity may be imposed in several coordinates. Positivity applies to the function, regardless of which coordinate carries its code.
`interval`	character; whether the input is a `"simultaneous"` band or `"pointwise"` limits. This describes the input, and does not calibrate it.
`level`	numeric; the nominal level of the input, strictly between zero and one.

Details

The operator first thresholds at zero if requested, then sorts each monotone coordinate in the order given by `constraints`, then takes the greatest convex minorant in the curvature coordinate. Concavity uses the negative of the greatest convex minorant of the negative values. The minorant uses actual coordinate distances. Sorting gives equal mass to each grid coordinate, including on irregular grids; it does not approximate Lebesgue-weighted rearrangement on such grids.

These finite-grid operators preserve order, leave a function with the requested shapes unchanged, and do not increase maximum absolute distance. Consequently, whenever the entire initial band contains a target with the requested shapes on the grid, the transformed band also contains it. Its maximum width cannot increase. This is a deterministic statement about a whole-grid event; it does not guarantee nominal pointwise coverage after transforming pointwise limits. Nor does it correct smoothing bias or an invalid initial uncertainty calculation. For a posterior supported on the restricted functions the analogous event inclusion applies to posterior probability; this operation does not construct that posterior.

The guarantees concern values on the supplied grid. Linear interpolation of curves, or multilinear interpolation on rectangles, also has the named directional shapes between grid points. The output is not a fitted spline and need not obey coefficient inequalities, centring, integral constraints, or other equalities. Use `shape_confidence_set()` to represent intersections with such restrictions. Several curvature coordinates are unsupported: sequential directional convexification need not preserve earlier convexity. Extreme numeric ranges that prevent reliable slope comparisons or interpolation in double precision produce an error.

Value

A tibble containing `data`, `.estimate`, `.lower_ci`, and `.upper_ci`, in the original row order. The `interval_info` attribute records the operator, restrictions, input interval type and coverage scope. Standard errors and Gaussian critical values are not recalculated.

References

Chernozhukov, V., Fernandez-Val, I. and Galichon, A. (2009). Improving point and interval estimators of monotone functions by rearrangement. *Biometrika*, **96**, 559–575. doi:10.1093/biomet/asp030.

Chen, X., Chernozhukov, V., Fernandez-Val, I., Kostyshak, S. and Luo, Y. (2021). Shape-Enforcing Operators for Generic Point and Interval Estimators of Functions. *Journal of Machine Learning Research*, **22**(220), 1–42. https://www.jmlr.org/papers/v22/20-513.html.

Examples

```
x <- c(0, 0.1, 0.4, 0.8, 1)
f <- c(0.1, 0.4, 0.2, 0.9, 0.7)
shape_intervals(f, f - 0.3, f + 0.3, data.frame(x = x),
  constraints = list(x = c("+", "m+")))
```

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

Description

Shift numeric values in a data frame by an amount `eps`

Usage

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

Arguments

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

---

simulate *Simulate responses from a fitted model*

Description

Generate replicated responses using fitted parameter values. posterior_samples() also includes uncertainty in the coefficients. See diagnostic_support.

Usage

```
## S3 method for class 'gamm4'
simulate(object, nsim = 1, seed = NULL, conditioning = NULL, ...)

## S3 method for class 'gamlss'
simulate(object, nsim = 1L, seed = NULL, ...)

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

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

```
  ...,
  newdata = NULL,
  conditioning = NULL,
  envir = NULL
)

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

Arguments

object	a fitted GAM, typically the result of a call to mgcv::gam(), mgcv::gamm(), or gamm4::gamm4().
nsim	numeric; number of response simulations to return. Supply one finite positive whole number, no larger than .Machine$integer.max.
seed	numeric or NULL; a random seed for the simulations. An explicit seed restores the previous random-number state, including its absence, when simulation finishes or fails. With NULL, simulation advances or initializes the random-number state.
conditioning	character or NULL; optional mixed-model conditioning mode. gamm4 defaults to "conditional", holding all fitted effects fixed. "gam_component" explicitly uses its extracted GAM. The existing gamm simulator retains its GAM-component default. Full mixed-model modes currently use fitting observations.
...	arguments passed to methods. simulate.gam() and simulate.scam() pass ... on to predict.gam(). As such you can pass additional arguments such as terms, exclude, to select which model terms are included in the predictions. This may be useful, for example, for excluding the effects of random effect terms.
data	data frame; new observations at which the posterior draws from the model should be evaluated. If not supplied, the data used to fit the model will be used for newdata, if available in object.
weights	numeric; response sampling weights. An explicit vector overrides the fitting weights; a single value applies to every observation. With data = NULL, the default is the fitted prior weights, except factor-valued binomial responses retain one Bernoulli trial per observation. Native gaulss() ignores fitting weights in its likelihood, so its default response weights are ones; explicit sampling weights still override this. With prediction data, the default is a vector of ones. For ordinary Gaussian and Gamma responses, larger weights reduce response variance

(precision weights). For binomial responses, weights give integer trial counts. Include positions restored by na.exclude when supplying weights for fitting rows.

newdata — Deprecated. Use data instead.

family_index — numeric; for grouped-family models, one integer family index per requested prediction row. See Details.

envir — environment or NULL; functions and constants needed to evaluate model expressions in prediction data. With NULL, use the model's formula environment. Supply raw covariates in data.

marginalise — NULL retains the usual predictions. TRUE selects all supported re and fs terms; character vectors select exact smooth labels. Fitted predictions integrate response means over new zero-mean Gaussian effects. Response samplers draw new effects before sampling responses. Smoothing parameters and dispersion remain fixed. Only ordinary gam() and bam() fits with a scalar response mean are supported.

Details

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

With marginalise, each replicate draws new coefficients for selected re and fs terms before drawing responses. Other coefficients and all variance parameters stay fixed. Supply grouping columns for new data; repeated group labels share effects within a replicate. Unfamiliar labels are allowed only when those factors occur exclusively in selected terms. The random_effect_info attribute records the fixed covariance estimates.

With data = NULL, na.exclude restores excluded observations as NA in every simulation; na.omit returns retained observations only. Positions refer to the fitting data after any subset. Explicit data are evaluated independently of training exclusions: missing responses do not prevent simulation, while missing required predictors or weights yield NA. Prediction na.action in ... is honoured. Missing rows are never passed to the response random-number generator.

Inverse Gaussian models use a native generator with observation dispersion equal to the model dispersion divided by the prior weight, consistently with their CDF and quantile helpers. Weights must be non-negative; a zero weight with positive dispersion uses the limiting point mass at zero. An existing random generator supplied in the family's rd component is preserved.

For models fitted with mgcv::gfam(), each observation is simulated from its component family using that component's fitted scale and distribution parameters. With data = NULL, fitted family membership is used. For explicit data, supply family_index or the index in the model's response expression: the second column of a matrix response, the second argument of cbind(y, index), or a numeric index vector in place of a named matrix response. Actual response observations are not needed. Missing indices follow prediction na.action; invalid indices are errors. Explicit indices must agree with recoverable ones.

Grouped simulations return one numeric response per observation: binomial proportions (with integer trial counts supplied in weights), count responses, or ordered-category codes, as appropriate. Component random generators retain their usual treatment of prior weights. Ordered-categorical and ziP components are simulated from their latent predictors. General families with multiple linear predictors and matrix responses encoding censoring or binomial counts are outside the scope

supported by `gfam()`. Scalar censored-family components produce draws from their uncensored latent distributions.

Value

(Currently) A data frame with `nsim` columns.

Author(s)

Gavin L. Simpson

See Also

gamm4_support

Examples

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

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

---

simultaneous_bands *Simultaneous bands on an evaluation grid or a continuous interval*

Description

Simulation calibrates Gaussian bands jointly over the returned evaluation points. Tube calibration covers a continuous interval for supported one-dimensional cubic splines, including the space between returned points.

Details

Request `interval = "simultaneous"` and `simultaneous_method = "tube"`. The simulation method remains the default. Tube bands use the same Gaussian coefficient covariance selected by `frequentist` and `unconditional` as simulation. For smooth estimates they also retain `overall_uncertainty`. They calibrate separately for each smooth or factor-level comparison; they do not cover all comparisons jointly.

Let $d(x)$ be the uncertainty design and V the coefficient covariance. With $V = LL^T$, set $u(x) = L^T d(x)/\sqrt{d(x)^T V d(x)}$. The method integrates $\kappa = \int_a^b \|u'(x)\|\, dx$ and chooses c so that

$$2\{1 - \Phi(c)\} + \kappa \exp(-c^2/2)/\pi = 1 - \text{level}.$$

This expression bounds the two-sided exceedance probability by counting threshold crossings of the standardized Gaussian process. Repeated paths can make it conservative. Regular components separated by zero variance receive separate endpoint contributions. Polynomial knots divide numerical

integration, but do not add endpoint contributions when the process is continuous. Zero-variance and rank-one processes are handled separately.

Under the selected Gaussian approximation, the band has at least the requested simultaneous probability over the stated interval, subject to numerical accuracy. Bayesian covariance gives an approximate posterior interpretation. Frequentist covariance calibrates centered Gaussian sampling errors; coverage of an unknown fixed curve also depends on smoothing bias. The `unconditional` covariance correction retains its existing meaning and does not integrate the full smoothing-parameter posterior. Numerical integration error estimates are diagnostics, not certified error bounds.

Supported quantities:

Ordinary one-dimensional cubic regression splines (`"cr"`, `"cs"`), cubic P-splines (`"ps"` with their ordinary cubic degree), and ordinary cubic thin plate bases (`"tp"`, `"ts"`) are supported. Factor-by smooths define separate curves. Numeric by multipliers must be fixed throughout each curve. Unconstrained GAMs and their BAM, GAMM, and GAMM4 adapters use their fitted prediction basis and covariance. Surfaces, constrained models, shape enforcement, and unsupported bases produce an error; there is no automatic switch to simulation.

First and second derivative bands retain the requested finite-difference functional, stencil, and step. The operator is evaluated using integrals of spline derivatives to avoid subtracting nearly equal basis values. This can improve floating-point accuracy relative to the simulation path without changing the mathematical functional. The continuous band refers to that derivative approximation. Differentiation must be on the smooth coordinate (`wrt = "smooth"`, or the same coordinate on the covariate scale). Raw-scale derivatives of transformed terms are not supported by tube calibration. Difference bands retain the joint covariance of the two smooths and include group means when requested. The selected parametric group contributions must be constant along the curve.

Domain and numerical controls:

`simultaneous_domain` accepts two increasing finite endpoints, or a named list keyed by coordinate. By default it is the range of returned evaluation coordinates, including gaps. A single evaluation location requires an explicit interval. All returned points must lie inside the domain, which must lie inside the fitted spline support; derivative stencils must also remain inside that support. The domain does not imply that each compared group has observations throughout it. With a fixed domain, `n` changes output density but does not change calibration. Connecting plotted bounds by straight segments is a rendering approximation to the continuous band.

`tube_control` accepts `rel_tol = 1e-6`, `abs_tol = 1e-8`, `subdivisions = 1000L`, and `root_tol = 1e-8`. Integration is split at polynomial boundaries and checked for convergence. Failures produce errors. Simulation controls (`n_sim`, `n_cores`/`ncores`, and `seed`) are ignored; tube calibration does not use random numbers. Computation controls require the fitted model. Drawing saved estimates uses their stored interval limits.

Returned metadata:

The `interval_info` attribute records `calibration = "tube"`, `calculation = "tube_crossing_bound"`, the Gaussian covariance flags, and the per-smooth or per-comparison coverage scope. Its `curves` list contains identities, coordinates, resolved domains and their sources, critical values, lengths, regular-component counts, numerical diagnostics, and derivative stencils and steps where applicable. Methodological details are not added to plot captions.

References

Sun, J. and Loader, C. R. (1994). Simultaneous confidence bands for linear regression and smoothing. *The Annals of Statistics*, **22**, 1328–1345. doi:10.1214/aos/1176325631.

Krivobokova, T., Kneib, T. and Claeskens, G. (2010). Simultaneous confidence bands for penalized spline estimators. *Journal of the American Statistical Association*, **105**, 852–863. doi:10.1198/jasa.2010.tm09165.

Wiesenfarth, M., Krivobokova, T., Klasen, S. and Sperlich, S. (2012). Direct simultaneous inference in additive models and its application to model undernutrition. *Journal of the American Statistical Association*, **107**, 1286–1296.

Liebl, D. and Reimherr, M. (2023). Fast and fair simultaneous confidence bands for functional parameters. *Journal of the Royal Statistical Society: Series B*, **85**, 842–868. doi:10.1093/jrsssb/qkad026.

See Also

`smooth_estimates()`, `derivatives()`, `difference_smooths()`, `confint.gam()`, `draw.gam()`

Examples

```
load_mgcv()
set.seed(4)
dat <- data.frame(x = seq(0, 1, length.out = 100))
dat$y <- sin(6 * dat$x) + rnorm(100, sd = 0.2)
model <- gam(y ~ s(x, bs = "cr", k = 8), data = dat, method = "REML")
band <- smooth_estimates(model,
  interval = "simultaneous",
  simultaneous_method = "tube", simultaneous_domain = c(0, 1)
)
draw(band)
rate <- derivatives(model,
  type = "central", interval = "simultaneous",
  simultaneous_method = "tube"
)
attr(rate, "interval_info")$curves[[1]]$critical
```

---

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

Description

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

Format

A data frame with 12 rows and 7 variables.

Details

The variables are as follows:

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

Source

Simpson, G.L. (Unpublished data).

---

smooth_coef_indices *Indices of the parametric terms for a particular smooth*

Description

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

Usage

```
smooth_coef_indices(smooth)
```

Arguments

smooth	an object that inherits from class mgcv.smooth

Value

A numeric vector of indices.

Author(s)

Gavin L. Simpson

See Also

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

smooth_coefs *Coefficients for a particular smooth*

Description

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

Usage

```
smooth_coefs(object, ...)

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

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

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

## S3 method for class 'gamm4'
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

```
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_data` *Generate regular data over the covariates of a smooth*

Description

Generate regular data over the covariates of a smooth

Usage

```
smooth_data(
  model,
  id,
  n = 100,
  n_2d = NULL,
  n_3d = NULL,
  n_4d = NULL,
  offset = NULL,
  include_all = FALSE,
  var_order = NULL,
  envir = NULL,
  back_transform = FALSE
)
```

Arguments

`model`	a fitted model
`id`	the number ID of the smooth within `model` to process.
`n`	numeric; the number of new observations to generate.
`n_2d`	numeric; the number of points along each of the first two axes of a smooth surface, including surface panels of higher-dimensional smooths. The default, `NULL`, uses `n` instead. Factor levels are retained, and curves with only one continuous covariate use `n`.

n_3d	numeric; the number of new observations to generate for the third dimension of a 3D smooth. For factor conditioning dimensions, retain at most this many levels, taking the first in fitted order.
n_4d	numeric; the number of new observations to generate for the dimensions higher than 2 (!) of a kD smooth (k >= 4). For example, if the smooth is a 4D smooth, each of dimensions 3 and 4 will get n_4d new observations. For factor conditioning dimensions, retain at most this many levels per dimension, taking the first in fitted order.
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.
envir	an optional environment supplying functions and constants used in model expressions. The available model formula environment is used when NULL. Covariate observations should be supplied in data.
back_transform	logical; generate the grid on original covariate scales? The default, FALSE, uses evenly spaced model coordinates. With TRUE, recover raw fitting rows and generate original-scale coordinates, then evaluate the model expressions forwards. Each axis must depend on one distinct original model covariate. Both raw and evaluated columns are returned, with auxiliary .gratia_raw_* columns for transformed axes. Missing raw inputs or unavailable functions produce an error. Numeric by multipliers are one on automatic grids, including when a multiplier shares an original axis name; raw axis coordinates are retained separately. If the multiplier is itself an untransformed grid axis, supply prediction data to smooth_estimates() instead. Typical values added by include_all = TRUE to model-scale grids do not provide raw coordinates for transformed axes.

Details

Spline-on-the-sphere grids use evenly spaced observed-range latitude and longitude values. Plotting constructs geographic cell edges separately, without moving these evaluation coordinates.

Examples

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

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

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

`smooth_dim` *Dimension of a smooth*

Description

Extracts the dimension of an estimated smooth.

Usage

```
## S3 method for class 'gamm4'
smooth_dim(object)

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

See Also

gamm4_support

`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_2d = 50,
  n_3d = 16,
  n_4d = 4,
  data = NULL,
  unconditional = FALSE,
  frequentist = FALSE,
  overall_uncertainty = TRUE,
  dist = NULL,
  unnest = TRUE,
  partial_match = FALSE,
  clip = FALSE,
  envir = NULL,
  back_transform = FALSE,
  interval = NULL,
  ci_level = 0.95,
  n_sim = 10000,
  n_cores = 1,
  seed = NULL,
  interval_method = c("auto", "gaussian", "bootstrap", "shape"),
  shape_source = c("gaussian", "bootstrap"),
  ...,
  variables = NULL,
  simultaneous_method = c("simulation", "tube"),
  simultaneous_domain = NULL,
  tube_control = list()
)
```

```
## S3 method for class 'gamlss'
smooth_estimates(
  object,
  select = NULL,
  smooth = deprecated(),
  ...,
  partial_match = FALSE
)

## S3 method for class 'gjrm'
smooth_estimates(
  object,
  select = NULL,
  smooth = deprecated(),
  ...,
  partial_match = FALSE
)
```

Arguments

object	a fitted model, including objects of class "gam", "gamm" or models fitted by GJRM::gamlss() or GJRM::gjrm().
...	arguments passed to other methods.
select	character, numeric or logical; smooths to evaluate, using labels from smooths() or indices in fitted smooth order. The default (NULL) evaluates all smooths.
smooth	**[Deprecated]** Use select instead.
n	numeric; the number of points over the range of the covariate at which to evaluate a univariate smooth.
n_2d	numeric; the number of points along each of the first two axes of a smooth surface, including surface panels of higher-dimensional smooths. The default is 50 in plotting and plot-preparation functions. If NULL, use n instead. Ignored when evaluation data are supplied. Factor levels are retained, and curves with only one continuous covariate use n.
n_3d, n_4d	numeric; the number of points along the third axis of a 3D smooth (n_3d, default 16), or each axis after the first two for smooths of dimension four or higher (n_4d, default 4). If NULL, use n for those axes. For factor conditioning dimensions, retain at most that many levels, taking the first in fitted order. The first two surface axes use n_2d. Supplied data are unchanged.
data	a data frame of covariate values at which to evaluate the smooth.
unconditional	logical; if TRUE (and only if frequentist == FALSE) then the bayesian smoothing parameter uncertainty-corrected covariance matrix is returned, if available. Whether it is available depends on which smoothness selection method was used to fit the model.
frequentist	logical; if FALSE, the default, the bayesian covariance matrix is returned, otherwise the frequentist covariance matrix.

`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.

`clip`
: logical; should evaluation points be clipped to the boundary of a soap film smooth? The default is `FALSE`, which will return `NA` for any point that is deemed to lie outside the boundary of the soap film.

`envir`
: an optional environment supplying functions and constants used in model expressions. The available model formula environment is used when `NULL`. Covariate observations should be supplied in `data`.

`back_transform`
: logical; generate automatic grids on the original covariate scale? The default, `FALSE`, preserves evenly spaced model-scale grids (e.g. in `log(x)`). With `TRUE`, recover raw fitting covariates and generate grids on their original scale, then evaluate the model expressions forwards. Each axis must depend on one distinct original model covariate. Functions such as `foo(x)` require no inverse. Fitted prediction parameters are retained where provided by the model, including for `scale(x)`. When `data` are supplied, use those rows without generating a grid; `TRUE` requires raw coordinates for each axis. Returned model-coordinate columns keep their usual names and raw coordinates are retained in auxiliary `.gratia_raw_*` columns. Draw these estimates with `draw(object, back_transform = TRUE)` to display the raw coordinates. Estimates containing only transformed coordinates give an informative error if drawn on the original scale. This option does not change effect values or their intervals at the chosen evaluation points.

`interval`
: character or `NULL`; `NULL` (the default) returns estimates and standard errors only. `"confidence"` adds pointwise intervals and `"simultaneous"` adds a band covering, by default, the finite evaluation grid of each fitted smooth. `"credible"` is an alias for `"confidence"`; the default Bayesian covariance gives approximate Bayesian credible intervals.

`ci_level`
: numeric; interval coverage, strictly between zero and one.

`n_sim`, `n_cores`
: positive integers; coefficient draws and simulation cores for simultaneous bands. Parallel simulation requires OpenMP support.

`seed`
: integer or `NULL`; an explicit simulation seed preserves the caller's RNG state. Shared coefficient draws make results independent of selected smooth order and parallel evaluation scheduling.

`interval_method`
: character; "auto" uses stored scasm bootstrap draws for pointwise intervals when available, and Gaussian intervals otherwise. Use "gaussian" to request covariance-based limits or "bootstrap" to require stored draws. Simultaneous bootstrap

bands are not supported. "shape" applies shape_intervals() to supported scasm smooth components, using the initial limits chosen by shape_source.

shape_source — character; initial uncertainty for interval_method = "shape": "gaussian" (the default) or stored "bootstrap" pointwise limits. Bootstrap sources require frequentist = FALSE and unconditional = FALSE because the stored draws hold smoothing fixed.

variables — Character vector giving every covariate of the selected smooths exactly once, in plotting order: x, y, then the conditioning covariate for 3D smooths, or facet rows and columns for 4D smooths. When supplied, selected smooths must use the same covariates. Use fitted coordinate names, such as "log(x)" for a transformed covariate. The default, NULL, favours a two-variable tensor margin for x and y, otherwise retaining fitted order. Grid resolution follows this order. Supplied data are evaluated unchanged. Drawing saved estimates can change the order but uses only points already evaluated; evaluate again to change grid resolution or conditioning values. Continuous covariates must precede factors. Spherical smooths retain their latitude/longitude convention and do not support reordering.

simultaneous_method — character; calibration for simultaneous bands. The default, "simulation", uses coefficient draws on the evaluation grid. "tube" uses a conservative Gaussian threshold-crossing bound over a continuous interval for supported one-dimensional cubic splines. See simultaneous_bands.

simultaneous_domain — numeric pair of increasing finite endpoints, or a named list of such pairs keyed by spline coordinate. For tube bands, NULL uses the range of returned evaluation coordinates, including gaps. All returned points must lie inside the interval and the interval must lie inside the fitted spline support. Requires tube calibration.

tube_control — named list controlling numerical tube calibration: rel_tol (default 1e-6), abs_tol (1e-8), subdivisions (1000L), and root_tol (1e-8). Requires tube calibration. Simulation controls n_sim, n_cores (or ncores), and seed are ignored by tube calibration.

Details

For GJRM::gamlss() and GJRM::gjrm() models, smooths are evaluated in equation and smooth order using the final fitted GJRM coefficients and covariance matrices. Selection uses the model-wide labels returned by smooths(), including suffixes such as s.1(x) for repeated labels. Equations without smooths are skipped; a model without any smooths gives an error. All grid, data, environment, coordinate transformation and interval controls are passed to the GAM method for each selected equation. frequentist = TRUE uses the final GJRM frequentist covariance. A smoothing-parameter uncertainty correction is unavailable: unconditional = TRUE warns and uses the uncorrected final covariance, as for GAMs without that correction.

For distributional and glmmTMB models with multiple linear predictors, an optional .model column identifies the model part or parameter associated with each smooth. This display metadata is retained in nested output and subsets, allowing draw() to reproduce model subtitles from saved estimates. Smooth labels and selection remain unchanged.

When distinct fitted terms share both their smooth label and type (for example, tensor products of soap-film components), `.smooth_id` contains their fitted smooth indices. This preserves their identity in saved estimates and plots without changing the labels supplied by mgcv.

By default, simultaneous coverage is per fitted smooth over its valid evaluation points, including all its slices and factor levels. Separate factor-by smooths remain separate coverage sets when grouped in one plot. This finite-grid band does not claim joint coverage across selected smooths or the full fitted model. Increasing grid resolution improves the approximation to whole-curve coverage. For supported univariate cubic splines, `simultaneous_method = "tube"` instead calibrates over a continuous interval; see simultaneous_bands. `overall_uncertainty` and covariance options apply to both interval types. SCAM smooth estimates retain the fitted centring and coefficient transformation when evaluation points are changed. Pointwise intervals use the covariance of the constrained coefficients. With `overall_uncertainty = TRUE`, their standard errors also include intercept uncertainty and its covariance with the smooth; estimates remain centred. Models without an intercept use smooth-only uncertainty for either setting. Native scasm fits use component-only uncertainty by default. Explicitly requesting `overall_uncertainty = TRUE` warns once if selected smooths lack the centring information needed to add mean uncertainty. Ordinary GAM defaults are unchanged.

Bootstrap pointwise limits follow mgcv's percentiles and its `svar.only` correction for smoothing bias. The correction can move limits beyond the fitted shape constraints. Estimates and covariance-based standard errors are unchanged; `.crit` is `NA` for asymmetric bootstrap limits. No new replicates are generated. Simultaneous intervals use Gaussian draws and warn if automatic selection leaves stored bootstrap draws unused.

The `interval_info` attribute records the interval source and correction. The `shape_constraints` attribute identifies requests and retained inequalities for each fitted smooth and tensor margin. Captions use plotmath arrows for increasing and decreasing constraints while `.type` and `smooth_type()` keep their existing basis names. Unsupported tensor requests produce a warning and are labelled as not retained.

SCAM captions describe the basis and its constraints separately from the compact `.type` values. Increasing and decreasing constraints use plotmath arrows. Tensor captions name each covariate and its constraint; local constraints include the fitted change point. The `scam_smooth_info` attribute preserves these descriptions when plotting saved estimates.

Shape enforcement is explicit and leaves the existing interval defaults unchanged. It preserves containment of a shape-restricted target whenever the entire initial band contains that target on the evaluation grid. This does not establish nominal coverage of the initial band, or preserve marginal coverage of transformed pointwise limits. The default Gaussian source uses approximate Bayesian covariance and does not construct a constrained posterior. Standard errors describe the initial calculation; `.crit` is `NA` after shape enforcement. Metadata records the initial source, correction, named shapes, model coordinates and whole-grid guarantee.

Every selected term must have retained named scasm restrictions. Supported combinations are positivity, monotonicity in any directions and curvature in at most one direction. Tensor smooths require `te(..., np = FALSE)` with univariate B-spline margins (`"sc"` or `"bs"`). Normal reparameterisation and other complementary bases can invalidate the named surface shape even when coefficient inequalities were retained. Curvature additionally requires equally spaced fitted knots to rounding precision and degree at least two. Irregular evaluation coordinates are supported. Complete product grids, finite endpoints and evaluation inside the fitted B-spline domains are required; use `dist = 0` for tensors. Numeric `by` variables are unsupported; factor-by components and separate linear predictors are treated separately. Function-valued bases are required; derivative bases are unsupported.

General pc inequalities can replace the named coefficient restrictions while leaving their original xt request in the fitted object. These replacements are labelled as general constraints and rejected for automatic shape intervals. Point equalities that retain the named inequalities are supported.

Shape intervals use component-only uncertainty: explicit overall_uncertainty = TRUE, effect shifts and effect transformations are rejected. Coordinate transformations used in the model remain supported; their named shapes concern model coordinates. The transformed endpoints need not belong to the fitted spline space or obey centring equalities. See shape_confidence_set() for explicit coefficient restrictions and the shape-constrained intervals article for examples.

Smoothing-parameter uncertainty is not currently supported for SCAM models; use unconditional = FALSE.

For explicit data, missing covariates needed by a smooth produce NA estimates and standard errors at those positions. Missing values in variables unrelated to that smooth do not discard its otherwise evaluable rows.

Raw data may contain x for a smooth such as s(log(x)); gratia evaluates the expression before constructing the prediction matrix. An evaluated column named "log(x)" can be supplied instead. When both are supplied, the raw inputs take precedence, except for stored model frames and grids prepared by gratia. Automatic grids are evenly spaced in the smooth coordinate (log(x)), and returned coordinate columns retain that name.

Stored evaluated columns allow automatic plotting even when a local function used to fit the model is no longer available. Evaluating that function at new raw data requires envir; training values are never substituted for new observations. Arbitrary transformations are not inverted to recover raw data. The coordinate_info attribute identifies retained raw coordinate columns. draw() uses them for back_transform = TRUE; the usual coordinate columns remain on the model scale. Automatic raw grids require recoverable fitting rows and matching functions and constants. Supplied raw data instead define the prediction rows directly. Typical covariate values added to evaluated grids by include_all = TRUE are not raw axis coordinates. Numeric by multipliers are set to one on automatic grids, even when their name also identifies a raw axis; supplied prediction rows retain their actual multipliers. Soap-film boundary rows retain model coordinates only. If a boundary axis is transformed, draw the estimates with back_transform = FALSE; raw-scale drawing gives an error rather than dropping the boundary.

Value

A data frame (tibble), which is of class "smooth_estimates". Its plot_variables attribute is a list of resolved covariate orders, named by smooth label, used when drawing the saved estimates. Its cyclic_info attribute retains the fitted cycle boundaries for supported one-dimensional cyclic components, allowing draw(object, cyclic = "polar") to configure the angular axis. This attribute is retained in nested and unnested output. When no evaluation rows remain, flat output has zero rows and retains the estimate, standard error and predictor columns with their original types. With unnest = FALSE, each selected smooth retains its metadata record and a data table with zero rows in the data list column.

Examples

```
load_mgcv()

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

```
## evaluate all smooths
smooth_estimates(m1)

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

# Direct evaluation of a GJRM distributional model
if (requireNamespace("GJRM", quietly = TRUE)) {
  set.seed(82)
  d <- data.frame(x = runif(100), z = runif(100))
  d$y <- sin(6 * d$x) + rnorm(100, sd = exp(0.3 * d$z))
  m <- GJRM::gamlss(list(y ~ s(x, k = 4), ~ s(z, k = 4)), data = d)
  estimates <- smooth_estimates(m, n = 50)
  draw(estimates)
}

# parallel processing of smooths
if (requireNamespace("mirai") && requireNamespace("carrier")) {
  library("mirai")
  daemons(2)                          # only low for CRAN requirements
  smooth_estimates(m1)
}
```

`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

```
## S3 method for class 'gamm4'
smooth_label(object, id, ...)

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.

See Also

gamm4_support

Examples

```
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

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

smooth_samples(model, ...)
```

```
## S3 method for class 'gam'
smooth_samples(
  model,
  ...,
  data = NULL,
  n_draws = NULL,
  chains = NULL,
  seed = NULL,
  method = c("gaussian", "mh", "inla", "user"),
  n_cores = 1L,
  n_threads = 1L,
  burnin = 1000,
  thin = 1L,
  t_df = 40,
  rw_scale = 0.25,
  frequentist = FALSE,
  unconditional = FALSE,
  draws = NULL,
  diagnostics = FALSE,
  mh_backend = c("mgcv", "gratia", "family"),
  mvn_method = c("mvnfast", "mgcv"),
  block_size = 10000L,
  dense_limit = 5e+06,
  select = NULL,
  n_grid = 200L,
  partial_match = NULL,
  envir = NULL
)
```

Arguments

model	a fitted model of the supported types; see diagnostic_support.
...	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.
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.
n_draws	positive integer; total retained draws across all chains. Required when generating draws. With method = "user", the supplied matrix determines the count; an explicit count must agree with its rows.
chains	positive integer or NULL; number of chains. NULL uses one chain, or the chain mapping in supplied draws. n_draws must be divisible by chains. Burn-in is applied separately to each MH chain.
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 as a coefficient matrix.
n_cores	positive integer; maximum concurrent MH chains using mirai, capped at chains. Existing workers are reused; otherwise temporary workers are created and cleaned up. Only used by method = "mh".
n_threads	positive integer; threads within Gaussian sampling using mvnfast. Defaults to one. This is separate from concurrent MH chains.
burnin	numeric; number of samples to discard as the burnin draws. Only used with method = "mh".
thin	positive integer; retain every thin iterations after burn-in.
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".
frequentist	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 frequentist == 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".
diagnostics	logical; calculate coefficient R-hat, effective sample sizes, and Monte Carlo standard errors using the optional posterior package.
mh_backend	character; "mgcv" preserves mgcv::gam.mh(). "gratia" uses compiled common-family likelihoods. "family" uses a compiled MH loop with callbacks to the fitted family object. Both support independent-observation bam fits. Smoothing parameters and parameters outside the fitted coefficient vector remain fixed. LSS scale and shape coefficients are sampled jointly with location. The family backend caches a full design matrix, smoothing penalty, and proposal factorisation once for all chains.
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.
block_size	positive integer; maximum observation rows per block in the gratia MH evaluator. Large designs are evaluated without storing a full matrix.
dense_limit	numeric; maximum number of design-matrix elements to cache for the gratia MH backend. Use zero to force blockwise evaluation.
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_grid	positive integer; resolution for constructing evaluation data if data not supplied
partial_match	logical; should smooths be selected by partial matches with select? If TRUE, select can only be a single string to match against.
envir	an optional environment for looking up data, functions and constants. Defaults to the model's evaluation environment.

Details

For LSS models, each smooth is a contribution to its own linear predictor. Select its full smooth label to distinguish predictors, for example s(x) and s.1(x). These are component draws on the predictor scale; use fitted_samples() with quantity = "expectation" for the expected response and derivative_samples() for its derivative. Coefficients are drawn jointly across all predictors.

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(nrow(data)), repeated if n_draws > 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.

Parallel calculation

With mirai and carrier installed, configure workers with mirai::daemons() to distribute calculations across selected smooths. difference_smooths() distributes factor-level comparisons instead. Without configured workers, or with only one calculation, evaluation is serial. These functions leave the caller's worker pool running.

Model selection, evaluation of model expressions and random draws are prepared in the caller. Workers receive the coordinates and coefficient columns needed for their calculation. Custom prediction methods are evaluated in the caller, including custom tensor margins and factor-smooth bases; workers receive their prediction matrices. Simultaneous intervals use successive draws in selection order, so `set.seed()` gives the same results with or without workers, independently of their number or completion order. `derivatives()` uses `set.seed()`; the other functions also accept `seed`, which restores the caller's random number state on exit.

`smooth_samples()` draws coefficients jointly once, then reuses those draws for every smooth, preserving dependence and chain identities. Sampling finishes before smooth calculations start, so MH chains and smooth workers are not nested. Sampling thread controls keep their existing meanings. Workers use their configured BLAS thread settings; when using a threaded BLAS, limit its threads before starting daemons to avoid competing threads.

Parallel work is most useful for expensive bases or interval calculations. Small grids can be slower because of transfer and scheduling costs. Large grids and many draws also require memory for worker inputs and returned results, in addition to the final output.

Author(s)

Gavin L. Simpson

See Also

gamm4_support

Examples

```
load_mgcv()

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

sms <- smooth_samples(m1, select = "s(x0)", n_draws = 5, seed = 42)

sms


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

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

smooth_terms *List the variables involved in smooths*

Description

[Experimental]

Usage

```
## S3 method for class 'gamm4'
smooth_terms(object, ...)

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

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.

See Also

gamm4_support

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

Description

Soap-film boundary and wiggly components use the sf.film and sf.wiggly classes introduced in mgcv 1.9-4.

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 'cpspline.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 'sf.wiggly'
smooth_type(smooth)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

## S3 method for class 'tesmi1.smooth'
smooth_type(smooth)

## S3 method for class 'tesmi2.smooth'
smooth_type(smooth)
```

```
## S3 method for class 'tesmd1.smooth'
smooth_type(smooth)

## S3 method for class 'tesmd2.smooth'
smooth_type(smooth)

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

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

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

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

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

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

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

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

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

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

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

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

Arguments

smooth	an object inheriting from class mgcv.smooth.

Details

SCAM smooth types use compact constraint descriptions: `inc` for increasing, `dec` for decreasing, and `pos` for positive. For tensor smooths, the two descriptions follow the covariate order in the smooth specification; `free` denotes an unconstrained direction. `Tensor int.` identifies an interaction. Explicit `By` bases share the type label of the corresponding ordinary basis.

`smooths` *Names of smooths in a GAM*

Description

Names of smooths in a GAM

Usage

```
## S3 method for class 'gamlss'
smooths(object, ...)

## S3 method for class 'gjrm'
smooths(object, ...)

smooths(object, ...)

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

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

## S3 method for class 'gamm4'
smooths(object, ...)
```

Arguments

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

Details

For GJRM models, labels follow equation and smooth order. Labels repeated in later equations receive suffixes such as `s.1(x)`. These labels can be supplied to `draw()` or `smooth_estimates()` to select smooths across equations.

`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,
  envir = NULL
)
```

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; if `TRUE` (and only if `frequentist == FALSE`) then the bayesian smoothing parameter uncertainty-corrected covariance matrix is returned, if available. Whether it is available depends on which smoothness selection method was used to fit the model.
`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; if `FALSE`, the default, the bayesian covariance matrix is returned, otherwise the frequentist covariance matrix.
`envir`	an optional environment supplying functions and constants used in model expressions. The available model formula environment is used when `NULL`. Covariate observations should be supplied in `data`.

`surrogate_residuals` *Surrogate residuals for categorical GAMs*

Description

Draw continuous residuals for models fitted with `mgcv::ocat()` or `mgcv::multinom()`, using the latent-variable methods of Liu and Zhang (2018) and Cheng, Wang and Zhang (2021). No additional package is required.

Usage

```
surrogate_residuals(model, seed = NULL, ...)

## S3 method for class 'gam'
surrogate_residuals(model, seed = NULL, conditioning = NULL, ...)

## Default S3 method:
surrogate_residuals(model, seed = NULL, ...)
```

Arguments

`model`	a fitted model object.
`seed`	integer or `NULL`; random seed for randomised residuals. With `NULL`, calculations use the current random-number state. An explicit seed preserves the caller's state, including when none was previously initialised.
`...`	arguments passed to other methods.
`conditioning`	character or `NULL`; only `"conditional"` is supported. The default, `NULL`, holds fitted random effects fixed in the same way.

Details

An ordered categorical model represents category `j` by an interval between two thresholds on a continuous latent scale. For each observation, a surrogate response is drawn from the fitted logistic distribution restricted to its observed category's interval. The residual subtracts the fitted latent mean from this surrogate response.

Equivalently, draw a probability uniformly between the fitted probabilities `Pr(Y < y)` and `Pr(Y <= y)`, then apply `stats::qlogis()`. Under the assumed model with known parameters, the residuals have a logistic distribution with location zero and scale one: mean zero and variance `pi^2 / 3`. Fitted parameters affect this reference approximation. The identity link reported by `ocat()` describes the latent mean; its latent distribution is logistic.

This calculation shares the randomised probabilities used by `quantile_residuals()`, which instead applies `qnorm()` for normal scores. Both tails are calculated using log probabilities to retain accuracy when ordinary probabilities round to zero or one. Fitting weights do not change the scale of the latent logistic distribution.

For a nominal multinomial response, there is a utility for every category, including category 0. Its fitted locations are `(0, eta_1, ..., eta_K)`, including offsets and fitted random effects,

with independent standard Gumbel errors having CDF `exp(-exp(-x))`. Draw the complete utility vector conditional on the observed category being its maximum. Subtract each location and Euler's constant (`-digamma(1)`) to obtain the residual vector. Under the known-parameter model, each component has a centred Gumbel reference with mean zero and variance `pi^2 / 6`; the joint null consists of independent centred Gumbel components. The conditional draw given an observed category preserves dependence among its utilities. Differences from the reference utility would have dependent logistic marginals. Even for `K = 1`, this function retains both Gumbel utility components.

The maximum has a Gumbel distribution with location equal to the log-sum-exp of the fitted locations, independently of the winning label. Conditional on that maximum and the observed winner, the other utilities have independent upper-truncated Gumbel distributions. The sampler uses this identity directly, with log calculations that retain accuracy for rare observed categories and extreme predictors, without rejection or Gibbs iterations. This construction does not impose an ordering on categories.

Supported fits are `gam` and `bam` models with a single `ocat()` family, and `gam` models with `multinom()`. Grouped families and other model classes are not supported. Fitted random effects are held fixed. Use `type = "surrogate"` in `appraise()`, `qq_plot()`, `worm_plot()` and residual plots. QQ and worm plots require `method = NULL`, with logistic references for `ocat()` and centred Gumbel references for `multinom()`. Pointwise order-statistic bands apply separately to each utility component, treating fitted parameters as fixed. They do not give simultaneous coverage across categories. Components are always shown separately, including when `by_family = FALSE`. For `ocat()`, `residuals_linpred_plot()` uses the fitted latent mean. For `multinom()`, it uses each category's fitted probability; observed-versus-fitted panels compare that probability with its category indicator. This also gives category 0 a varying horizontal axis despite its fixed utility location. Covariate plots separate the category utilities. The ordered correlation helpers do not yet accept multinomial utilities.

Ordinary and scalable plots use exactly the same conditional draw for a given seed. `appraise()` reuses one draw across all its panels. Scalable plots retain the residual matrix and process one component at a time before combining display summaries; observations, rather than utility components, determine automatic dispatch thresholds. Changing sampler chunk sizes does not change the draw. QQ binning and density grids change only the display.

Each call draws one residual vector per observation. Set `seed` for reproducibility; repeating plots with different seeds helps assess whether a pattern persists across residual draws. An explicit seed preserves the caller's random-number state. With `seed = NULL`, sampling uses and advances the current state.

Value

For `ocat()`, a numeric vector with one residual per observation. For `multinom(K)`, a numeric matrix with one row per observation and `K+1` columns named `"0"`, ..., `"K"`, identifying the category utilities. With `na.exclude`, excluded fitting rows are restored as `NA`; with `na.omit`, only observations used for fitting are returned.

References

Cheng, C., Wang, R. and Zhang, H. (2021). Surrogate Residuals for Discrete Choice Models. Journal of Computational and Graphical Statistics, 30(1), 67–77. doi:10.1080/10618600.2020.1775618.

Liu, D. and Zhang, H. (2018). Residuals and Diagnostics for Ordinal Regression Models: A Surrogate Approach. Journal of the American Statistical Association, 113(522), 845–854. doi:10.1080/01621459.2017.1292915.

See Also

quantile_residuals(), appraise(), diagnostic_support

Examples

```
load_mgcv()
set.seed(359)
dat <- data.frame(x = runif(200, -2, 2))
latent <- sin(dat$x) + rlogis(nrow(dat))
dat$y <- findInterval(latent, c(-1, 0, 2)) + 1L
fit <- gam(y ~ s(x), data = dat, family = ocat(R = 4), method = "REML")
head(surrogate_residuals(fit, seed = 42))
appraise(fit, type = "surrogate", seed = 42)
```

---

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

```
## S3 method for class 'gamm4'
term_names(object, ...)

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

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

See Also

gamm4_support

`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

```
## S3 method for class 'gamm4'
term_variables(object, ...)

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

term_variables(object, term, ...)

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

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

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

Arguments

`object`	an R object on which method dispatch is performed
`...`	arguments passed to other methods.
`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.

Value

A character vector of variable names.

See Also

gamm4_support

theta *General extractor for additional parameters in mgcv models*

Description

General extractor for additional parameters in mgcv models

Usage

```
theta(object, ...)

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

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

## S3 method for class 'gamm4'
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

See Also

gamm4_support

Examples

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

tidy_basis *A tidy basis representation of a smooth object*

Description

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

Usage

```
tidy_basis(
  smooth,
  data = NULL,
  at = NULL,
  coefs = NULL,
  p_ident = NULL,
  envir = 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`	retained for compatibility. Fitted SCAM smooths use their stored coefficient constraints to match basis columns to coefficients.
`envir`	an optional environment supplying functions and constants used in model expressions. The available model formula environment is used when `NULL`. Covariate observations should be supplied in `data`.

Value

A tibble.

Author(s)

Gavin L. Simpson

Examples

```
load_mgcv()

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

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

# tidy representaition 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))
```

---

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`

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

Description

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

Usage

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

Arguments

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

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`

`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

```
typical_values(object, ...)

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

## S3 method for class 'data.frame'
typical_values(object, vars = everything(), ...)

## S3 method for class 'gamm4'
typical_values(object, ...)
```

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.
data	an optional data frame supplying covariate classes. By default, these are taken from the stored model frame; typical values come from the fitted model's covariate summaries.

See Also

gamm4_support

user_draws *Handle user-supplied posterior draws*

Description

Handle user-supplied posterior draws

Usage

```
## S3 method for class 'gamm4'
user_draws(model, ..., draws, n_draws = NULL, chains = NULL)

user_draws(model, ..., draws)

## S3 method for class 'gam'
user_draws(model, ..., n_draws = NULL, chains = NULL, draws, index = NULL)

## S3 method for class 'scam'
user_draws(
  model,
  ...,
  draws = NULL,
  n_draws = NULL,
  chains = NULL,
  index = NULL
)
```

Arguments

model	a fitted model from mgcv::gam(), mgcv::bam(), scam::scam(), or gamm4::gamm4(), or a model inheriting from these classes. See gamm4_support for sampling from gamm4 fits.
...	arguments passed to methods.
draws	matrix; user supplied posterior draws to be used when method = "user".
n_draws	positive integer; total retained draws across all chains. Required when generating draws. With method = "user", the supplied matrix determines the count; an explicit count must agree with its rows.
chains	positive integer or NULL; number of chains. NULL uses one chain, or the chain mapping in supplied draws. n_draws must be divisible by chains. Burn-in is applied separately to each MH chain.
index	a vector to index (subset) the columns of draws.

Details

The supplied draws must be a matrix (currently), with 1 column per model coefficient, and 1 row per posterior draw. The "gam" method has argument index, which can be used to subset (select) coefficients (columns) of draws. index can be any valid way of selecting (indexing) columns of a matrix. index is useful if you have a set of posterior draws for the entire model (say from mgcv::gam.mh()) and you wish to use those draws for an individual smooth, via smooth_samples().

For SCAM models, supplied draws must contain finite transformed coefficients in the order returned by coef(model). If column names are supplied they must match exactly. Entries identified by model$p.ident must be nonnegative. Draws returned with parametrized = FALSE cannot be supplied directly for prediction; transform them first.

See Also

gamm4_support

`variance_comp` *Variance components and fitted term variances for GAMs*

Description

Extract variance components for smooth penalties and random effects, or calculate the variances and covariances of fitted term contributions.

Usage

```
## S3 method for class 'gamm4'
variance_comp(
  object,
  rescale = TRUE,
  coverage = 0.95,
  interval_method = c("profile", "none"),
  type = c("penalty", "terms"),
  n_draws = 1000L,
  seed = NULL,
  unconditional = FALSE,
  include_total = TRUE,
  ...
)

## S3 method for class 'scam'
variance_comp(
  object,
  rescale = TRUE,
  coverage = 0.95,
  type = c("penalty", "terms"),
  n_draws = 1000L,
  seed = NULL,
  unconditional = FALSE,
  include_total = TRUE,
  ...
)

variance_comp(object, ...)

## S3 method for class 'gam'
variance_comp(
  object,
  rescale = TRUE,
```

```
  coverage = 0.95,
  type = c("penalty", "terms"),
  n_draws = 1000L,
  seed = NULL,
  unconditional = FALSE,
  include_total = TRUE,
  ...
)

## S3 method for class 'gamm'
variance_comp(
  object,
  rescale = TRUE,
  coverage = 0.95,
  type = c("penalty", "terms"),
  n_draws = 1000L,
  seed = NULL,
  unconditional = FALSE,
  include_total = TRUE,
  ...
)
```

Arguments

object	a model fitted by mgcv::gam(), mgcv::bam(), mgcv::gamm(), or gamm4::gamm4(). See gamm4_support for mixed-model variance components.
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.
interval_method	character; for gamm4() fits, "profile" computes likelihood profile intervals and "none" returns estimates only.
type	character; "penalty" (the default) returns variance parameters associated with penalties and random effects. "terms" returns sample variances of fitted term contributions and their covariance matrix.
n_draws	integer; number of joint posterior coefficient draws for type = "terms". Defaults to 1000. Use zero for point estimates only; otherwise at least two draws are required. Ignored for type = "penalty".
seed	optional random seed for term intervals. Supplying a seed preserves the caller's random-number state.
unconditional	logical; include the available correction for smoothing-parameter uncertainty in term intervals. Defaults to FALSE. If the corrected covariance is unavailable, a warning is issued and the uncorrected covariance is used. SCAM fits require FALSE.

include_total — logical; include total rows for type = "terms"? Defaults to TRUE. Ignored for type = "penalty".

... — arguments passed to other methods. For gamm4() fits, additional arguments are passed to stats::profile(), for example maxpts. With type = "terms", no additional arguments are accepted.

Details

For SCAM fits with type = "penalty", .variance is the fitted response scale divided by the smoothing parameter, optionally undoing penalty rescaling. This ratio describes the Gaussian penalty on unrestricted fitting parameters; it is not a Gaussian variance for the transformed smooth coefficients. SCAM's GCV and UBRE fits do not supply the likelihood-based smoothing-parameter covariance needed for confidence limits, so both limit columns are NA.

With type = "penalty", models fitted by mgcv::gam() and mgcv::bam() have one row per penalty, including penalties supplied through paraPen. Fixed smoothing parameters are included, and terms sharing a smoothing parameter each have their own rows. A smooth with several penalties has several rows. Unpenalized parametric terms have no variance parameter and are omitted.

The variance component is the fitted scale parameter divided by the smoothing parameter. With rescale = TRUE, the smoothing parameter is first divided by the penalty's scaling factor. These components describe the penalty on coefficients, rather than the variation in fitted term values over the observations.

Confidence intervals use the Hessian of the likelihood-based smoothness selection criterion, transformed to the log standard deviation scale. Shared smoothing parameters retain their shared uncertainty. A fixed smoothing parameter can still have an interval when the model scale is estimated. The scale is included as a separate row when its uncertainty is available. Unavailable intervals are NA.

For mgcv::gamm() fits, components are extracted from lme. Smooth variances are expressed on the same penalty scale as for gam() and bam(), with one row per penalty. Additional random effects have one row per coefficient variance, labelled by grouping factor and coefficient; their covariances are omitted. Intervals use nlme::intervals() when the fitted covariance parameters support this calculation. Bounds are NA when that information is unavailable, including fits containing pdTens tensor-product blocks. The estimated residual scale is labelled scale; a fixed scale is omitted. With residual variance weights, scale is the baseline variance before those weights are applied. For models fitted by penalized quasi-likelihood (PQL), these are approximate components of the working mixed model.

For gamm4::gamm4() fits, components are extracted from mer. Smooth components use the same penalty scale and labels as the other methods; additional random effects retain their mixed-model variances. By default, intervals are computed separately for each standard deviation using stats::profile() and stats::confint(). The profiling uses the fitted random-effects design matrices. Gaussian models are refitted by maximum likelihood for these profiles, including models fitted by REML; the reported point estimates remain those of the original fit. Profiling can take substantially longer than extracting estimates. Use interval_method = "none" for estimates without intervals. Failed profiles warn and return NA bounds for the affected component. Profiles are unavailable for generalized mixed models with an estimated scale.

With type = "terms", each component is a fitted term's contribution to the linear predictor, evaluated over the observations used to fit the model. Parametric formula terms are included, with all coefficients of a factor, polynomial or interaction combined into one term. Each smooth has one row,

including smooths with several penalties. Additional random effects in `gamm()` and `gamm4()` contribute one component per grouping factor and coefficient, evaluated at their fitted values. Formula terms follow the model's contrast coding. Duplicated component labels receive numeric suffixes.

Term variances and covariances use equal observation weights and the sample denominator, the number of fitting observations minus one. Likelihood weights do not change this averaging. Intercepts, offsets and residual variation are excluded. `rescale` and `interval_method` apply only to `type = "penalty"`; no profiling is performed for term variances.

A final row labelled `total` gives the variance of the sum of the term contributions, including additional fitted random effects. It includes covariance between terms: the total is the sum of all entries of their covariance matrix, rather than just its diagonal. Models with several linear predictors have one total per predictor, labelled `total`, `total.1`, and so on in predictor order. A term shared by several predictors contributes to each of their totals. An intercept-only predictor has total variance zero. These labels are reserved; a formula term with the same name receives a numeric suffix so it remains distinct from the total. Use `include_total = FALSE` to omit the total rows and their columns in the posterior draw matrices. Totals are still calculated for proportions; the individual term estimates and intervals are unchanged.

Proportions allocate the total variance using

$$p_j = \frac{\mathrm{Cov}(f_j, \sum_k f_k)}{\mathrm{Var}(\sum_k f_k)}.$$

Each term receives its own variance plus its covariance with the other terms. This splits each pair's covariance contribution equally between its two terms, so the individual proportions sum to one. Proportions can be negative or greater than one when terms offset each other's variation; they are not restricted to the interval from zero to one.

`.proportion` reports these allocations, with equal-tailed posterior bounds in `.proportion_lower_ci` and `.proportion_upper_ci`. These bounds are on the proportion scale. Each posterior proportion uses the term contributions and total from the same joint draw. Individual interval limits need not sum to one. The total row's proportion is one. For additional linear predictors, the columns receive suffixes `.1`, `.2`, and so on, matching the total labels; for example, `.proportion.1` and `.proportion_lower_ci.1`. A shared term has a separate allocation for each predictor. Entries for unrelated terms and totals are `NA`.

Proportions are undefined when the total variance is zero and are returned as `NA`, including for an intercept-only predictor. Posterior draws with zero total variance retain `NA` proportions and are omitted when computing proportion intervals; if none are defined, both bounds are `NA`.

Term intervals are equal-tailed posterior intervals, computed by drawing the full coefficient vector from its approximate Gaussian posterior and recalculating each term's variance over the same fitting observations. All terms use the same coefficient draw in each replicate, preserving their dependence. Each total is calculated by summing the term contributions within that draw and then computing their variance. Additional fitted random effects in mixed models are drawn conditional on the GAM coefficients, retaining their covariance with the GAM terms and with each other. Smoothing parameters, random-effect covariance parameters and the response scale are held at their fitted values, unless `unconditional = TRUE` supplies a correction for smoothing-parameter uncertainty in the GAM coefficient covariance. PQL and generalized mixed models use the Gaussian approximation at the fitted working model. For SCAM fits, draws are made jointly on the unrestricted fitting scale and transformed before evaluating the terms.

Point estimates remain the variances of the fitted contributions. Interval bounds are quantiles of the posterior term standard deviations; square the bounds to obtain intervals for the term variances. Use `seed` for reproducible draws and `n_draws = 0` to skip intervals and return `NA` bounds.

The `covariance` attribute contains the covariance matrix of the fitted term contributions, with rows and columns in `.component` order, excluding the total rows. Its diagonal equals `.variance` for the individual terms. For a model with a single linear predictor, the sum of all matrix entries equals the `total` row's `.variance`. Adding only the diagonal omits covariance between terms. This matrix describes variation across observations, rather than estimation uncertainty. When intervals are computed, the `variance_draws` attribute contains a matrix with one row per posterior draw and one column per component, on the variance scale. Rows are paired across terms. Its column names match `.component`; the `coverage` attribute records the requested interval coverage. The total columns include covariance between term contributions within each draw. Summing the individual term variances would omit that covariance. The `proportion_draws` attribute is a named list with one matrix per total (`total`, `total.1`, and so on). Each matrix has the same rows and columns as `variance_draws`, with `NA` entries for unrelated components. Its rows retain the pairing with the coefficient and variance draws. Proportions and their intervals are added only for `type = "terms"`. The `term_groups` attribute lists the component labels belonging to each predictor, named by its total label. Use `draw.variance_comp()` to plot components or proportions with their stored intervals.

Value

A tibble of class `variance_comp`, with `.component`, `.variance`, `.std_dev`, `.lower_ci` and `.upper_ci`. Both confidence limits are on the standard deviation scale. For `type = "terms"`, obtain the named covariance matrix with `attr(result, "covariance")` and posterior variance draws with `attr(result, "variance_draws")` when intervals were computed. Term results also include proportion estimates and their interval columns; `attr(result, "proportion_draws")` contains the paired posterior proportions.

See Also

gamm4_support

`mgcv::gam.vcomp()`, `draw.variance_comp()`

---

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

```
## S3 method for class 'gamm4'
which_smooths(object, ...)

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

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.

See Also

gamm4_support

worm_plot *Worm plot of model residuals*

Description

Worm plot of model residuals

Usage

```
worm_plot(model, ...)

## S3 method for class 'gam'
worm_plot(
  model,
  method = NULL,
  type = NULL,
  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 = "grey20",
  point_alpha = 1,
  line_col = "red",
  seed = NULL,
  by_family = NULL,
  conditioning = NULL,
  ...,
  n_dharma = 250L
)

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

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

## S3 method for class 'gamm4'
worm_plot(model, conditioning = NULL, ...)

## S3 method for class 'gjrm'
worm_plot(model, ..., type = "quantile", seed = NULL, response = NULL)
```

```
## S3 method for class 'gamm'
worm_plot(model, conditioning = NULL, ...)

## S3 method for class 'gratia_diagnostic_proxy'
worm_plot(model, type = "quantile", ...)

## S3 method for class 'lme'
worm_plot(model, conditioning = NULL, ...)

## S3 method for class 'merMod'
worm_plot(model, conditioning = NULL, ...)

## S3 method for class 'gamlss'
worm_plot(model, ..., type = "quantile", seed = NULL)

## S3 method for class 'sdmTMB'
worm_plot(
  model,
  conditioning = NULL,
  component = "response",
  field_method = c("mle-mvn", "mle-eb", "mle-mvn-integrated"),
  ...
)
```

Arguments

`model`	a fitted model supported by diagnostic_support.
`...`	arguments passed to other methods.
`method`	character or `NULL`; how to calculate reference quantiles. Probability integral transform (PIT) and quantile residuals use fixed uniform and standard-normal references, respectively. For quantile residuals, `"normal"` also selects that fixed reference; other named methods require traditional residuals. For deviance, response and Pearson residuals:

- `"uniform"` applies the fitted inverse cumulative distribution function (CDF) to randomised uniform quantiles and averages the resulting ordered residuals over `n_uniform` draws.
- `"simulate"` generates `n_simulate` response data sets at the fitting covariates. Responses are generated together when needed to preserve dependence, and their ordered residuals supply reference values and bands.
- `"marginal"` approximates the pooled distribution of individual residuals, without bands for dependence. This is selected for BAM GEE and PQL correlations that do not define a joint response distribution. Explicit joint simulation is unavailable in those cases.
- `"normal"` uses a normal approximation fitted to the residual location and scale. For dependent responses it omits bands based on independence.
- `"direct"` is deprecated in favour of `"uniform"`.

	For ordinary GAMs, NULL selects "simulate" for Tweedie and "uniform" otherwise. Missing inverse CDFs cause a switch to simulation; missing response generators can cause a switch to a normal approximation for independent responses. Dependent responses use joint simulation when available, or marginal checks otherwise. For grouped families, NULL selects simulation if every component has a generator, or inverse-CDF sampling with a message if every component has an inverse CDF. If neither is available, it gives an error; a normal approximation must be requested explicitly.
type	character or NULL; residual type: "deviance", "response", "pearson", "pit", "quantile", "dharma", or "surrogate" (for ocat() or multinom() GAMs). NULL uses the model-specific default: deviance for ordinary lm/glm/GAM fits, and quantile for supported grouped families, GJRM, sdmTMB and full mixed-model checks. glmmTMB uses Pearson residuals conditionally and response residuals marginally. Extracted GAM components follow the ordinary GAM defaults. See diagnostic_support.
n_uniform	numeric; number of inverse-CDF draws to average when 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. Used for quantile reference bands and traditional method = "simulate" or "normal" bands.
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	Deprecated; use gratia_theme_get() theme settings instead. Fill colour for reference intervals, including bands for quantile residuals.
ci_alpha	Deprecated; use gratia_theme_get() theme settings instead. Alpha transparency for reference intervals.
point_col	Deprecated; use gratia_theme_get() theme settings instead. Constant point colour when family components are not mapped to colour. Pooled grouped diagnostics instead colour points by component.
point_alpha	Deprecated; use gratia_theme_get() theme settings instead. Alpha transparency of points on the QQ plot.
line_col	Deprecated; use gratia_theme_get() theme settings instead. Colour used to draw the reference line.
seed	numeric; the random number seed to use for method = "simulate" and method = "uniform", and for randomised quantile residuals.

by_family	logical or NULL; separate indexed grouped-family components into facets. The default pools quantile residuals with component colours and facets traditional residuals. TRUE and FALSE explicitly select the layout. In appraise(), observed-versus-fitted values always use component facets. Standalone observed_fitted_plot() facets by default and permits explicit pooling. Ignored for ordinary models.
conditioning	character or NULL; "conditional" holds fitted random effects fixed, "marginal" averages over new effects, and "gam_component" uses the extracted GAM. Availability and defaults depend on the model; see diagnostic_support.
n_dharma	positive integer; number of response simulations for type = "dharma". This is separate from n_simulate, which controls traditional QQ reference calculations. Defaults to 250.
response	NULL uses all responses; a single integer selects one. For mgcv::gfam() models, the integer indexes the family list and selects observations assigned to that component, including when families repeat. For joint GJRM models, it selects a response in model order. Selection is independent of by_family.
component	character; for delta sdmTMB models, select the complete "response", occurrence ("binomial"), or "positive" component. Positive-component checks use only observations with positive responses.
field_method	how to treat fitted sdmTMB fields: hold one posterior draw fixed ("mle-mvn"), hold estimates fixed ("mle-eb"), or average over fitted-field uncertainty ("mle-mvn-integrated"). See quantile_residuals() and diagnostic_support.

Details

See appraise() for grouped-family defaults, supported distributions, pooling and separate component panels. PIT and quantile residuals use fixed uniform and standard-normal reference quantiles. For ideal independent responses, bands are pointwise intervals for ordered residuals. For dependent responses, bands use simulations that preserve dependence when available; otherwise they are omitted. Residuals calculated from simulation ranks also use simulated bands. See diagnostic_support for these choices and limits. Bands condition on fitted parameters and exclude fitting uncertainty; they do not provide simultaneous coverage. Traditional residuals use the fitted response distributions or a normal approximation.

For mgcv::ocat() fits, type = "surrogate" uses logistic reference quantiles with location zero and scale one. Set method = NULL. Independent observations have approximate pointwise bands obtained by applying qlogis() to beta order-statistic limits. Worm plots subtract the same logistic reference quantiles from residuals and bands. These references treat fitted parameters as fixed. For mgcv::multinom() GAMs, surrogate residuals retain a utility component for every category, including 0. Components have separate facets and centred Gumbel reference quantiles, including when by_family = FALSE. The same reference is subtracted in worm plots, and pointwise beta order-statistic bands use the centred Gumbel quantile function separately for each component. Ordinary and scalable displays share their draw. See surrogate_residuals().

Subtitles name the method. Captions identify conditioning and field choices, joint simulation bands or omitted bands. Routine pointwise bands are unlabelled. Supply caption to replace this text, or caption = "" to omit it. Ordinary and scalable QQ and worm plots use the same labels.

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

gamm4_support

Examples

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

## The default uses inverse-CDF sampling of randomised uniform quantiles.
## Note no reference bands are drawn with this method.
worm_plot(m)

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

## ... or request a normal approximation for the residuals.
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, Kegnonsa, 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

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

References

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

Index

∗ **data**
bird_move, 41
gss_vocab, 184
ref_sims, 257
smallAges, 295
zooplankton, 341
∗ **draw methods**
draw.rootogram, 130
∗ **utility**
boundary, 42
n_eta, 206

add_confint, 6
add_constant, 6
add_fitted, 7
add_fitted.gam, 8
add_fitted_samples, 10
add_partial_residuals, 12
add_posterior_samples *(*add_fitted_samples*)*, 10
add_predicted_samples *(*add_fitted_samples*)*, 10
add_residuals, 13
add_residuals.gam, 14
add_sizer, 15
add_smooth_samples *(*add_fitted_samples*)*, 10
ale, 16
ale(), *94*
all_levels *(*evenly*)*, 153
appraise, 19
appraise(), *80*, *87*, *156*, *183*, *210*, *229*, *231*, *245*, *253, 254*, *262–267*, *277–280*, *321, 322*, *340*
appraise.glmmTMB *(*qq_plot.glmmTMB*)*, 247
as_mcmc_draws, 26
as_mcmc_draws(), *133*, *178*, *194*, *200*, *227*, *274, 275*
assemble, 27
assemble(), *33*, *108*, *114*, *118*, *127*, *137*, *143, 144*, *284*
assemble.gam, 28
assemble.gam(), *120*
assemble.gamlss *(*draw.gamlss*)*, 118
assemble.gamm *(*assemble.gam*)*, 28
assemble.gjrm *(*draw.gamlss*)*, 118
assemble.glmmTMB *(*qq_plot.glmmTMB*)*, 247
assemble.scam *(*assemble.gam*)*, 28
assemble.sdmTMB *(*sdmTMB-methods*)*, 281

base::levels(), *69*
base::seq(), *153*
base::set.seed(), *61*
base::zapsmall(), *223*
basis, 37
basis(), *95, 96*, *123*
basis_size, 40
bird_move, 41
boundary, 42
by_level *(*is_by_smooth*)*, 186
by_variable *(*is_by_smooth*)*, 186

check_is_mgcv_smooth *(*is_mgcv_smooth*)*, 188
check_user_select_smooths, 42
coef.scam, 43
compare_smooths, 44
compare_smooths(), *97, 98*
concrvity *(*model_concurvity*)*, 201
concrvity(), *125*
conditional_differences, 46
conditional_differences(), *48*, *53*, *99*
conditional_moments, 50
conditional_moments(), *101*, *162*
conditional_values, 52
conditional_values(), *19*, *47, 48*, *50–53*, *102*
confint.fderiv, 54
confint.gam, 56

confint.gam(), *295*
confint.gamm *(*confint.gam*)*, 56
confint.gamm4 *(*confint.gam*)*, 56
confint.scam *(*confint.gam*)*, 56

data_combos, 59
data_sim, 60
data_sim(), *63*, *65–67*, *257*
data_sim_families, *61*, 63, *63*, *67*
data_sim_shapes, *62*, *63*, *65*, 66
data_slice, 67
data_slice(), *69*, *117*
derivative_samples, 70
derivative_samples(), *26*, *311*
derivatives, 75
derivatives(), *15*, *92*, *295*
dharma_residuals, 79
dharma_residuals(), *24*, *86*, *259*
diagnostic_support, *21–25*, *79*, 80, *156*, *164*, *175*, *210*, *230*, *233*, *240*, *243–245*, *248*, *249*, *253*, *255*, *260–262*, *265–267*, *275*, *277*, *278*, *283*, *290*, *309*, *322*, *338–340*
difference_smooths, 87
difference_smooths(), *46*, *78*, *91*, *219*, *295*, *311*
dispersion, 92
draw, 93
draw(), *6*, *28*, *39*, *40*, *90*, *118*, *143*, *144*, *182*, *183*, *284*, *304*, *306*, *318*
draw.ale, 94
draw.ale(), *19*
draw.basis, 95
draw.compare_smooths, 97
draw.compare_smooths(), *45*
draw.conditional_differences, 99
draw.conditional_moments, 100
draw.conditional_values, 101
draw.conditional_values(), *52*, *99*
draw.derivatives, 103
draw.difference_smooth, 105
draw.evaluated_parametric_term, 107
draw.gam, 108
draw.gam(), *35*, *36*, *96*, *122*, *139*, *213*, *249*, *282*, *283*, *295*
draw.gamlss, 118
draw.gamm *(*draw.gam*)*, 108
draw.gjrm *(*draw.gamlss*)*, 118
draw.glmmTMB *(*qq_plot.glmmTMB*)*, 247
draw.mgcv_smooth, 123
draw.overall_concurvity *(*draw.pairwise_concurvity*)*, 124
draw.pairwise_concurvity, 124
draw.parametric_effects, 125
draw.parametric_effects(), *284*
draw.partial_derivatives *(*draw.derivatives*)*, 103
draw.penalty_df, 129
draw.rootogram, 130
draw.sampling_diagnostics, 132
draw.sampling_diagnostics(), *274*, *275*
draw.scam *(*draw.gam*)*, 108
draw.scasm *(*draw.gam*)*, 108
draw.sdmTMB *(*sdmTMB-methods*)*, 281
draw.smooth_estimates, 134
draw.smooth_samples, 139
draw.variance_comp, 141
draw.variance_comp(), *335*
draw_label_control, 143
draw_label_control(), *27*, *36*, *93*, *94*, *96*, *98*, *100–102*, *104*, *106*, *108*, *118*, *122*, *124*, *125*, *128*, *130*, *131*, *133*, *139*, *141*, *143*, *250*, *284*

edf, 145
eval_smooth, 146
evaluate_parametric_term, 151
evaluate_parametric_term(), *107*
evaluate_smooth, 152
evenly, 153
evenly(), *69*
extract_link *(*link*)*, 190

factor_combos, 154
family(), *159*, *185*, *191*
family.bam *(*family.gam*)*, 154
family.gam, 154
family.gamm *(*family.gam*)*, 154
family.gamm4 *(*family.gam*)*, 154
family_diagnostics, *87*, 155, *194*
family_name, 157
family_parameters, 158
family_parameters(), *166*, *170*
family_type, 159
fitted_moments, 159
fitted_moments(), *50*, *51*
fitted_samples, 162

fitted_samples(), *10, 11*, *26*, *48, 49*, *73*, *158*, *161, 162*, *170, 171*, *175*, *256*, *311*
fitted_values, 168
fitted_values(), *53*, *57*, *158*, *162*, *256*
fix_offset, 173
fixed_effects, 174

gamm4::gamm4(), *155*, *174, 175*, *177*, *191*, *199*, *225*, *291*, *325*, *330*, *332, 333*
gamm4_support, *8*, *12*, *14*, *25*, *27*, *41, 42*, *50*, *53*, *59*, *74*, *78*, *87*, *93*, *146*, *152*, *154*, *167*, *172*, 175, *177, 178*, *187*, *197*, *199*, *201–203*, *206–208*, *211*, *214, 215*, *219*, *223*, *225*, *228*, *236*, *241*, *246*, *255*, *271*, *273*, *293*, *300*, *308*, *312, 313*, *323, 324*, *329–332*, *335, 336*, *341*
gaussian_draws, 176
generate_draws *(*post_draws*)*, 224
get_by_smooth, 179
get_smooth, 179
get_smooth(), *283*
get_smooths_by_id, 180
get_smooths_by_id(), *283*
get_smooths_by_id.glmmTMB *(*qq_plot.glmmTMB*)*, 247
get_smooths_by_id.sdmTMB *(*sdmTMB-methods*)*, 281
get_vcov, 180
get_vcov(), *17*
ggcube::coord_3d(), *33*, *100*, *102*, *113*, *127*, *136*
ggplot2::continuous_scale(), *280*
ggplot2::coord_radial(), *35*, *116*, *138*
ggplot2::coord_sf(), *32*, *113*, *136*
ggplot2::element_geom(), *183*
ggplot2::facet_grid(), *53*
ggplot2::facet_wrap(), *52*, *99*, *101, 102*, *133*
ggplot2::geom_contour(), *31*, *96*, *98*, *100*, *102*, *106*, *112*, *127*, *135*, *140*
ggplot2::geom_line(), *136*, *140*
ggplot2::ggplot(), *27, 28*, *36*, *93, 94*, *108*, *118*, *124*, *130*, *133*, *142*, *272*
ggplot2::guide_axis(), *32*, *96*, *104*, *106*, *113*, *123*, *127*, *136*, *141, 142*
ggplot2::guides(), *280*
ggplot2::label_both(), *95*, *123*
ggplot2::labs(), *94, 95*, *108*, *123*, *129, 130*, *140*, *210*, *229*, *244*, *262*, *265, 266*, *278*, *339*
ggplot2::theme(), *183*
GJRM::gamlss(), *118*, *120*, *302*, *304*
GJRM::gjrm(), *118*, *120*, *302*, *304*
glmmTMB-methods *(*qq_plot.glmmTMB*)*, 247
graphics::hist.default(), *273*
graphics::par(), *22*
gratia_palette, 181
gratia_theme_get, 182
gratia_theme_get(), *22, 23*, *31*, *33*, *94*, *96*, *104*, *106*, *112*, *114*, *125–128*, *131*, *133*, *135–137*, *140*, *142–144*, *210*, *229*, *245*, *262*, *267*, *278*, *339*
gratia_theme_set *(*gratia_theme_get*)*, 182
gratia_theme_update *(*gratia_theme_get*)*, 182
grid::unit(), *144*
gss_vocab, 184
gw_f0, 184
gw_f1 *(*gw_f0*)*, 184
gw_f2 *(*gw_f0*)*, 184
gw_f3 *(*gw_f0*)*, 184
gw_functions, *61*
gw_functions *(*gw_f0*)*, 184

has_theta, 185

in_shape_set *(*shape_confidence_set*)*, 285
inv_link *(*link*)*, 190
is_by_smooth, 186
is_continuous_by_smooth *(*is_by_smooth*)*, 186
is_factor_by_smooth *(*is_by_smooth*)*, 186
is_factor_term, 187
is_mgcv_family, 188
is_mgcv_smooth, 188
is_mrf_smooth *(*is_mgcv_smooth*)*, 188
is_multivariate_y, 189
is_offset, 189

KernSmooth::bkde2D(), *23*, *263*, *279*

level *(*ref_level*)*, 257
level(), *69*
link, 190
link(), *17*
load_mgcv, 192

log_lik, 192
log_lik(), *155, 156*, *196*, *238, 239*, *275*
loo.bam *(*loo.gam*)*, 195
loo.gam, *194*, 195
loo.scam *(*loo.gam*)*, 195
loo::loo(), *195*
loo::waic(), *195*
lp_matrix, 196

marginaleffects::plot_predictions(), *52*
mgcv::bam(), *16*, *41*, *150*, *155*, *160*, *169*, *174*, *177*, *191*, *199*, *207*, *225*, *255*, *263*, *307*, *313*, *318, 319*, *330*, *332, 333*
mgcv::exclude.too.far(), *31*, *45*, *112*, *150*, *303*, *327*
mgcv::family.mgcv(), *65*
mgcv::fix.family.rd(), *292*
mgcv::gam(), *16*, *29*, *41*, *103*, *105*, *110*, *129*, *135*, *140*, *150*, *155*, *160*, *169*, *174*, *177*, *191*, *199*, *203*, *207*, *225*, *255*, *291*, *307*, *313*, *318, 319*, *330*, *332, 333*
mgcv::gam.mh(), *73*, *165*, *199*, *226*, *234*, *310*, *330*
mgcv::gam.vcomp(), *256*, *335*
mgcv::gamm(), *28*, *41*, *108*, *155*, *174*, *191*, *207*, *222*, *291*, *313*, *318*, *325*, *332, 333*
mgcv::gammals(), *191*, *235*
mgcv::gamSim(), *60*
mgcv::gaulss(), *191*, *235*
mgcv::gevlss(), *191*, *235*
mgcv::gfam(), *24*, *62*, *83*, *165*, *170*, *229*, *235*, *245*, *253, 254*, *292*, *340*
mgcv::gumbls(), *235*
mgcv::multinom(), *14*, *24*, *81*, *191*, *246*, *255*, *320*, *340*
mgcv::mvn(), *191*
mgcv::ocat(), *14*, *24*, *81*, *245*, *320*, *340*
mgcv::predict.bam(), *197*, *203*
mgcv::predict.gam(), *9*, *17*, *49*, *69*, *160*, *164*, *167*, *169*, *172*, *197*, *203*, *233*, *236*, *240*, *309*
mgcv::PredictMat(), *299*
mgcv::qq.gam, *246*
mgcv::qq.gam(), *25*, *246*, *341*
mgcv::residuals.gam(), *14*
mgcv::s(), *37, 38*
mgcv::scasm(), *66, 67*
mgcv::shash(), *191*, *235*
mgcv::smooth.construct, *322*
mgcv::smoothCon(), *39*, *222*
mgcv::soap, *42*
mgcv::t2(), *37, 38*
mgcv::te(), *37, 38*
mgcv::ti(), *37, 38*
mgcv::twlss(), *191*, *235*
mgcv::vcov.gam(), *194*
mgcv::ziplss(), *191*, *235*
mh_draws, 198
mirai::daemons(), *18*
model.matrix.scam *(*lp_matrix*)*, 196
model_concurvity, 201
model_concurvity(), *125*
model_constant, 202
model_edf *(*edf*)*, 145
model_terms, 203
model_vars, 205
mvnfast::rmvn(), *48*, *54*, *57*, *76*, *89*, *169*, *218*

n_eta, 206
n_smooths, 207
n_smooths(), *283*
n_smooths.sdmTMB *(*sdmTMB-methods*)*, 281
nb_theta, 208
nlme::intervals(), *333*
null_deviance, 209

observed_fitted_plot, 209
observed_fitted_plot(), *25*
overview, 210

parametric_effects, 212
parametric_effects(), *30*, *111*, *117*, *121*, *151*, *250*, *283, 284*
parametric_effects.glmmTMB *(*qq_plot.glmmTMB*)*, 247
parametric_effects.sdmTMB *(*sdmTMB-methods*)*, 281
parametric_terms, 215
parametric_terms(), *282*
parametric_terms.glmmTMB *(*qq_plot.glmmTMB*)*, 247
parametric_terms.sdmTMB *(*sdmTMB-methods*)*, 281
partial_derivatives, 216
partial_derivatives(), *92*

partial_residuals, 220
partial_residuals(), *264*
patchwork::plot_layout(), *22*, *32*, *96*, *104*, *106*, *113*, *120*, *127*, *130*, *141*
patchwork::wrap_plots(), *21*, *36*, *97, 98*, *101*, *104*, *106*, *115*, *118*, *123*, *127*, *130*, *138*, *141*
penalty, 221
post_draws, 224
post_draws(), *26*, *160, 161*, *175*, *181*, *193, 194*
posterior_predictive_plot, 228
posterior_predictive_plot(), *23*, *25*, *87*, *235*, *238, 239*, *249*, *254*
posterior_samples *(*posterior_samples.lm*)*, 231
posterior_samples(), *10, 11*, *26*, *87*, *166, 167*, *175*, *194*, *229–231*, *241*, *256*, *283*, *290*
posterior_samples.lm, 231
pp_check, 236
predicted_samples, 239
predicted_samples(), *10, 11*, *166*, *175*, *256*
print.sampling_diagnostics *(*sampling_diagnostics*)*, 274

qq_plot, 242
qq_plot(), *25*, *80*, *87*, *260*, *280*, *321*
qq_plot.glmmTMB, 247
quantile_residuals, 250
quantile_residuals(), *24*, *80*, *87*, *156*, *245*, *320*, *322*, *340*

random_effect_samples, 255
random_effect_samples(), *166*, *236*
RColorBrewer::brewer.pal(), *280*
ref_level, 257
ref_level(), *69*
ref_sims, 257
rep_first_factor_value, 258
residual_correlation, 258
residual_correlation(), *84*
residual_correlation_plot *(*residual_correlation*)*, 258
residual_correlation_plot(), *84*
residuals_covariate_plot, 260
residuals_hist_plot, 264
residuals_hist_plot(), *25*
residuals_linpred_plot, 266
residuals_linpred_plot(), *25*, *264*, *279*, *321*
response_derivatives, 267
response_derivatives(), *19*, *73*
rootogram, 272
rootogram(), *131*
rtw, 273

sampling_diagnostics, 274
sampling_diagnostics(), *132, 133*, *167*, *178*, *200*, *227*, *236*
scalable_observed_fitted_plot *(*scalable_qq_plot*)*, 275
scalable_qq_plot, 275
scalable_qq_plot(), *87*
scalable_residuals_covariate_plot *(*residuals_covariate_plot*)*, 260
scalable_residuals_hist_plot *(*scalable_qq_plot*)*, 275
scalable_residuals_linpred_plot *(*scalable_qq_plot*)*, 275
scalable_worm_plot *(*scalable_qq_plot*)*, 275
scale_fill_partial_effect, 280
scam::plot.scam(), *35*, *116*
scam::scam(), *177*, *199*, *225*, *330*
sdmTMB-methods, *214*, 281
seq_min_max *(*evenly*)*, 153
seq_min_max_eps, 285
shape_confidence_set, 285
shape_confidence_set(), *286*, *289*, *306*
shape_intervals, 288
shape_intervals(), *34*, *115*, *304*
shift_values, 289
simulate, 290
simulate.gam(), *239*, *256*
simultaneous_bands, *34*, *57*, *77*, *89, 90*, *115*, *218*, 293, *304, 305*
smallAges, 295
smooth_coef_indices, 296
smooth_coef_indices(), *298*
smooth_coefs, 297
smooth_coefs(), *296*
smooth_coefs.glmmTMB *(*qq_plot.glmmTMB*)*, 247
smooth_coefs.sdmTMB *(*sdmTMB-methods*)*, 281
smooth_data, 298
smooth_data(), *91*

smooth_dim, 300
smooth_estimates, 301
smooth_estimates(), *17*, *35*, *57, 58*, *77*, *116, 117*, *138*, *152*, *169*, *218*, *295*, *318*
smooth_estimates.glmmTMB *(qq_plot.glmmTMB)*, 247
smooth_estimates.sdmTMB *(sdmTMB-methods)*, 281
smooth_label, 307
smooth_samples, 308
smooth_samples(), *26*, *78*, *91*, *175*, *219*, *312*, *330*
smooth_terms, 313
smooth_type, 313
smooth_type(), *305*
smooths, 318
smooths(), *88*, *120, 121*, *261*, *282, 283*, *302*, *304*
smooths.glmmTMB *(qq_plot.glmmTMB)*, 247
smooths.sdmTMB *(sdmTMB-methods)*, 281
spline_values, 319
stats::confint(), *333*
stats::family(), *154*
stats::glm(), *174*, *191*
stats::lm(), *174*
stats::logLik(), *192*, *195*
stats::predict(), *8, 9*, *14*
stats::profile(), *333*
stats::qlogis(), *320*
stats::residuals(), *12, 13*
stop_if_not_mgcv_smooth *(is_mgcv_smooth)*, 188
surrogate_residuals, 320
surrogate_residuals(), *14*, *24*, *81*, *246*, *259*, *277*, *340*

term_names, 322
term_variables, 323
theta, 324
tidy_basis, 325
to_na, 326
too_far, 327
too_far_to_na, 327
transform_fun, 328
typical_values, 329
typical_values(), *47*, *69*

user_draws, 329

variance_comp, *175*, 331
variance_comp(), *142, 143*
vars_from_label, 335

waic.bam *(loo.gam)*, 195
waic.gam *(loo.gam)*, 195
waic.scam *(loo.gam)*, 195
which_smooths, 336
withr::with_options(), *183*
worm_plot, 337
worm_plot(), *80*, *280*, *321*
worm_plot.glmmTMB *(qq_plot.glmmTMB)*, 247

zooplankton, 341