Package: gamutil (via r-universe)

October 16, 2026

Title Utilities to Facilitate Modeling Routines with Generalized Additive Models

Version 0.8.1

Description After fitting a Generalized Additive (Mixed) Model, the next step is often to obtain predicted values for certain combinations of predictors for visualization of estimated effects in the model. It involves constructing a new data frame, add predicted values, and finally makes a (contour) plot. This package is intended to facilitate these steps to visualize estimated effects in a generalized additive model. The underlying modeling methodology is described in Wood (2017, ISBN:9781498728331).

Imports ggplot2, lifecycle, metR, mgcv, RColorBrewer

License MIT + file LICENSE

URL https://github.com/msaito8623/gamutil

BugReports https://github.com/msaito8623/gamutil/issues

Encoding UTF-8

Language en-US

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Suggests testthat (>= 3.0.0), knitr, rmarkdown, RhpcBLASctl

Config/testthat/edition 3

Depends R (>= 3.5)

VignetteBuilder knitr

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev libicu-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

Repository https://msaito8623.r-universe.dev

Date/Publication 2026-05-19 16:41:45 UTC

RemoteUrl https://github.com/msaito8623/gamutil

RemoteRef HEAD

RemoteSha e2d3e6eb5468a4890d19adfc7ff21f9e54d0f327

Contents

add_fit 2
constant 4
mdl_to_ndat 5
ndat_to_contour 6
plot_contour 8
plot_curve 11
plot_partial 12
vary 14

Index **15**

`add_fit` *Add predicted values with standard errors to a data.frame*

Description

In order to obtain predictions from a regression model, you need to first create a data.frame with combinations of values of the variables in the model, feed it to some function for prediction (e.g., predict.gam), and add upper and lower boundaries for confidence intervals yourself. This function takes such a data.frame (i.e., "ndat") and add predicted values and standard errors based on the model provided through "mdl". "ndat" can be created by gamutil::to_ndat.

Usage

```
add_fit(
  ndat,
  mdl,
  terms = NULL,
  cond = list(),
  terms.size = "min",
  ci.mult = qnorm(0.975),
  verbose = FALSE,
  include.parametric = TRUE,
  joint.se = FALSE
)
```

Arguments

`ndat`	A data.frame, which should already contain combinations of values of the predictors in the model.
`mdl`	A GAM object, from which predicted values are taken.

`terms` A character string, which specifies which terms in the model you would like to use to obtain predicted values.

`cond` A list, whose names are variable names and whose values are their corresponding values.

`terms.size` A character of length 1, which is "min" (default), "medium", or "max". "min" selects only one term that exactly contains the variables specified by "terms" and "cond" (i.e. the target variables) with no other variable. Therefore, it corresponds to a partial effect by the term, as produced by mgcv::plot.gam or itsadug::pvisgam. "medium" selects the terms that contain at least one of the target variables but no other. This option is useful to see the sum of partial effects, e.g. s(x0) + s(x1)

- ti(x0,x1). "max" selects all the terms that contain at least one of the target variables even if the term has other variables than given by "terms" and "cond". For example, terms=c(x0,x1) and cond=list(fac='a') with terms.size='max' also selects ti(x0,x2,by=fac), which contains "x2" that are not specified by either "terms" or "cond", in addition to s(x0,by=fac), s(x1,by=fac), ti(x0,x1,by=fac), and so on.

`ci.mult` A numeric to multiply standard errors.

`verbose` Logical. With this argument TRUE, it is printed out which terms are selected to calculate predicted values and some explanations will be printed when an error occurs.

`include.parametric`
Logical. With TRUE (default), parametric terms (e.g., main effects like `x` or `:`-interactions like `fac:x`) are eligible for inclusion in the partial effect, just like smooth terms. With FALSE, only smooth terms are kept. This is useful for visualizing the smooth-only contribution in models that mix parametric and smooth predictors.

`joint.se` Logical. With FALSE (default), the standard error reported for the summed partial effect is the square root of the sum of per-term variances returned by `predict.gam(type='terms')`, which assumes the selected terms are independent. With TRUE, the joint standard error is computed via the lpmatrix and full `vcov(mdl)`, accounting for cross-term covariances. This is the statistically correct SE when the selected terms are correlated (e.g., when a parametric main effect and a `:`-interaction are summed). The fit is unchanged. Has no effect when `terms` is `NULL`.

Value

The data.frame provided through "ndat" with additional columns for predicted values (i.e., fit) and upper and lower confidence interval boundaries (i.e., upr and lwr).

Author(s)

Motoki Saito, <motoki.saito@uni-oldenburg.de>

Examples

```
library(mgcv)
set.seed(534)
dat <- gamSim(eg=6, verbose=FALSE)
model <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3)
               + ti(x0,x1) + ti(x0,x2) + ti(x0,x3)
               + ti(x1,x2) + ti(x1,x3) + ti(x2,x3), data=dat)
target_to_vary <- c('x0','x1')
ndat <- mdl_to_ndat(mdl=model, target=target_to_vary, len=4, method=median)
terms_to_predict <- target_to_vary
# Prediction of the summed effect by all the terms:
ndat1 <- add_fit(ndat, model, verbose=TRUE)
# Partial effect of ti(x0, x1):
ndat2 <- add_fit(ndat, model, terms=terms_to_predict, terms.size="min",
                 verbose=TRUE)
# Sum of subordinate partial effects (i.e., s(x0), s(x1), ti(x0,x1)):
ndat3 <- add_fit(ndat, model, terms=terms_to_predict, terms.size="medium",
                 verbose=TRUE)
# Sum of all the pertinent partial effects (i.e., s(x0), s(x1), ti(x0,x1),
# ti(x0,x2), ti(x0,x3), ti(x1,x2), ti(x1,x3)):
ndat4 <- add_fit(ndat, model, terms=terms_to_predict, terms.size="max",
                 verbose=TRUE)
```

`constant` *Keep a variable constant*

Description

For prediction of a regression model, you need to create a data.frame for the combinations of predictor values you would like to get the model's predictions for. Sometimes you would like to keep some of the predictors constant, e.g., their median values. It can be done with this function. This function returns a single value from the vector provided through "vec" with the function provided through "func" (median as default), if the provided vector is numeric. If the vector is character or factor, then the value in the vector with the most occurrences is taken as a representative value and returned.

Usage

```
constant(vec, func = median)
```

Arguments

`vec`	A vector of numeric, character, or factor.
`func`	A function, which should take a numeric vector and returns a single numeric value.

Value

A new format-like string, from which the terms you specified have been removed.

Author(s)

Motoki Saito, <motoki.saito@uni-oldenburg.de>

Examples

```
# The most frequent occurrence is returned for character/factor.
vec <- c(rep('A',2), rep('B',3), rep('C',1))
print(constant(vec))
# The representative value by "func" is returned for numeric.
# (default=median)
set.seed(716)
vec <- rnorm(10)
print(constant(vec))
# Another function can be used, too (e.g., max).
print(constant(vec, max))
```

---

mdl_to_ndat *Create a data.frame from a GAM object for prediction*

Description

In order to obtain predicted values from a GAM object, you need to provide a data.frame with combinations of values for the predictors in the model to mgcv::predict.gam. This function does this job only by giving it a GAM object (i.e., mdl) and telling it which variables you would like to vary (i.e, target).

Usage

```
mdl_to_ndat(mdl, target = c(), cond = list(), len = 100, method = median)
```

Arguments

mdl	A GAM object (gam or bam), from which you would like to obtain predictions.
target	A character vector, which indicates variables to vary.
cond	A list of variables (as names) and their values. "cond" overrides "target".
len	A numeric, which specifies length of each varied variable, when the variable is numeric.
method	A function that takes a vector of numerics and returns a single value (e.g., median as default). The function given through this argument is used to produce a constant value for each of the numeric variables you would like to keep constant.

Value

A data.frame with the variables specified by "target" being varied and the other variables being kept constant. The returned data.frame should be ready to be used for, e.g., predict.gam.

Author(s)

Motoki Saito, <motoki.saito@uni-oldenburg.de>

Examples

```
library(mgcv)
set.seed(534)
dat <- gamSim(verbose=FALSE)
model <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3), data=dat)
target_to_vary <- c('x0','x1')
condlist <- list('x2'=quantile(dat$x2, probs=c(0.25,0.75)))
ndat <- mdl_to_ndat(mdl=model, target=target_to_vary,
                    cond=condlist, len=4, method=median)
print(length(unique(ndat$x0))==4) # x0 is varied with length=4.
print(length(unique(ndat$x1))==4) # x1 is varied with length=4.
print(length(unique(ndat$x2))==2) # x2 is 25% or 75% quantiles.
print(length(unique(ndat$x3))==1) # x3 is fixed the median.
```

---

ndat_to_contour *Contour plot from a data.frame with combinations of predictor values.*

Description

This function takes a new data.frame which contains combinations of predictor values of interest (i.e., newdata or ndat) and returns a contour plot from the data.frame. "ndat" necessary for this function can be generated with gamutil::mdl_to_ndat and gamutil::add_fit.

Usage

```
ndat_to_contour(
  ndat,
  x,
  y,
  z,
  z.lwr = "lwr",
  z.upr = "upr",
  facet.col = NULL,
  facet.labeller = NULL,
  se = TRUE,
  break.interval = NULL,
  line.breaks = NULL,
  color.breaks = NULL,
  zlim = NULL,
```

```
  contour.labels = TRUE,
  contour.line.size = 0.5
)
```

Arguments

ndat	A data.frame with combinations of predictor values and predicted values and confidence intervals.
x	A character string with length 1 for the name of the variable used for the x-axis.
y	A character string with length 1 for the name of the variable used for the y-axis.
z	A character string with length 1 for the name of the variable used for the z-axis (color).
z.lwr	A character string with length 1 for the name of the column for lower confidence interval boundary.
z.upr	A character string with length 1 for the name of the column for upper confidence interval boundary.
facet.col	A character of length 1. Split plots will be drawn for the column with the name specified by this argument.
facet.labeller	A named vector, whose names are old labels and whose labels are corresponding new labels. This named vector is used to rename names of facets.
se	Logical to indicate whether contour lines should be drawn for 1 x standard error.
break.interval	A numeric, indicating the interval from one to another contour line. This argument is not used if "line.breaks" and "color.breaks" are provided.
line.breaks	A numeric vector for contour lines.
color.breaks	A numeric vector for colors for the z-axis.
zlim	A numeric vector with length 2, which indicates the range of the z-axis.
contour.labels	Logical. With TRUE (default), contour labels will be drawn.
contour.line.size	A numeric with its length 1. It controls thickness of contour lines. The default is 0.5.

Value

A ggplot object of a contour plot with predicted values being colors (z-axis).

Author(s)

Motoki Saito, <motoki.saito@uni-oldenburg.de>

Examples

```
library(mgcv)
set.seed(534)
dat <- gamSim(eg=6,verbose=FALSE)
model <- gam(y ~ s(x0,by=fac) + s(x1,by=fac) + s(x2) + ti(x0,x1,by=fac),
             data=dat)
```

```
view <- c('x0','x1')
cnd  <- list(fac=c('1','4'))
ndat <- mdl_to_ndat(mdl=model, target=view, cond=cnd, len=10, method=median)
ndat <- add_fit(ndat, model, terms=view, cond=cnd, terms.size="min",
                ci.mult=1)
plt <- ndat_to_contour(ndat, x='x0', y='x1', z='fit', z.lwr='lwr',
                       z.upr='upr', facet.col='fac', se=TRUE)
print(plt)
```

---

plot_contour *Contour plot from a GAM model object.*

Description

This function takes a gam/bam object and names of predictors of interest and draw a contour plot. Its usage is similar to itsadug::fvisgam and itsadug::pvisgam. The current version of this function produces "partially summed" effects of pertinent terms specified by the "view" argument (see examples).

Usage

```
plot_contour(
  mdl,
  view,
  cond = list(),
  summed = TRUE,
  axis.len = 50,
  terms.size = "min",
  se = TRUE,
  break.interval = NULL,
  contour.line.breaks = NULL,
  contour.color.breaks = NULL,
  zlim = NULL,
  facet.labeller = NULL,
  verbose = FALSE,
  contour.labels = TRUE,
  contour.line.size = 0.5,
  include.parametric = TRUE,
  joint.se = FALSE,
  too.far = NULL
)
```

Arguments

mdl	An object of mgcv::gam or mgcv::bam.
view	A character vector with length of 2, which will be the x- and y-axes of the generated plot.

`cond` A list of variables names and their corresponding values, e.g. list(fac='A'), which are not included in "view".

`summed` Logical. With summed=TRUE (default), the sum of all the terms in a model will be drawn as a contour plot. With summed=FALSE, only the terms specified by "view" and "cond" are included to obtain predicted values. See also the argument "terms.size".

`axis.len` A single numeric value, which will be length of each of the x- and y-axes. Since this function builds a contour plot, length of x-axis times length of y-axis is the total size of the contour plot to be produced. Reducing the size of the contour plot to be produced would help computation.

`terms.size` A character of length 1, which is "min", "medium", or "max". This argument controls which terms to include to make prediction. "min" produces a partial effect of a single term that contains no more or less than the variables specified by "view" and "cond" (i.e., target variables). "medium" includes all the terms that have at least one of the target variables but exclude any term that has an extra variable. "max" includes all the terms that have at least one of the target variables, including also the terms with extra variables that are not specified by either "view" or "cond" as long as they have at least one of the target variables.

`se` Logical, which indicates whether standard error contour lines should be drawn (default=TRUE).

`break.interval` A single numeric value, indicating the interval from one to another contour line. This argument is not used if "contour.line.breaks" and "contour.color.breaks" are provided.

`contour.line.breaks`
A numeric vector for contour lines.

`contour.color.breaks`
A numeric vector for colors for the z-axis.

`zlim` A numeric vector with length of 2, which indicates the range of the z-axis.

`facet.labeller` A named vector, whose names are old labels and whose values are new values. Facet labels are renamed according to this named vector.

`verbose` Logical. With verbose=TRUE, terms selected for prediction were printed out and also some explanation will be provided when no term is matched.

`contour.labels` Logical. With TRUE (default), contour labels will be drawn.

`contour.line.size`
A numeric with its length 1. It controls thickness of contour lines. The default is 0.5.

`include.parametric`
Logical, passed through to `add_fit`. With TRUE (default), parametric terms (main effects and `:`-interactions) are eligible for the partial-effect computation. With FALSE, only smooth terms (`s`, `te`, `ti`, `t2`) are kept.

`joint.se` Logical, passed through to `add_fit`. With FALSE (default), the SE of the summed partial effect is computed by summing per-term variances; with TRUE, the joint SE is computed via the lpmatrix and full `vcov(mdl)`, accounting for cross-term covariances.

too.far — A single numeric or NULL. If non-NULL, grid cells whose nearest data point is farther than too.far (Euclidean distance after each axis is rescaled to [0, 1] via the data's min/max) are masked out by setting fit, se, lwr, upr to NA. This prevents the plot from showing extrapolated regions the model has no support for. Typical values: 0.05 (strict), 0.10 (moderate), 0.20 (permissive). NULL (default) disables masking. Mirrors the too.far argument of mgcv::vis.gam and itsadug::fvisgam.

Value

A ggplot object, which is a contour line plot with predicted values as colors (z-axis).

Author(s)

Motoki Saito, <motoki.saito@uni-oldenburg.de>

Examples

```
library(mgcv)
set.seed(534)
dat <- gamSim(eg=6, verbose=FALSE)

# Without "by"
mdl1 <- gam(y ~ s(x0) + s(x1) + s(x2) + ti(x0,x1), data=dat)

# With "by"
mdl2 <- gam(y ~ s(x0,by=fac) + s(x1,by=fac) + s(x2) + ti(x0,x1,by=fac),
            data=dat)

# Summed effects (i.e., s(x0) + s(x1) + s(x2) + ti(x0,x1)
plt <- plot_contour(mdl1, view=c('x0','x1'), summed=TRUE)

# Partial effect (i.e., ti(x0,x1))
plt <- plot_contour(mdl1, view=c('x0','x1'), summed=FALSE, terms.size='min')

# The sum of pertinent partial effects (i.e., s((x0) + s(x1) + ti(x0,x1))
# with confidence interval contour lines.
plt <- plot_contour(mdl1, view=c('x0','x1'), summed=FALSE,
                    terms.size='medium')

# The sum of pertinent partial effects (i.e., s((x0) + s(x1) + ti(x0,x1))
# without confidence interval contour lines.
plt <- plot_contour(mdl1, view=c('x0','x1'), summed=FALSE,
                    terms.size='medium', se=FALSE)

# Summed effects for fac='1'
plt <- plot_contour(mdl2, view=c('x0','x1'), cond=list(fac='1'),
                    summed=TRUE)

# Partial effect (i.e., ti(x0,x1)) for fac='1'
plt <- plot_contour(mdl2, view=c('x0','x1'), cond=list(fac='1'),
                    summed=FALSE, terms.size='min')
```

```
# Summed effects for fac='1' and '2' in separate panels
plt <- plot_contour(mdl2, view=c('x0','x1'), cond=list(fac=c('1','2')),
                    summed=TRUE)

# Partial effects (i.e., ti(x0,x1)) for fac='1' and '2' in separate panels
plt <- plot_contour(mdl2, view=c('x0','x1'), cond=list(fac=c('1','2')),
                    summed=FALSE, terms.size='min')
```

`plot_curve` *Curve plot of a partial effect from a GAM model.*

Description

Smooth/line counterpart to `plot_contour`. Builds a ggplot showing a fitted curve (with optional standard-error ribbon) for the partial effect of one continuous predictor, optionally split by a factor. Mirrors the argument conventions of `plot_contour` (`summed`, `cond`, `terms.size`, `joint.se`, ...) for symmetry.

Usage

```
plot_curve(
  mdl,
  view,
  cond = list(),
  summed = TRUE,
  axis.len = 100,
  terms.size = "min",
  se = TRUE,
  verbose = FALSE,
  include.parametric = TRUE,
  joint.se = FALSE
)
```

Arguments

`mdl`	An object of `mgcv::gam` or `mgcv::bam`.
`view`	A character vector of length 1 or 2 naming the predictor(s) to vary. At least one must be numeric.
`cond`	A list of variable names and their values to hold constant for prediction. See `plot_contour`.
`summed`	Logical. With `TRUE` (default), the full summed effect (including the intercept) is drawn. With `FALSE`, only the terms selected by `terms.size` on the variables in `view` are included. See `plot_contour`.
`axis.len`	Number of points along the continuous axis (default 100).

terms.size	Passed through to add_fit.
se	Logical. With TRUE (default), draw a +/- SE ribbon.
verbose	Logical, passed through to add_fit.
include.parametric	Logical, passed through to add_fit.
joint.se	Logical, passed through to add_fit.

Details

Two view configurations are supported in this version:

- one continuous predictor view = c("x") – a single line with an SE ribbon;
- one continuous + one categorical view = c("x", "f") (in either order) – one line per level of f, coloured by level.

When view contains a factor, the factor is automatically added to cond (with all levels) if it is not already present, so that add_fit's by-variable expansion has the levels it needs.

Value

A ggplot object.

Author(s)

Motoki Saito, <motoki.saito@uni-oldenburg.de>

Examples

```
library(mgcv)
set.seed(1)
n <- 400
d <- data.frame(x = rnorm(n),
                f = factor(sample(c("A","B","C"), n, TRUE)))
d$y <- d$x + ifelse(d$f == "A", 0.5*d$x, 0) + rnorm(n, 0, 0.5)
m <- bam(y ~ f + s(x, by = f, k = 3), data = d, method = "ML")
plot_curve(m, view = c("x", "f"), summed = FALSE,
           terms.size = "medium", joint.se = TRUE)
```

---

plot_partial *Plot a partial effect, dispatching by view-variable types.*

Description

Convenience wrapper that picks between plot_contour and plot_curve based on the types of the variables named in view:

- two numeric -> plot_contour (2D contour plot);
- one numeric, or one numeric + one factor -> plot_curve (line plot, optionally split by factor).

Additional arguments are forwarded to whichever underlying function is selected. Arguments that the dispatched target does not accept (e.g. contour-specific arguments like too.far or zlim when dispatching to plot_curve) are silently dropped, so the same call site can be reused across model shapes.

Usage

```
plot_partial(mdl, view, ...)
```

Arguments

mdl	An object of mgcv::gam or mgcv::bam.
view	A character vector of length 1 or 2.
...	Further arguments forwarded to plot_contour or plot_curve, depending on the dispatch.

Value

A ggplot object.

Author(s)

Motoki Saito, <motoki.saito@uni-oldenburg.de>

Examples

```
library(mgcv)
set.seed(1)

# Two numeric view variables -> contour plot.
dat2 <- gamSim(eg=2, verbose=FALSE)$data
mdl_2num <- gam(y ~ s(x, z), data=dat2)
plot_partial(mdl_2num, view = c("x", "z"), summed = FALSE,
             terms.size = "medium", joint.se = TRUE, too.far = 0.10)

# One numeric + one factor -> curve plot, one line per factor level.
n <- 400
d  <- data.frame(x = rnorm(n),
                 f = factor(sample(c("A","B","C"), n, TRUE)))
d$y <- d$x + ifelse(d$f == "A", 0.5*d$x, 0) + rnorm(n, 0, 0.5)
mdl_num_fac <- bam(y ~ f + s(x, by = f, k = 3), data = d, method = "ML")
plot_partial(mdl_num_fac, view = c("x", "f"), summed = FALSE,
             terms.size = "medium", joint.se = TRUE)
```

vary *Varying values for prediction*

Description

For prediction with a regression model, you need to create a new data.frame to specify which combinations of predictors you would like to use for prediction. This function constitutes a part of the process and produces a sequence of values based on the vector provided through the argument "vec". If the vector is numeric, a vector of numeric values in the range between min(vec) and max(vec) with the length of "len" is returned. If the vector is character or factor, the sorted unique values of the elements in the vector are returned.

Usage

```
vary(vec, len = 100)
```

Arguments

vec	A vector of numeric, character, or factor, based on which values are produced for prediction.
len	A single numeric value, which indicates how many unique values are required for each of the varied variables.

Value

A vector of numeric or character (depending on the class of "vec" provided).

Author(s)

Motoki Saito, <motoki.saito@uni-oldenburg.de>

Examples

```
# The sorted unique values are returned for character/factor.
vec <- c(rep('A',2), rep('B',3), rep('C',1))
print(constant(vec)) # 'A', 'B', 'C'
# A equally-spaced numeric sequence for the length of "len" is
# returned for numeric, with the min and max being the min and
# max of the input vector (i.e., "vec")
set.seed(716)
vec <- rnorm(10)
print(vary(vec))
print(vary(vec, 4))
```

Index

∗ **utilities**
add_fit, 2
constant, 4
mdl_to_ndat, 5
ndat_to_contour, 6
plot_contour, 8
plot_curve, 11
plot_partial, 12
vary, 14

add_fit, 2

constant, 4

mdl_to_ndat, 5

ndat_to_contour, 6

plot_contour, 8, *11*, *13*
plot_curve, 11, *13*
plot_partial, 12

vary, 14