Package ‘gammi’

January 30, 2025

Type Package

Title Generalized Additive Mixed Model Interface

Version 0.2

Date 2025-01-30

Description An interface for fitting generalized additive models (GAMs) and generalized additive mixed models (GAMMs) using the 'lme4' package as the computational engine, as described in Helwig (2024) <doi:10.3390/stats7010003>. Supports default and formula methods for model specification, additive and tensor product splines for capturing nonlinear effects, and automatic determination of spline type based on the class of each predictor. Includes an S3 plot method for visualizing the (nonlinear) model terms, an S3 predict method for forming predictions from a fit model, and an S3 summary method for conducting significance testing using the Bayesian interpretation of a smoothing spline.

License GPL (>= 2)

Encoding UTF-8

Depends R (>= 3.5.0)

Imports lme4, Matrix, methods

NeedsCompilation no

Author Nathaniel E. Helwig [aut, cre]

Maintainer Nathaniel E. Helwig <helwig@umn.edu>

Repository CRAN

Date/Publication 2025-01-30 14:50:01 UTC

Contents

exam — *Cross-Classified Examination Data*

Description

Scores on secondary school leaving examinations (response) and verbal reasoning scores in primary school (fixed effect) for 3435 students in Fife, Scotland. The students are cross-classified in 148 primary schools (random effect) and 19 secondary schools (random effect).

Usage

```
data("exam")
```

Format

A data frame with 3435 observations on the following 4 variables.

`VRQ.score` Verbal Reasoning Quotient obtained in primary school (integer vector ranging from 70 to 140)

`Exam.score` Leaving examination score obtained in secondary school (integer vector ranging from 1 to 10)

`Primary.school` Primary school identifier (factor with 148 levels)

`Secondary.school` Secondary school identifier (factor with 19 levels)

Details

The VRQ scores were obtained at age 12 (right before entering secondary school), and the Exam scores were obtained at age 16 (right before leaving secondary school). The VRQ scores are constructed to have a population mean of 100 and population standard deviation of 15. The goal is to predict the leaving Exam scores from the VRQ scores while accounting for the primary and secondary school cross-classifications.

Source

Data Obtainable from: `https://www.bristol.ac.uk/cmm/team/hg/msm-3rd-ed/datasets.html`

References

Goldstein, H. (2011). Multilevel Statistical Models, 4th Edition. Chapter 12: Cross-classified data structures (pages 243-254). doi:10.1002/9780470973394

Paterson, L. (1991). Socio-economic status and educational attainment: a multidimensional and multilevel study. Evaluation and Research in Education, 5, 97-121. doi:10.1080/09500799109533303

Examples

```
# load 'gammi' package
library(gammi)

# load 'exam' help file
?exam

# load data
data(exam)

# header of data
head(exam)

# fit model
mod <- gammi(Exam.score ~ VRQ.score, data = exam,
             random = ~ (1 | Primary.school) + (1 | Secondary.school))

# plot results
plot(mod)

# summarize results
summary(mod)

# variance parameters
mod$VarCorr
```

---

gammi *Fit a Generalized Additive Mixed Model*

Description

Fits generalized additive models (GAMs) and generalized additive mixed model (GAMMs) using **lme4** as the tuning engine. Predictor groups can be manually input (default S3 method) or inferred from the model (S3 "formula" method). Smoothing parameters are treated as variance components and estimated using REML/ML (gaussian) or Laplace approximation to ML (others).

Usage

```
gammi(x, ...)

## Default S3 method:
gammi(x,
      y,
      group,
      family = gaussian,
      fixed = NULL,
      random = NULL,
      data = NULL,
```

```
      REML = TRUE,
      control = NULL,
      start = NULL,
      verbose = 0L,
      nAGQ = 10L,
      subset,
      weights,
      na.action,
      offset,
      mustart,
      etastart,
      ...)

## S3 method for class 'formula'
gammi(formula,
      data,
      family = gaussian,
      fixed = NULL,
      random = NULL,
      REML = TRUE,
      control = NULL,
      start = NULL,
      verbose = 0L,
      nAGQ = 10L,
      subset,
      weights,
      na.action,
      offset,
      mustart,
      etastart,
      ...)
```

Arguments

`x`	Model (design) matrix of dimension `nobs` by `nvars` ($n \times p$).
`y`	Response vector of length n.
`group`	Group label vector (factor, character, or integer) of length p. Predictors with the same label are assumed to have the same variance parameter.
`formula`	Model formula: a symbolic description of the model to be fitted. Uses the same syntax as `lm` and `glm`.
`family`	Assumed exponential `family` (and link function) for the response variable.
`fixed`	For default method: a character vector specifying which `group` labels should be treated as fixed effects. For formula method: a one-sided formula specifying the fixed effects model structure.
`random`	A one-sided formula specifying the random effects structure using **lme4** syntax. See Note.

data	Optional data frame containing the variables referenced in formula, fixed, and/or random.
REML	Logical indicating whether REML versus ML should be used to tune the smoothing parameters and variance components.
control	List containing the control parameters (output from lmerControl or glmerControl).
start	List (with names) of starting parameter values for model parameters.
verbose	Postive integer that controls the level of output displayed during optimization.
nAGQ	Numer of adaptive Gaussian quadrature points. Only used for non-Gaussian responses with a single variance component.
subset	Optional expression indicating the subset of rows to use for the fitting (defaults to all rows).
weights	Optional vector indicating prior observations weights for the fitting (defaults to all ones).
na.action	Function that indicates how NA data should be dealt with. Default (of na.omit) will omit any observations with missing data on any variable.
offset	Optional vector indicating each observation's offset for the fitting (defaults to all zeros).
mustart	Optional starting values for the mean (fitted values).
etastart	Optional starting values for the linear predictors.
...	Optional arguments passed to the spline.model.matrix function, e.g., spline knots or df for each term.

Details

Fits a generalized additive mixed model (GAMM) of the form

$$g(\mu) = f(\mathbf{X}, \mathbf{Z}) + \mathbf{X}^\top \boldsymbol{\beta} + \mathbf{Z}^\top \boldsymbol{\alpha}$$

where

- $\mu = E(Y|\mathbf{X}, \mathbf{Z})$ is the conditional expectation of the response Y given the predictor vectors $\mathbf{X} = (X_1, \ldots, X_p)^\top$ and $\mathbf{Z} = (Z_1, \ldots, Z_q)^\top$
- the function $g(\cdot)$ is a user-specified (invertible) link function
- the function $f(\cdot)$ is an unknown smooth function of the predictors (specified by formula)
- the vector $\mathbf{X}$ is the fixed effects component of the design (specified by fixed)
- the vector $\mathbf{Z}$ is the random effects component of the design (specified by random)
- the vector $\boldsymbol{\beta}$ contains the unknown fixed effects coefficients
- the vector $\boldsymbol{\alpha}$ contains the unknown Gaussian random effects

Note that the mean function $f(\cdot)$ can include main and/or interaction effects between any number of predictors. Furthermore, note that the fixed effects in $\mathbf{X}^\top \boldsymbol{\beta}$ and the random effects in $\mathbf{Z}^\top \boldsymbol{\alpha}$ are both optional.

Value

An object of class "gammi" with the following elements:

- `fitted.values` model predictions on the data scale
- `linear.predictors` model predictions on the link scale
- `coefficients` coefficients used to make the predictions
- `random.coefficients` coefficients corresponding to the `random` argument, i.e., the BLUPs.
- `term.labels` labels for the terms included in the `coefficients`
- `dispersion` estimated dispersion parameter = `deviance/df.residual` when `is.null(random)`
- `vcovchol` Cholesky factor of covariance matrix such that `tcrossprod(vcovchol)` gives the covariance matrix for the combined coefficient vector `c(coefficients, random.coefficients)`
- `family` exponential family distribution (same as input)
- `logLik` log-likelihood for the solution
- `aic` AIC for the solution
- `deviance` model deviance, i.e., two times the negative log-likelihood
- `null.deviance` deviance of the null model, i.e., intercept only. Will be `NA` if the `random` argument is used.
- `r.squared` proportion of null deviance explained = `1 - deviance/null.deviance`. Will be `NA` if the `random` argument is used; see Note.
- `nobs` number of observations used in fit
- `leverages` leverage scores for each observation
- `edf` effective degrees of freedom = `sum(leverages)`
- `df.random` degress of freedom corresponding to `random` formula, i.e., number of co/variance parameters
- `df.residual` residual degrees of freedom = `nobs - edf`
- `x` input `x` matrix (default method only)
- `group` character vector indicating which columns of `x` belong to which model terms
- `scale` numeric vector giving the scale parameter used to z-score each term's data
- `fixed` fixed effects terms (default method) or formula (formula method); will be `NULL` if no fixed terms are included
- `random` random effects formula
- `mer` object of class "merMod", such as output by lmer, with model fit information on a standardized scale
- `VarCorr` data frame with variance and covariance parameter estimates from `mer` transformed back to the original scale
- `call` function call
- `data` input data
- `contrasts` list of contrasts applied to `fixed` terms; will be `NULL` if no fixed terms are included
- `spline.info` list of spline parameters for terms in `x` or `formula`
- `formula` input model formula

Random Syntax

The `random` argument uses standard lmer syntax:

- `(1 | g)` for a random intercept for each level of `g`
- `(1 | g1) + (1 | g2)` for random intercepts for `g1` and `g2`
- `(1 | g1/g2) = (1 | g1) + (1 | g1:g2)` for random intercepts for `g1` and `g2` nested within `g1`
- `(x | g) = (1 + x | g)` for a correlated random intercept and slope of `x` for each level of `g`
- `(x || g) = (1 | g) + (0 + x | g)` for an uncorrelated random intercept and slope of `x` for each level of `g`

Warning

For stable computation, any terms entered through `x` (default method) or `formula` and/or `fixed` (formula method) are z-scored prior to fitting the model. Note that terms entered through `random` are not standardized.

The `"mer"` component of the output contains the model fitting results for a z-scored version of the original data (i.e., this fit is on a different scale). Consequently, the `"mer"` component should **not** be used for prediction and/or inference purposes. All prediction and inference should be conducted using the plot, predict, and summary methods mentioned in the 'See Also' section.

The `"VarCorr"` component contains the estimated variance/covariance parameters transformed back to the original scale.

Note

The model R-squared is the proportion of the null deviance that is explained by the model, i.e.,

`r.squared = 1 - deviance / null.deviance`

where `deviance` is the deviance of the model, and `null.deviance` is the deviance of the null model.

When the `random` argument is used, `null.deviance` and `r.squared` will be `NA`. This is because there is not an obvious null model when random effects are included, e.g., should the null model include or exclude the random effects? Assuming that is it possible to define a reasonable `null.deviance` in such cases, the above formula can be applied to calculate the model R-squared for models that contain random effects.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Bates, D., Maechler, M., Bolker, B., & Walker, S. (2015). Fitting linear mixed-effects models using lme4. *Journal of Statistical Software, 67*(1), 1–48. doi:10.18637/jss.v067.i01

Helwig, N. E. (2024). Precise tensor product smoothing via spectral splines. *Stats, 7*(1), 34-53, doi:10.3390/stats7010003

See Also

plot.gammi for plotting effects from gammi objects

predict.gammi for predicting from gammi objects

summary.gammi for summarizing results from gammi objects

Examples

```
##############***##############   EXAM EXAMPLE   ##############***##############

# load 'gammi' package
library(gammi)

# load 'exam' help file
?exam

# load data
data(exam)

# header of data
head(exam)

# fit model
mod <- gammi(Exam.score ~ VRQ.score, data = exam,
             random = ~ (1 | Primary.school) + (1 | Secondary.school))

# plot results
plot(mod)

# summarize results
summary(mod)




#############***#############   GAUSSIAN EXAMPLE   #############***#############

#~~~Example 1:  Single Predictor ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# generate data
n <- 1000
x <- seq(0, 1, length.out = n)
fx <- sin(2 * pi * x)
set.seed(1)
y <- fx + rnorm(n)

# fit model via formula method
mod <- gammi(y ~ x)
mod

# fit model via default method
```

```
modmat <- spline.model.matrix(y ~ 0 + x)
tlabels <- attr(modmat, "term.labels")
tassign <- attr(modmat, "assign")
g <- factor(tlabels[tassign], levels = tlabels)
mod0 <- gammi(modmat, y, g)
mod0

# summarize fit model
summary(mod)

# plot function estimate
plot(mod)



#~~~Example 2:  Additive Model ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# mean function
eta <- function(x, z, additive = TRUE){
  mx1 <- cos(2 * pi * (x - pi))
  mx2 <- 30 * (z - 0.6)^5
  mx12 <- 0
  if(!additive) mx12 <- sin(pi * (x - z))
  mx1 + mx2 + mx12
}

# generate data
set.seed(1)
n <- 1000
x <- runif(n)
z <- runif(n)
fx <- eta(x, z)
y <- fx + rnorm(n)

# fit model via formula method
mod <- gammi(y ~ x + z)
mod

# fit model via default method
modmat <- spline.model.matrix(y ~ 0 + x + z)
tlabels <- attr(modmat, "term.labels")
tassign <- attr(modmat, "assign")
g <- factor(tlabels[tassign], levels = tlabels)
mod0 <- gammi(modmat, y, g)
mod0

# summarize fit model
summary(mod)

# plot function estimate
plot(mod)
```

```
#~~~Example 3:  Interaction Model ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# mean function
eta <- function(x, z, additive = TRUE){
  mx1 <- cos(2 * pi * (x - pi))
  mx2 <- 30 * (z - 0.6)^5
  mx12 <- 0
  if(!additive) mx12 <- sin(pi * (x - z))
  mx1 + mx2 + mx12
}

# generate data
set.seed(1)
n <- 1000
x <- runif(n)
z <- runif(n)
fx <- eta(x, z, additive = FALSE)
y <- fx + rnorm(n)

# fit model via formula method
mod <- gammi(y ~ x * z)
mod

# fit model via default method
modmat <- spline.model.matrix(y ~ 0 + x * z)
tlabels <- attr(modmat, "term.labels")
tassign <- attr(modmat, "assign")
g <- factor(tlabels[tassign], levels = tlabels)
mod0 <- gammi(modmat, y, g)
mod0

# summarize fit model
summary(mod)

# plot function estimate
plot(mod)



#~~~Example 4:  Random Intercept ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# mean function
eta <- function(x, z, additive = TRUE){
  mx1 <- cos(2 * pi * (x - pi))
  mx2 <- 30 * (z - 0.6)^5
  mx12 <- 0
  if(!additive) mx12 <- sin(pi * (x - z))
  mx1 + mx2 + mx12
}

# generate mean function
set.seed(1)
```

```
n <- 1000
nsub <- 50
x <- runif(n)
z <- runif(n)
fx <- eta(x, z)

# generate random intercepts
subid <- factor(rep(paste0("sub", 1:nsub), n / nsub),
                levels = paste0("sub", 1:nsub))
u <- rnorm(nsub, sd = sqrt(1/2))

# generate responses
y <- fx + u[subid] + rnorm(n, sd = sqrt(1/2))

# fit model via formula method
mod <- gammi(y ~ x + z, random = ~ (1 | subid))
mod

# fit model via default method
modmat <- spline.model.matrix(y ~ 0 + x + z)
tlabels <- attr(modmat, "term.labels")
tassign <- attr(modmat, "assign")
g <- factor(tlabels[tassign], levels = tlabels)
mod0 <- gammi(modmat, y, g, random = ~ (1 | subid))
mod0

# summarize fit model
summary(mod)

# plot function estimate
plot(mod)



#############***#############   BINOMIAL EXAMPLE   #############***#############

#~~~Example 1:  Single Predictor ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# generate data
n <- 1000
x <- seq(0, 1, length.out = n)
fx <- sin(2 * pi * x)
set.seed(1)
y <- rbinom(n = n, size = 1, prob = 1 / (1 + exp(-fx)))

# fit model
mod <- gammi(y ~ x, family = binomial)
mod

# summarize fit model
summary(mod)
```

```
# plot function estimate
plot(mod)



#~~~Example 2:  Additive Model ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# mean function
eta <- function(x, z, additive = TRUE){
  mx1 <- cos(2 * pi * (x - pi))
  mx2 <- 30 * (z - 0.6)^5
  mx12 <- 0
  if(!additive) mx12 <- sin(pi * (x - z))
  mx1 + mx2 + mx12
}

# generate data
set.seed(1)
n <- 1000
x <- runif(n)
z <- runif(n)
fx <- 1 + eta(x, z)
y <- rbinom(n = n, size = 1, prob = 1 / (1 + exp(-fx)))

# fit model
mod <- gammi(y ~ x + z, family = binomial)
mod

# summarize fit model
summary(mod)

# plot function estimate
plot(mod)



#~~~Example 3:  Interaction Model ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# mean function
eta <- function(x, z, additive = TRUE){
  mx1 <- cos(2 * pi * (x - pi))
  mx2 <- 30 * (z - 0.6)^5
  mx12 <- 0
  if(!additive) mx12 <- sin(pi * (x - z))
  mx1 + mx2 + mx12
}

# generate data
set.seed(1)
n <- 1000
x <- runif(n)
z <- runif(n)
fx <- eta(x, z, additive = FALSE)
```

```
y <- rbinom(n = n, size = 1, prob = 1 / (1 + exp(-fx)))

# fit model
mod <- gammi(y ~ x * z, family = binomial)
mod

# summarize fit model
summary(mod)

# plot function estimate
plot(mod)



#~~~Example 4:  Random Intercept ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# mean function
eta <- function(x, z, additive = TRUE){
  mx1 <- cos(2 * pi * (x - pi))
  mx2 <- 30 * (z - 0.6)^5
  mx12 <- 0
  if(!additive) mx12 <- sin(pi * (x - z))
  mx1 + mx2 + mx12
}

# generate mean function
set.seed(1)
n <- 1000
nsub <- 50
x <- runif(n)
z <- runif(n)
fx <- 1 + eta(x, z)

# generate random intercepts
subid <- factor(rep(paste0("sub", 1:nsub), n / nsub),
                levels = paste0("sub", 1:nsub))
u <- rnorm(nsub, sd = sqrt(1/2))

# generate responses
y <- rbinom(n = n, size = 1, prob = 1 / (1 + exp(-(fx+u[subid]))))

# fit model
mod <- gammi(y ~ x + z, random = ~ (1 | subid), family = binomial)
mod

# summarize fit model
summary(mod)

# plot function estimate
plot(mod)
```

plot.gammi *Plot Method for gammi Fits*

Description

Plots main and interaction effects from a fit gammi object.

Usage

```
## S3 method for class 'gammi'
plot(x, terms = x$term.labels, conf.int = TRUE, n = 400,
     intercept = FALSE, random = TRUE, ask = dev.interactive(),
     xlab = NULL, ylab = NULL, zlab = NULL, main = NULL, ...)
```

Arguments

`x`	Object of class "gammi"
`terms`	Which model term(s) should be plotted? Default plots all terms.
`conf.int`	Should a 95% confidence interval be added to the plot(s)?
`n`	Number of points used to plot each of the (continuous) terms.
`intercept`	Should the intercept be added to the y-axis of the plot(s)?
`random`	Should Q-Q plots of the random coefficients be produced?
`ask`	Should the user be asked before each plot is produced?
`xlab`	Optional x-axis label for plot(s).
`ylab`	Optional y-axis label for plot(s).
`zlab`	Optional z-axis label for plot(s).
`main`	Optional title for plot(s).
`...`	Additional arguments passed to internal plotting functions.

Details

Default use plots each effect function along with a 95% confidence interval (if applicable). Line plots are used for continuous predictors, bar plots are used for categorical predictors, Q-Q plots are used for random effects, and image plots are used for two-way interactions. The `visualizer1` and `visualizer2` functions are used to plot main and interaction effects, respectively.

Value

A plot is produced and nothing is returned.

Note

Three-way and higher-order interactions are not currently supported.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Helwig, N. E. (2024). Precise tensor product smoothing via spectral splines. *Stats, 7*(1), 34-53, doi:10.3390/stats7010003

See Also

`gammi` for fitting generalized additive mixed models

`predict.gammi` for predicting from gammi objects

`summary.gammi` for summarizing results from gammi objects

Examples

```
# load 'gammi' package
library(gammi)

# load data
data(exam)

# header of data
head(exam)

# fit model
mod <- gammi(Exam.score ~ VRQ.score, data = exam,
             random = ~ (1 | Primary.school) + (1 | Secondary.school))

# plot terms
plot(mod)

# refit model with Secondary.school as penalized nominal effect
mod <- gammi(Exam.score ~ Secondary.school + VRQ.score, data = exam,
             random = ~ (1 | Primary.school))

# plot terms
plot(mod)
```

`predict.gammi` *Predict Method for gammi Fits*

Description

Obtain predictions from a fit generalized additive mixed model (gammi) object.

Usage

```
## S3 method for class 'gammi'
predict(object,
        newx,
        newdata,
        se.fit = FALSE,
        type = c("link", "response", "terms"),
        conf.int = FALSE,
        conf.level = 0.95,
        ...)
```

Arguments

`object`	Object of class "gammi"
`newx`	Matrix of new `x` scores for prediction (default S3 method). Must have p columns arranged in the same order as the `x` matrix used to fit the model.
`newdata`	Data frame of new data scores for prediction (S3 "formula" method). Must contain all variables in the `formula` (and `fixed` formula if applicable) used to fit the model.
`se.fit`	Logical indicating whether standard errors of predictions should be returned.
`type`	Type of prediction to return: link = linear prediction, response = fitted value, and terms = matrix where each columns contains each term's linear predictor contribution.
`conf.int`	Logical indicating whether confidence intervals for predictions should be returned.
`conf.level`	Scalar between 0 and 1 controlling the confidence level for the interval. Ignored if `conf.int = FALSE`.
`...`	Additional arugments (ignored).

Details

The default of `type = "link"` returns the model implied linear predictor corresponding to `newx` or `newdata`, i.e.,

$$g(\hat{\boldsymbol{\mu}}_{\theta(\mathrm{new})}) = \hat{f}_\theta(\mathbf{X}_{\mathrm{new}}, \mathbf{Z}_{\mathrm{new}}) + \mathbf{X}_{\mathrm{new}}^\top \hat{\boldsymbol{\beta}}_\theta$$

where $\hat{f}_\theta(\cdot)$ is the estimated smooth function (with the subscript of θ denoting the dependence on the variance parameters), and $\hat{\boldsymbol{\beta}}_\theta$ are the fixed effect estimates (if applicable). Note that $\mathbf{X}_{\mathrm{new}}$ and $\mathbf{Z}_{\mathrm{new}}$ denote the new data at which the predictions will be formed.

Using `type = "response"` returns the predictions on the fitted value scale, i.e.,

$$\hat{\boldsymbol{\mu}}_{\theta(\mathrm{new})} = g^{-1}\left(\hat{f}_\theta(\mathbf{X}_{\mathrm{new}}, \mathbf{Z}_{\mathrm{new}}) + \mathbf{X}_{\mathrm{new}}^\top \hat{\boldsymbol{\beta}}_\theta\right)$$

where $g^{-1}(\cdot)$ denotes the inverse of the chosen link function.

Using `type = "terms"` returns a matrix where each column contains the linear predictor contribution for a different model term, i.e., the k-th column contains

$$\hat{f}_{\theta k}(\mathbf{X}_{\mathrm{new}}, \mathbf{Z}_{\mathrm{new}}) + \mathbf{X}_{\mathrm{new}k}^\top \hat{\boldsymbol{\beta}}_{\theta k}$$

where $\hat{f}_{\theta k}$ is the k-th additive function, i.e., $\hat{f}_{\theta}(\mathbf{X}_{\text{new}}, \mathbf{Z}_{\text{new}}) = \sum_{k=1}^{K} \hat{f}_{\theta k}(\mathbf{X}_{\text{new}}, \mathbf{Z}_{\text{new}})$ and the second term denotes the (optional) fixed-effect contribution for the k-th term, i.e., $\mathbf{X}_{\text{new}}^{\top} \hat{\boldsymbol{\beta}}_{\theta} = \sum_{k=1}^{K} \mathbf{X}_{\text{new}k}^{\top} \hat{\boldsymbol{\beta}}_{\theta k}$

Value

If `type = "link"` or `type = "response"`, returns either a vector (of predictions corresponding to the new data) or a data frame that contains the predictions, along with their standard errors and/or confidence interval endpoints (as controlled by `se.fit` and `conf.int` arguments).

If `type = "terms"`, returns either a matrix (with columns containing predictions for each term) or a list that contains the term-wise predictions, along with their standard errors and/or confidence interval endpoints (as controlled by `se.fit` and `conf.int` arguments).

Note

Terms entered through the `random` argument of the `gammi` function are **not** included as a part of predictions.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Helwig, N. E. (2024). Precise tensor product smoothing via spectral splines. *Stats, 7*(1), 34-53, doi:10.3390/stats7010003

See Also

`gammi` for fitting generalized additive mixed models

`plot.gammi` for plotting effects from gammi objects

`summary.gammi` for summarizing results from gammi objects

Examples

```
# load 'gammi' package
library(gammi)

# mean function
eta <- function(x, z, additive = TRUE){
  mx1 <- cos(2 * pi * (x - pi))
  mx2 <- 30 * (z - 0.6)^5
  mx12 <- 0
  if(!additive) mx12 <- sin(pi * (x - z))
  mx1 + mx2 + mx12
}

# generate mean function
set.seed(1)
n <- 1000
```

```
nsub <- 50
x <- runif(n)
z <- runif(n)
fx <- eta(x, z)

# generate random intercepts
subid <- factor(rep(paste0("sub", 1:nsub), n / nsub),
                levels = paste0("sub", 1:nsub))
u <- rnorm(nsub, sd = sqrt(1/2))

# generate responses
y <- fx + u[subid] + rnorm(n, sd = sqrt(1/2))

# fit model via formula method
mod <- gammi(y ~ x + z, random = ~ (1 | subid))
mod

# get fitted values via predict
fit <- predict(mod, newdata = data.frame(x = x, z = z))
max(abs(fit - mod$fitted.values))

# get fitted values with SE and CI
fit <- predict(mod, newdata = data.frame(x = x, z = z), conf.int = TRUE)
head(fit)

# get fitted values with SE and CI for each term
fit <- predict(mod, newdata = data.frame(x = x, z = z),
               type = "terms", conf.int = TRUE)
str(fit)                                  # list with 4 components
head(sapply(fit, function(x) x[,1]))      # for x effect
head(sapply(fit, function(x) x[,2]))      # for z effect
```

`spline.basis` *Spectral Spline Basis*

Description

Generate a spectral spline basis matrix for a nominal, ordinal, or polynomial smoothing spline.

Usage

```
spline.basis(x, df = NULL, knots = NULL, m = NULL, intercept = FALSE,
             Boundary.knots = NULL, warn.outside = TRUE,
             periodic = FALSE, xlev = levels(x))
```

Arguments

`x`	the predictor vector of length `n`. Can be a factor, integer, or numeric, see Note.
`df`	the degrees of freedom, i.e., number of knots to place at quantiles of `x`. Defaults to 10 but ignored if `knots` are provided.

knots	the breakpoints (knots) defining the spline. If knots are provided, the df is defined as length(unique(c(knots, Boundary.knots))).
m	the derivative penalty order: 0 = ordinal spline, 1 = linear spline, 2 = cubic spline, 3 = quintic spline
intercept	should an intercept be included in the basis?
Boundary.knots	the boundary points for spline basis. Defaults to range(x).
warn.outside	if TRUE, a warning is provided when x values are outside of the Boundary.knots
periodic	should the spline basis functions be constrained to be periodic with respect to the Boundary.knots?
xlev	levels of x (only applicable if x is a factor)

Details

This is a reproduction of the rk function in the **grpnet** package (Helwig, 2024b).

Given a vector of function realizations f, suppose that $f = X\beta$, where X is the (unregularized) spline basis and β is the coefficient vector. Let Q denote the postive semi-definite penalty matrix, such that $\beta^\top Q\beta$ defines the roughness penalty for the spline. See Helwig (2017) for the form of X and Q for the various types of splines.

Consider the spectral parameterization of the form $f = Z\alpha$ where

$$Z = XQ^{-1/2}$$

is the regularized spline basis (that is returned by this function), and $\alpha = Q^{1/2}\beta$ are the reparameterized coefficients. Note that $X\beta = Z\alpha$ and $\beta^\top Q\beta = \alpha^\top\alpha$, so the spectral parameterization absorbs the penalty into the coefficients (see Helwig, 2021, 2024).

Syntax of this function is designed to mimic the syntax of the bs function.

Value

Returns a basis function matrix of dimension n by df (plus 1 if an intercept is included) with the following attributes:

df	degrees of freedom
knots	knots for spline basis
m	derivative penalty order
intercept	was an intercept included?
Boundary.knots	boundary points of x
periodic	is the basis periodic?
xlev	factor levels (if applicable)

Note

The (default) type of spline basis depends on the class of the input x object:

* If x is an unordered factor, then a nominal spline basis is used

* If x is an ordered factor (and m = NULL), then an ordinal spline basis is used

* If x is an integer or numeric (and m = NULL), then a cubic spline basis is used

Note that you can override the default behavior by specifying the m argument.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Helwig, N. E. (2021). Spectrally sparse nonparametric regression via elastic net regularized smoothers. *Journal of Computational and Graphical Statistics, 30*(1), 182-191. doi:10.1080/10618600.2020.1806855

Helwig, N. E. (2024a). Precise tensor product smoothing via spectral splines. *Stats, 7*(1), 34-53, doi:10.3390/stats7010003

Helwig, N. E. (2024b). grpnet: Group Elastic Net Regularized GLMs and GAMs. R package version 0.4. doi:10.32614/CRAN.package.grpnet

See Also

`spline.model.matrix` for building model matrices using tensor products of spline bases

Examples

```
######***######   LOAD GAMMI PACKAGE   ######***######
library(gammi)


######***######   NOMINAL SPLINE BASIS   ######***######

x <- as.factor(LETTERS[1:5])
basis <- spline.basis(x)
plot(1:5, basis[,1], t = "l", ylim = extendrange(basis))
for(j in 2:ncol(basis)){
  lines(1:5, basis[,j], col = j)
}


######***######   ORDINAL SPLINE BASIS   ######***######

x <- as.ordered(LETTERS[1:5])
basis <- spline.basis(x)
plot(1:5, basis[,1], t = "l", ylim = extendrange(basis))
for(j in 2:ncol(basis)){
  lines(1:5, basis[,j], col = j)
}


######***######   LINEAR SPLINE BASIS   ######***######

x <- seq(0, 1, length.out = 101)
basis <- spline.basis(x, df = 5, m = 1)
plot(x, basis[,1], t = "l", ylim = extendrange(basis))
for(j in 2:ncol(basis)){
  lines(x, basis[,j], col = j)
}
```

```
######*****######   CUBIC SPLINE BASIS   ######*****######

x <- seq(0, 1, length.out = 101)
basis <- spline.basis(x, df = 5)
basis <- scale(basis)  # for visualization only!
plot(x, basis[,1], t = "l", ylim = extendrange(basis))
for(j in 2:ncol(basis)){
  lines(x, basis[,j], col = j)
}


######*****######   QUINTIC SPLINE BASIS   ######*****######

x <- seq(0, 1, length.out = 101)
basis <- spline.basis(x, df = 5, m = 3)
basis <- scale(basis)  # for visualization only!
plot(x, basis[,1], t = "l", ylim = extendrange(basis))
for(j in 2:ncol(basis)){
  lines(x, basis[,j], col = j)
}
```

`spline.model.matrix` *Construct Design Matrices via Spectral Splines*

Description

Creates a design (or model) matrix using the `spline.basis` function to expand variables via a spectral spline basis.

Usage

```
spline.model.matrix(object, data, ...)

rowKronecker(X, Y)
```

Arguments

`object`	a `formula` or `terms` object describing the fit model
`data`	a data frame containing the variables referenced in `object`
`...`	additional arguments passed to the `spline.basis` function, e.g., `df`, `knots`, `m`, etc. Arguments must be passed as a named list, see Examples.
`X`	matrix of dimension $n \times p$
`Y`	matrix of dimension $n \times q$

Details

This is a reproduction of the `rk.model.matrix` function in the **grpnet** package (Helwig, 2024b).

Designed to be a more flexible alternative to the `model.matrix` function. The `spline.basis` function is used to construct a marginal basis for each variable that appears in the input `object`. Tensor product interactions are formed by taking a rowwise Kronecker product of marginal basis matrices. Interactions of any order are supported using standard formulaic conventions, see Note.

Value

The design matrix corresponding to the input formula and data, which has the following attributes:

`assign`	an integer vector with an entry for each column in the matrix giving the term in the formula which gave rise to the column
`term.labels`	a character vector containing the labels for each of the terms in the model
`knots`	a named list giving the knots used for each variable in the formula
`m`	a named list giving the penalty order used for each variable in the formula
`periodic`	a named list giving the periodicity used for each variable in the formula
`xlev`	a named list giving the factor levels used for each variable in the formula

Note

For formulas of the form `y ~ x + z`, the constructed model matrix has the form `cbind(spline.basis(x), spline.basis(z))`, which simply concatenates the two marginal basis matrices. For formulas of the form `y ~ x : z`, the constructed model matrix has the form `rowKronecker(spline.basis(x), spline.basis(z))`, where `rowKronecker` denotes the row-wise kronecker product. The formula `y ~ x * z` is a shorthand for `y ~ x + z + x : z`, which concatenates the two previous results. Unless it is suppressed (using `0+`), the first column of the basis will be a column of ones named `(Intercept)`.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Helwig, N. E. (2021). Spectrally sparse nonparametric regression via elastic net regularized smoothers. *Journal of Computational and Graphical Statistics, 30*(1), 182-191. doi:10.1080/10618600.2020.1806855

Helwig, N. E. (2024a). Precise tensor product smoothing via spectral splines. *Stats, 7*(1), 34-53, doi:10.3390/stats7010003

Helwig, N. E. (2024b). grpnet: Group Elastic Net Regularized GLMs and GAMs. R package version 0.4. doi:10.32614/CRAN.package.grpnet

See Also

See `spline.basis` for details on the spectral spline basis

Examples

```
# load 'gammi' package
library(gammi)

# load data
data(exam)

# header of data
head(exam)

# make basis matrix
x <- spline.model.matrix(Exam.score ~ ., data = exam)

# check dimension (= 3435 by 178)
dim(x)

# check term labels
attr(x, "term.labels")

# check which columns of x belong to which terms
attr(x, "assign")            # note: 0 = (Intercept)
```

StartupMessage *Startup Message for gammi*

Description

Prints the startup message when gammi is loaded. Not intended to be called by the user.

Details

The ‘gammi’ ascii start-up message was created using the taag software.

References

https://patorjk.com/software/taag/

---

summary.gammi *Summary Method for gammi Fits*

Description

Obtain summary statistics from a fit generalized additive mixed model (gammi) object.

Usage

```
## S3 method for class 'gammi'
summary(object, ...)
```

Arguments

`object`	Object of class "gammi"
`...`	Additional arguments (currently ignored)

Details

Produces significance testing and model diagnostic information. The significance tests use the Bayesian interpretation of a smoothing spline. The variable importance indices sum to 100 but can be negative for some terms. The variance inflation factors should ideally be 1 for all terms; values greater than 5 or 10 can indicate noteworthy multicollinearity.

Value

An object of class "summary.gammi", which is a list with components:

`call`	the model call, i.e., `object$call`
`term.labels`	the model term labels (character vector)
`family`	the exponential `family` object
`logLik`	log-likelihood for the solution
`aic`	AIC for the solution
`deviance`	the model deviance (numeric)
`deviance.resid`	the deviance residuals
`r.squared`	the model R-squared (numeric); see Note
`df`	the total degrees of freedom = `object$edf` + `object$df.random`
`significance`	the signififance testing information (matrix)
`importance`	the variable importance information (numeric)
`vif`	the variance inflation factors (numeric)

Note

The model R-squared is the proportion of the null deviance that is explained by the model, i.e.,

`r.squared = 1 - deviance / null.deviance`

where `deviance` is the deviance of the model, and `null.deviance` is the deviance of the null model.

When the `random` argument is used, `null.deviance` and `r.squared` will be `NA`. This is because there is not an obvious null model when random effects are included, e.g., should the null model include or exclude the random effects? Assuming that is it possible to define a reasonable `null.deviance` in such cases, the above formula can be applied to calculate the model R-squared for models that contain random effects.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Helwig, N. E. (2024). Precise tensor product smoothing via spectral splines. *Stats, 7*(1), 34-53, doi:10.3390/stats7010003

See Also

`gammi` for fitting generalized additive mixed models

`plot.gammi` for plotting effects from gammi objects

`predict.gammi` for predicting from gammi objects

Examples

```
# load 'gammi' package
library(gammi)

# load data
data(exam)

# header of data
head(exam)

# fit model
mod <- gammi(Exam.score ~ VRQ.score, data = exam,
             random = ~ (1 | Primary.school) + (1 | Secondary.school))

# summarize results
summary(mod)

# refit model with Secondary.school as penalized nominal effect
mod <- gammi(Exam.score ~ Secondary.school + VRQ.score, data = exam,
             random = ~ (1 | Primary.school))

# summarize results
summary(mod)
```

---

visualizers   *Internal Functions for Plot Method*

Description

Internal functions used by the `plot.gammi` function to visualize main effects and two-way interaction effects in fit gammi objects.

Usage

```
visualizer1(x, y, bars = FALSE, bw = 0.02, lty = 1, lwd = 2, col = "black",
            lwr = NULL, upr = NULL, ci.lty = 2, ci.lwd = 1.25, ci.col = "black",
            zero = TRUE, zero.lty = 3, xlim = NULL, ylim = NULL,
```

```
            xlab = NULL, ylab = NULL, main = NULL, add = FALSE, ...)

visualizer2(x, y, z, col = NULL, ncolor = 21,
            xlim = NULL, ylim = NULL, zlim = NULL, zline = 1.5,
            xlab = NULL, ylab = NULL, zlab = NULL, main = NULL,
            xticks = NULL, xlabels = NULL, yticks = NULL, ylabels = NULL, ...)
```

Arguments

x, y, z	For 1D plots: x and y are the primary inputs to the plot function. For 2D plots: these are the primary inputs to the image function.
bars	For 1D plots: logical indicating whether to create a line plot (default) or a bar plot (bars = TRUE).
bw	For 1D plots: width of the bars relative to range of x (ignored if bars = FALSE).
lty, lwd	For 1D plots: line type and width for 1D plots.
col	For 1D plots: single color for line/bar plot. For 2D plots: vector of colors for image plot.
ncolor	For 2D plots: number of colors used for image plot and color legend, see Note.
lwr, upr	For 1D plots: number vectors defining the lower and upper bounds to plot for a confidence interval. Must be the same length as x and y.
ci.lty, ci.lwd, ci.col	For 1D plots: the type, width, and color for the confidence interval lines drawn from the lwr and upr arguments.
zero, zero.lty	For 1D plots: zero is a logical indicating whether a horizontal line at y = 0 should be included, and zero.lty controls the line type
xlim, ylim, zlim	For 1D plots: xlim and ylim are the axis limits input to the plot function. For 2D plots: these are the axis limits input to the image function (note: zlim controls range for color legend).
xlab, ylab, zlab	For 1D plots: xlab and ylab are the axis labels input to the plot function. For 2D plots: these are the axis labels input to the image function (note: zlab controls label for color legend).
main	Title of the plot.
add	Should lines/bars be added to current plot?
zline	For 2D plots: margin line for the z-axis label.
xticks, yticks	For 2D plots: tick marks for x-axis and y-axis grid lines.
xlabels, ylabels	For 2D plots: labels corresponding to the input tick marks that define the grid lines.
...	Additional arguments passed to the plot and image functions.

Details

The visualizer1 function is used to plot 1D (line/bar) plots, and the visaulizer2 function is used to plot 2D (image) plots. These functions are not intended to be called by the user, but they may be useful for producing customized visualizations that are beyond the scope of the plot.gammi function.

Value

A plot is produced and nothing is returned.

Note

The vector of colors used to construct the plots is defined as `colorRampPalette(col)(ncolor)`, which interpolates a color palette of length `ncolor` from the input colors in the vector `col`.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Helwig, N. E. (2024). Precise tensor product smoothing via spectral splines. *Stats, 7*(1), 34-53, doi:10.3390/stats7010003

See Also

`plot.gammi` for plotting effects from `gammi` objects

Examples

```
# load 'gammi' package
library(gammi)

# load 'exam' help file
?exam

# load data
data(exam)

# header of data
head(exam)

# fit model
mod <- gammi(Exam.score ~ VRQ.score, data = exam,
             random = ~ (1 | Primary.school) + (1 | Secondary.school))

# plot results (using S3 method)
plot(mod, include.random = FALSE)

# plot results (using visualizer)
xnew <- seq(min(exam$VRQ.score), max(exam$VRQ.score), length.out = 400)
pred <- predict(mod, newdata = data.frame(VRQ.score = xnew),
                type = "terms", conf.int = TRUE)
visualizer1(x = xnew, y = pred$fit, lwr = pred$lwr, upr = pred$upr,
            xlab = "VRQ.score", ylab = "Exam.score", main = "VRQ.score effect")
```

Index