Package 'FlexGAM'

May 7, 2026

Type Package

Title Generalized Additive Models with Flexible Response Functions

Version 0.7.2

Date 2020-06-06

Author Elmar Spiegel [aut, cre]

Maintainer Elmar Spiegel <espiege@uni-goettingen.de>

Depends R (>= 3.4.0), mgcv (>= 1.8-23)

Imports graphics, MASS, Matrix, scam, splines, stats

Description Standard generalized additive models assume a response function, which induces an assumption on the shape of the distribution of the response. However, miss-specifying the response function results in biased estimates. Therefore in Spiegel et al. (2017) <doi:10.1007/s11222-017-9799-6> we propose to estimate the response function jointly with the covariate effects. This package provides the underlying functions to estimate these generalized additive models with flexible response functions. The estimation is based on an iterative algorithm. In the outer loop the response function is estimated, while in the inner loop the covariate effects are determined. For the response function a strictly monotone P-spline is used while the covariate effects are estimated based on a modified Fisher-Scoring algorithm. Overall the estimation relies on the 'mgcv'-package.

License GPL-2

LazyLoad yes

LazyData yes

NeedsCompilation no

Repository CRAN

Date/Publication 2020-06-07 09:10:03 UTC

Contents

`deviance.flexgam` *Deviance of FlexGAM model*

Description

Calculating the (predictive) deviance of the model.

Usage

```
## S3 method for class 'flexgam'
deviance(object, newdata=NULL, use_penalty=FALSE, ...)
```

Arguments

`object`	Object of class `flexgam`.
`newdata`	Data to estimate the (predictive) deviance.
`use_penalty`	If the deviance should be penalized according to the smoothing penalties.
`...`	Currently not used

Details

Calculates the model deviance of the object for the given data. To get the same value as for `object$deviance` you need to set `use_penalty = TRUE`. This is due to the fact that the deviance element of the object is the penalized deviance used for step halving.

Value

Estimated deviance

Note

To get numeric stability the fitted values are truncated (`min_mu_k`) to achieve numeric stability.

Author(s)

Elmar Spiegel

References

Spiegel, Elmar, Thomas Kneib and Fabian Otto-Sobotka. Generalized additive models with flexible response functions. Statistics and Computing (2017). https://doi.org/10.1007/s11222-017-9799-6

See Also

flexgam, predict.flexgam

Examples

```
set.seed(1)
n <- 1000
x1 <- runif(n)
x2 <- runif(n)
x3 <- runif(n)
eta_orig <- -1 + 2*sin(6*x1) + exp(x2) + x3
pi_orig <- pgamma(eta_orig, shape=2, rate=sqrt(2))
y <- rbinom(n,size=1,prob=pi_orig)

Data <- data.frame(y,x1,x2,x3)
formula <- y ~ s(x1,k=20,bs="ps") + s(x2,k=20,bs="ps") + x3

# Fix smoothing parameters to save computational time.
control2 <- list("fix_smooth" = TRUE, "quietly" = TRUE, "sm_par_vec" =
                    c("lambda" = 100, "s(x1)" = 2000, "s(x2)" = 9000))
set.seed(2)
model_2 <- flexgam(formula=formula, data=Data, type="FlexGAM2",
                  family=binomial(link=logit), control = control2)
(deviance          <- deviance(model_2, newdata=Data))
```

`flexgam` *Estimation of generalized additive model with flexible response function*

Description

Main function to estimate generalized additive models with flexible response function. Currently the response must be binomial, Poisson, Gaussian or Gamma distributed.

Usage

```
flexgam(formula, data, type = c("FlexGAM1","FlexGAM2","FlexGAM1n",
        "FlexGAM2n"), family, control = NULL)
```

Arguments

formula	Formula of the covariate effects. The formula must be in the design of the mgcv package.
data	Data to fit the model.
type	Should the response function be estimated completely flexible ("FlexGAM2") or with a surrounding canonical link ("FlexGAM1")? "FlexGAM1n" and "FlexGAM2n" are similar to "FlexGAM1" and "FlexGAM2", but without monotonicity constraint. Therefore we do not recommend to use "FlexGAM1n" and "FlexGAM2n".
family	Family of the data. Currently only binomial(link="logit"), poisson(link="log"), gaussian(link="identity") and Gamma(link="log") are possible.
control	Control parameters to fit the model. The default values are described in match_flexgam_control.

Details

To reduce the bias of miss-specified response functions the function estimates the response function jointly with the covariate effects. The covariate effects are build similar to the standard mgcv::gam, while the response function is either estimated as a strictly monotone P-spline or a combination of the canonical link and a transformation of the "linear"-predictor. In the outer loop the response function is estimated, while in the inner loop a modified version of the Fisher-Scoring algorithm is applied to get the covariate effects. In the algorithm step-halving is applied. Identifiability is achieved due to at least two smooth effects and scaling of the predictors.

Value

Object of class flexgam. The list includes the

- f_k: The estimated response function.
- gam_temp: The final step of the Fisher-Scoring algorithm, so the weighted linear model based on the mgcv-package.
- sm_par_vec: The estimated smoothing parameters.
- coefficients: The coefficients of the predictor as well as of the response function.
- se: The standard deviation of the coefficients.
- mean_eta_k, sd_eta_k: Some information about the scaling of the predictor.
- control: The applied control parameters.
- control_input: The control parameters in the input
- details: Information about the occurrence of modifications due to extreme values and the convergence. As well as information of the steps done in the algorithm (if saved).
- As well as other stuff for internal usage

Author(s)

Elmar Spiegel

References

Spiegel, Elmar, Thomas Kneib and Fabian Otto-Sobotka. Generalized additive models with flexible response functions. Statistics and Computing (2017). https://doi.org/10.1007/s11222-017-9799-6

See Also

predict.flexgam, plot.flexgam, deviance.flexgam, match_flexgam_control

Examples

```
set.seed(1)
n <- 1000
x1 <- runif(n)
x2 <- runif(n)
x3 <- runif(n)
eta_orig <- -1 + 2*sin(6*x1) + exp(x2) + x3
pi_orig <- pgamma(eta_orig, shape=2, rate=sqrt(2))
y <- rbinom(n,size=1,prob=pi_orig)

Data <- data.frame(y,x1,x2,x3)
formula <- y ~ s(x1,k=20,bs="ps") + s(x2,k=20,bs="ps") + x3

# Fix smoothing parameters to save computational time.
control2 <- list("fix_smooth" = TRUE, "quietly" = TRUE, "sm_par_vec" =
                    c("lambda" = 100, "s(x1)" = 2000, "s(x2)" = 9000))

set.seed(2)
model_2 <- flexgam(formula=formula, data=Data, type="FlexGAM2",
                   family=binomial(link=logit), control = control2)

print(model_2)
summary(model_2)

plot(model_2, type = "response")
plot(model_2, type = "covariate")
```

---

flexgam_outputs *Prints object of class flexgam*

Description

Prints information for objects of class flexgam.

Usage

```
## S3 method for class 'flexgam'
print(x, ...)
## S3 method for class 'flexgam'
summary(object, ...)
```

Arguments

object	Object of class flexgam.
x	Object of class flexgam.
...	Currently not used

Details

print prints a short list of information about the given object. summary prints the same list as print and adds the values of the estimated coefficients and the corresponding standard deviations.

Value

None, only printing of object details.

Author(s)

Elmar Spiegel

References

Spiegel, Elmar, Thomas Kneib and Fabian Otto-Sobotka. Generalized additive models with flexible response functions. Statistics and Computing (2017). https://doi.org/10.1007/s11222-017-9799-6

See Also

flexgam

Examples

```
set.seed(1)
n <- 1000
x1 <- runif(n)
x2 <- runif(n)
x3 <- runif(n)
eta_orig <- -1 + 2*sin(6*x1) + exp(x2) + x3
pi_orig <- pgamma(eta_orig, shape=2, rate=sqrt(2))
y <- rbinom(n,size=1,prob=pi_orig)

Data <- data.frame(y,x1,x2,x3)
formula <- y ~ s(x1,k=20,bs="ps") + s(x2,k=20,bs="ps") + x3

# Fix smoothing parameters to save computational time.
control2 <- list("fix_smooth" = TRUE, "quietly" = TRUE, "sm_par_vec" =
                    c("lambda" = 100, "s(x1)" = 2000, "s(x2)" = 9000))

set.seed(2)
model_2 <- flexgam(formula=formula, data=Data, type="FlexGAM2",
                   family=binomial(link=logit), control = control2)

print(model_2)
```

```
summary(model_2)
```

`match_control` *Function to check the control parameters*

Description

Controls if the values of the control parameters are correct. If any control parameters are missing those are set to default. For internal use only.

Usage

```
match_flexgam_control(control = NULL, formula = formula, data=NULL)
```

Arguments

`control`	List of control parameters or NULL.
`formula`	Formula of the model to calculate the initial smoothing parameters.
`data`	Data of the model to calculate the initial smoothing parameters.

Details

Controls if the values of the control parameters are correct. If control parameters are missing they are set to default:

- `"max_iter_in"` = `100` Maximal number of inner iterations.
- `"max_iter_out"` = `100` Maximal number of outer iterations.
- `"delta_in"` = `1e-06` Convergence of inner iterations.
- `"delta_out"` = `1e-06` Convergence of outer iterations.
- `"min_mu_k"` = `1e-06` Minimal value of the fitted value. Also used to generate the upper limit for binomial data. Used to truncate the fitted values for numeric stability. (Occurrence can be read in the details).
- `"min_Psi_d"` = `1e-06` Minimal value of the derivative of the outer function. Used to truncate the derivatives for numeric stability. (Occurrence can be read in the details).
- `"min_increase"` = `1e-04` Minimal increase of the outer function.
- `"delta_halving"` = `1e-06` Minimal difference at step-halving.
- `"min_iter_halving_in"` = `1` From which inner iteration should step halving be possible?
- `"min_iter_halving_out"` = `2` From which outer iteration the deviance stopping criterion should be applied? The minimum value is 2, to get the algorithm always starting.
- `"opt_function"` = `"optim"` Which optimization function should be used to optimize the smoothing parameters? (`nlminb` or `optim(Nelder-Mead)`)

- "initial_sm" = TRUE Should the smoothing parameters of the standard mgcv::gam be used as initial values for the covariates smoothing parameters and a grid search be applied to get initial values for the smoothing parameter of the outer function?
- "fix_smooth" = FALSE Should the initial smoothing parameters (sm_par_vec) be used without optimization?
- "sm_par_vec" = c("lambda"=1,"s(x1)"=...) Initial smoothing parameters. Vector must start with "lambda" for the response function. The names of the covariate effects must fit to the mgcv output of the specified formula. There is no need to specify the initial parameters, if initial_sm = TRUE and fix_smooth = FALSE.
- "sp_range" = c(1e-8, 1e15) Range of all smoothing parameters.
- "reltol_opt" = 1e-06 Relative tolerance for optimizing the smoothing parameters.
- "quietly" = FALSE Should the algorithm print steps of optimizing the smoothing parameters and iteration procedure for the final model?
- "save_step_response" = FALSE Should the steps of the algorithm be saved for convergences checks?
- "initial_model" = c("with_intercept","no_intercept") Whether the initial model should be estimated with or without intercept.

Value

List of control parameters to fit the flexgam model.

Note

The function is designed for internal usage.

Author(s)

Elmar Spiegel

References

Spiegel, Elmar, Thomas Kneib and Fabian Otto-Sobotka. Generalized additive models with flexible response functions. Statistics and Computing (2017). https://doi.org/10.1007/s11222-017-9799-6

See Also

flexgam

Examples

```
# Only for internal usage.
```

`Methods` *Extract model parameters*

Description

These are the standard functions to extract parameters of the estimated object.

Usage

```
## S3 method for class 'flexgam'
coefficients(object, ...)
## S3 method for class 'flexgam'
coef(object, ...)
## S3 method for class 'flexgam'
fitted(object, ...)
## S3 method for class 'flexgam'
fitted.values(object, ...)
## S3 method for class 'flexgam'
residuals(object, ...)
## S3 method for class 'flexgam'
resid(object, ...)
```

Arguments

`object`	Object of class `flexgam`.
`...`	Currently not used

Details

These functions extract the coefficients, fitted values or residuals of the given object.

Value

Coefficients, fitted values or residuals of the given object.

Author(s)

Elmar Spiegel

References

Spiegel, Elmar, Thomas Kneib and Fabian Otto-Sobotka. Generalized additive models with flexible response functions. Statistics and Computing (2017). https://doi.org/10.1007/s11222-017-9799-6

See Also

`flexgam`

Examples

```
set.seed(1)
n <- 1000
x1 <- runif(n)
x2 <- runif(n)
x3 <- runif(n)
eta_orig <- -1 + 2*sin(6*x1) + exp(x2) + x3
pi_orig <- pgamma(eta_orig, shape=2, rate=sqrt(2))
y <- rbinom(n,size=1,prob=pi_orig)

Data <- data.frame(y,x1,x2,x3)
formula <- y ~ s(x1,k=20,bs="ps") + s(x2,k=20,bs="ps") + x3

# Fix smoothing parameters to save computational time.
control2 <- list("fix_smooth" = TRUE, "quietly" = TRUE, "sm_par_vec" =
                    c("lambda" = 100, "s(x1)" = 2000, "s(x2)" = 9000))

set.seed(2)
model_2 <- flexgam(formula=formula, data=Data, type="FlexGAM2",
                  family=binomial(link=logit), control = control2)

coefficients(model_2)
```

`plot.flexgam` *Plots object of class flexgam.*

Description

Plots resulting response (link) function or estimated smooth effects of objects of class `flexgam`.

Usage

```
## S3 method for class 'flexgam'
plot(x, type=c("response","covariate"), ci = TRUE,  rug = TRUE, ...)
```

Arguments

`x`	Object of class flexgam.
`type`	Whether the response function or the smooth covariate effects should be plotted.
`ci`	Include confidence intervals?
`rug`	Include rug plots?
`...`	Standard plot add-ons should work. ylab and xlab check for the length of the smooth predictor.

Details

Plots either the estimated response function or the estimated smooth covariate effects. Valid confidence intervals are currently only plotted for P-splines.

Value

Plots

Note

If other smooth effects than P-splines are used in the formula the plot function internally calls `mgcv::plot.gam` to show their effect. Then P-splines are plotted twice.

Author(s)

Elmar Spiegel

References

Spiegel, Elmar, Thomas Kneib and Fabian Otto-Sobotka. Generalized additive models with flexible response functions. Statistics and Computing (2017). https://doi.org/10.1007/s11222-017-9799-6

See Also

`flexgam`

Examples

```
set.seed(1)
n <- 1000
x1 <- runif(n)
x2 <- runif(n)
x3 <- runif(n)
eta_orig <- -1 + 2*sin(6*x1) + exp(x2) + x3
pi_orig <- pgamma(eta_orig, shape=2, rate=sqrt(2))
y <- rbinom(n,size=1,prob=pi_orig)

Data <- data.frame(y,x1,x2,x3)
formula <- y ~ s(x1,k=20,bs="ps") + s(x2,k=20,bs="ps") + x3

# Fix smoothing parameters to save computational time.
control2 <- list("fix_smooth" = TRUE, "quietly" = TRUE, "sm_par_vec" =
                    c("lambda" = 100, "s(x1)" = 2000, "s(x2)" = 9000))

set.seed(2)
model_2 <- flexgam(formula=formula, data=Data, type="FlexGAM2",
                   family=binomial(link=logit), control = control2)

plot(model_2,type="response")
plot(model_2,type="covariate")
```

`predict.flexgam` *Predicts values for the object of class flexgam*

Description

Predicts values for the given object and a given dataset.

Usage

```
## S3 method for class 'flexgam'
predict(object, newdata=NULL, type=c("response","linear.predictor",
        "terms"), ...)
```

Arguments

`object`	Object of class flexgam.
`newdata`	New data to build predicted values. Same behaviour as in standard predict.
`type`	Should the fitted values (`'response'`) or the linear predictor (`'linear.predictor'`) be predicted. Alternatively the linear predictor for each covariate separately is given (`'terms'`).
`...`	Currently not used

Details

Calculates the predicted values for the given model.

Value

Numeric vector or matrix of fitted values

Note

The sum of the `'terms'` is not the `'linear.predictor'` since the `'terms'` misses the scaling.

Author(s)

Elmar Spiegel

References

Spiegel, Elmar, Thomas Kneib and Fabian Otto-Sobotka. Generalized additive models with flexible response functions. Statistics and Computing (2017). https://doi.org/10.1007/s11222-017-9799-6

See Also

`flexgam`, `deviance.flexgam`, `response.flexgam`

Examples

```
set.seed(1)
n <- 1000
x1 <- runif(n)
x2 <- runif(n)
x3 <- runif(n)
eta_orig <- -1 + 2*sin(6*x1) + exp(x2) + x3
pi_orig <- pgamma(eta_orig, shape=2, rate=sqrt(2))
y <- rbinom(n,size=1,prob=pi_orig)

Data <- data.frame(y,x1,x2,x3)
formula <- y ~ s(x1,k=20,bs="ps") + s(x2,k=20,bs="ps") + x3

# Fix smoothing parameters to save computational time.
control2 <- list("fix_smooth" = TRUE, "quietly" = TRUE, "sm_par_vec" =
                    c("lambda" = 100, "s(x1)" = 2000, "s(x2)" = 9000))

set.seed(2)
model_2 <- flexgam(formula=formula, data=Data, type="FlexGAM2",
                  family=binomial(link=logit), control = control2)


set.seed(2)
n <- 1000
x1 <- runif(n)
x2 <- runif(n)
x3 <- runif(n)
eta_orig <- -1 + 2*sin(6*x1) + exp(x2) + x3
pi_orig <- pgamma(eta_orig, shape=2, rate=sqrt(2))
y <- rbinom(n,size=1,prob=pi_orig)

newData <- data.frame(y,x1,x2,x3)

fitted_2 <- predict(model_2, newdata=newData)
```

`response.flexgam` *Predicts values for the object of class flexgam*

Description

Predicts values for the given object, if a new predictor is given.

Usage

```
## S3 method for class 'flexgam'
response(object, linear.predictor = NULL, ...)
response(object, ...)
```

Arguments

`object`	Object of class flexgam.
`linear.predictor`	New predictor, which has to be centered and scaled properly.
`...`	Currently not used

Details

Calculates predicted values for the given model. Used to get the estimated response function separately from the covariate effects. For the same predictor `predict.flexgam(...,type="response")` and `response(...)` are equal.

Value

Numeric vector of fitted values.

Author(s)

Elmar Spiegel

References

Spiegel, Elmar, Thomas Kneib and Fabian Otto-Sobotka. Generalized additive models with flexible response functions. Statistics and Computing (2017). https://doi.org/10.1007/s11222-017-9799-6

See Also

`flexgam`, `deviance.flexgam`, `predict.flexgam`

Examples

```
set.seed(1)
n <- 1000
x1 <- runif(n)
x2 <- runif(n)
x3 <- runif(n)
eta_orig <- -1 + 2*sin(6*x1) + exp(x2) + x3
pi_orig <- pgamma(eta_orig, shape=2, rate=sqrt(2))
y <- rbinom(n,size=1,prob=pi_orig)

Data <- data.frame(y,x1,x2,x3)
formula <- y ~ s(x1,k=20,bs="ps") + s(x2,k=20,bs="ps") + x3

# Fix smoothing parameters to save computational time.
control2 <- list("fix_smooth" = TRUE, "quietly" = TRUE, "sm_par_vec" =
                   c("lambda" = 100, "s(x1)" = 2000, "s(x2)" = 9000))

set.seed(2)
model_2 <- flexgam(formula=formula, data=Data, type="FlexGAM2",
                  family=binomial(link=logit), control = control2)
```

```
set.seed(2)
n <- 1000
x1 <- runif(n)
x2 <- runif(n)
x3 <- runif(n)
eta_orig <- -1 + 2*sin(6*x1) + exp(x2) + x3
pi_orig <- pgamma(eta_orig, shape=2, rate=sqrt(2))
y <- rbinom(n,size=1,prob=pi_orig)

newData <- data.frame(y,x1,x2,x3)



fitted_1 <- predict(model_2, newdata=newData, type="response")
predictor1 <- predict(model_2, newdata=newData, type="linear.predictor")
fitted_2 <- response(model_2, linear.predictor=predictor1)
all.equal(fitted_1,fitted_2)
```

Index