Package ‘plsmselect’

July 23, 2025

Title Linear and Smooth Predictor Modelling with Penalisation and Variable Selection

Version 0.2.0

Description Fit a model with potentially many linear and smooth predictors. Interaction effects can also be quantified. Variable selection is done using penalisation. For l1-type penalties we use iterative steps alternating between using linear predictors (lasso) and smooth predictors (generalised additive model).

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 7.0.1

Depends R (>= 3.5.0)

Imports dplyr (>= 0.7.8), glmnet (>= 2.0.16), mgcv (>= 1.8.26), survival (>= 2.43.3)

Suggests knitr, rmarkdown, kableExtra, purrr

VignetteBuilder knitr

NeedsCompilation no

Author Indrayudh Ghosal [aut, cre],
Matthias Kormaksson [aut]

Maintainer Indrayudh Ghosal <ig248@cornell.edu>

Repository CRAN

Date/Publication 2019-11-24 09:50:03 UTC

Contents

`cbh` *Internal Function*

Description

Undocumented function. Do not use directly

Usage

```
cbh(lp, event.time, status)
```

Arguments

`lp`	The linear predictor to be used as offset
`event.time`	The event times
`status`	Status indicating the complement of censoring

`create_dataset` *Function to create the simulated dataset*

Description

Undocumented function. Do not use directly

Usage

```
create_dataset()
```

cumbasehaz *Cumulative Baseline Hazard of a gamlasso object*

Description

This is only used when with family="cox"

Usage

```
cumbasehaz(object)
```

Arguments

`object`	fitted model object of the class `gamlasso` as produced by `gamlasso`

Value

This function returns the cumulative baseline hazard function of a `gamlasso` object if fitted using `family = "cox"`. More specifically, cumbasehaz(object) is the cumulative baseline hazard function corresponding to the linear predictor predict(object).

See Also

`gamlasso`

Examples

```
library(plsmselect)

data(simData)

## Fit Cox gamlasso model using the formula approach:
## (L1-penalty both on X terms and smooth terms (bs="ts"))
simData$X = model.matrix(~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10, data=simData)[,-1]

cfit = gamlasso(time ~ X +
                  s(z1, bs="ts", k=5) +
                  s(z2, bs="ts", k=5) +
                  s(z3, bs="ts", k=5) +
                  s(z4, bs="ts", k=5),
                data = simData,
                family = "cox",
                weights="status",
                seed=1)

## Obtain and plot predicted cumulative baseline hazard:
H0.pred <- cumbasehaz(cfit)

time.seq <- seq(0, 60, by=1)
```

```
plot(time.seq, H0.pred(time.seq), type="l", ylab="Predicted Cumulative Baseline Hazard")

## Obtain predicted survial probabilities at month 1 and 2 (days 30 & 60):

lp <- predict(cfit) # estimated linear predictor

S.pred <- cbind(exp(-H0.pred(30)*exp(lp)), exp(-H0.pred(60)*exp(lp)))

## Obtain predicted survival at month 1 and 2 directly:
S.pred2 <- predict(cfit, type="response", new.event.times=c(30,60))

## Confirm that the two arrived at the same values:
all.equal(S.pred, S.pred2)

# See ?gamlasso for an example fitting a gaussian response model
# See ?summary.gamlasso for an example fitting a binomial response model
# See ?predict.gamlasso for an example fitting a poisson response model
```

find_family *Internal Function*

Description

Undocumented function. Do not use directly

Usage

```
find_family(fam)
```

Arguments

fam	Family in character form

formula_setup *Internal Function*

Description

Undocumented function. Do not use directly

Usage

```
formula_setup(
  formula = NULL,
  response.name = NULL,
  linear.name = NULL,
  smooth.name = NULL,
  family = NULL,
  smooth.penalty = NULL,
  num.knots = NULL,
  offset.name = NULL,
  interactions = F
)
```

Arguments

`formula` A formula to be parsed

`response.name` The name of the response variable. Vector of two if `family = "binomial"`

`linear.name` The names of the variables to be used as linear predictors

`smooth.name` The names of the variables to be used as smoothers

`family` The family describing the error distribution and link function to be used in the model. A character string which can only be `"gaussian"` (default), `"binomial"`, `"poisson"` or `"cox"`. For `family = "binomial"`, `response` can be a vector of two and for `family="cox"`, `weights` must be provided (see details below).

`smooth.penalty` The penalty used on the smoothers. Can be 1 or 2

`num.knots` Number of knots for each smoothers. Can be a single integer (recycled for each smoother variable) or a vector of integers the same length as the number of smoothers.

`offset.name` The name of the offset variable. `NULL` (default) if not provided

`interactions` logical. Should interactions be included.

`gamlasso` *Fitting a gamlasso model*

Description

This function will fit a gamlasso model with the given penalties. For some special cases using `gam` or `glmnet` might be more efficient and/or flexible

Usage

```
## S3 method for class 'formula'
gamlasso(
  formula,
  data,
  family = "gaussian",
  linear.penalty = "l1",
  smooth.penalty = "l2",
  num.knots = 5,
  offset = NULL,
  weights = NULL,
  interactions = F,
  seed = .Random.seed[1],
  num.iter = 100,
  tolerance = 1e-04,
  ...
)

## Default S3 method:
gamlasso(
  response,
  linear.terms,
  smooth.terms,
  data,
  family = "gaussian",
  linear.penalty = "l1",
  smooth.penalty = "l2",
  num.knots = 5,
  offset = NULL,
  weights = NULL,
  interactions = F,
  seed = .Random.seed[1],
  num.iter = 100,
  tolerance = 1e-04,
  prompts = F,
  verbose = T,
  ...
)
```

Arguments

`formula` A formula describing the model to be fitted

`response` The name of the response variable. Could be two variables in case of a general binomial fit (see details below)

`linear.terms` The names of the variables to be used as linear predictors

`smooth.terms` The names of the variables to be used as smoothers

`data` The data with which to fit the model

`family` The family describing the error distribution and link function to be used in the model. A character string which can only be `"gaussian"` (default), `"binomial"`, `"poisson"` or `"cox"`. For `family = "binomial"`, `response` can be a vector of two and for `family="cox"`, `weights` must be provided (see details below).

`linear.penalty` The penalty used on the linear predictors. A character string which can be `"none"` (default), `"l1"` or `"l2"`. If `"l1"` is used then we use the gam and lasso loop. Otherwise only a gam model is fitted (with penalities on parametric terms if `linear.penalty = "l2"`).

`smooth.penalty` The penalty used on the smoothers. A character string which can be `"l1"` or `"l2"` (default). `"l2"` refers to the inherent second order penalty smoothers have for controlling their shape, so `"none"` is not an option. For `"l1"` basis is specified by `bs='ts'`, else `bs='tp'` is used. (see `gam` for details on basis types)

`num.knots` Number of knots for each smoothers. Can be a single integer (recycled for each smoother variable) or a vector of integers the same length as the number of smoothers.

`offset` The name of the offset variable. `NULL` (default) if not provided

`weights` The name of the weights variable. `NULL` (default) if not provided. See details below.

`interactions` logical. Should interactions be included as covariates. If `TRUE` then the smoothers are fitted with `ti` instead of `s` so that the added effects of the interactions can be quantified separately.

`seed` The random seed can be specified for reproducibility. This is used for fitting the gam and lasso models, or fixed before each loop of gamlasso.

`num.iter` Number of iterations for the gamlasso loop

`tolerance` Tolerance for covergence of the gamlasso loop

`prompts` logical. Should `gamlassoChecks` provide interactive user prompts for corrective action when needed.

`verbose` logical. Should there be "progress reports" printed to the console while fitting the model.

`...` Additional arguments

Details

`gamlasso` allows for specifying models in two ways: 1) with the the formula approach, and 2) with the term specification approach.

The formula approach is appropriate for when the user wants an L1-penalty on the linear terms of the model, in which case the user is required to specify the linear terms in a model matrix named "X" appended to the input data frame. A typical formula specification would be "`y ~ X + s(z) + ...`" where "`X`" corresponds to the model-matrix of linear terms subject to an L1-penalty, while everything to the right of "`X`" is considered part of the gam formula (i.e. all smooth terms). In light of the above formula, gamlasso iterates (until convergence) between the following two lines of pseudo code:

- `model.cv.glmnet <- cv.glmnet(y=y, x=X, offset="model.gam fitted values")`
- `model.gam <- gam(y ~ s(z) + ..., offset="model.cv.glmnet fitted values")`

The term specification approach can fit the same type of models as the formula approach (i.e. models with L1-penalty on the linear terms). However, it is more flexible in terms of penalty-structure and can be useful if the user has big data sets with lots of variables making the formula specification cumbersome. In the term specification approach the user simply specifies the names of the data columns corresponding to the `response`, `linear.terms` and `smooth.terms` and then specifies whether to put a `linear.penalty="l1"`, `"l2"` or `"none"` (on `linear.terms`) and whether to put a `smooth.penalty="l1"` or `"l2"` (on `smooth.terms`).

While fitting a binomial model for binary responses (0/1) include the response variable before "~" if using the formula approach or when using the term- specification approach the `response` argument will be a single variable name. In general if the responses are success/failure counts then the formula should start with something similar to `cbind(success,failure) ~ ...` and for using the term-specification approach the `response` argument should be a vector of length two giving the success and failure variable names.

If `family="cox"` then the `weights` argument must be provided and should correspond to a status variable (1-censor). For other models it should correspond to a custom weights variables to be used for the weighted log-likelihood, for example the total counts for fitting a binomial model. (weights for families other than "cox" currently not implemented)

Both the formula and term-specification approaches can fit interaction models as well. There are three kinds of interactions - those between two linear predictors, between two smooth predictors and between linear and smooth predictors. For the formula approach the first type of interaction must be included as additional columns in the "`X`" matrix and the other two types must be mentioned in the smooth terms part of the formula. For the term-specification approach the argument `interaction` must be `TRUE` in which case all the pairwise interactions are used as predictors and variable selection is done on all of them.

Value

If the arguments fail the basic checking by `gamlassoChecks` then returns `NULL`. Else the function calls `gamlassoFit` which returns a list of two models, `gam` and `cv.glmnet`. Either of these could be `NULL` but if both are non-null then `convergence`, a matrix of values determining the convergence of the gamlasso loop is also returned. `gamlassoFit` also returns `inherit`, a list of select arguments used to fit the `gamlasso` model and some more values needed for prediction.

Note

The default values of `num.iter` and `tolerance` are essentially arbitrary. Also for each step when we check for convergence between the new and old predictions by the gam and lasso predictions, we use the following distance metric

$$d(x,y) = \frac{1}{length(x)} \sum_{i=1}^{length(x)} (x_i - y_i)^2$$

See Also

`gam`, `glmnet`

Examples

```
library(plsmselect)
```

```
data(simData)

## Fit gaussian gamlasso model using the formula approach:
## (L1-penalty both on model matrix (X) and smooth terms (bs=”ts”))
simData$X = model.matrix(~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10, data=simData)[,-1]

gfit = gamlasso(Yg ~ X +
                   s(z1, k=5, bs=”ts”) +
                   s(z2, k=5, bs=”ts”) +
                   s(z3, k=5, bs=”ts”) +
                   s(z4, k=5, bs=”ts”),
                   data = simData,
                   seed=1)

## Equivalently with term specification approach:
gfit = gamlasso(response=”Yg”,
                  linear.terms=paste0(”x”,1:10),
                  smooth.terms=paste0(”z”,1:4),
                  data=simData,
                  linear.penalty = ”l1”,
                  smooth.penalty = ”l1”,
                  num.knots = 5,
                  seed=1)

## The two main components of gfit are
## gfit$cv.glmnet (LASSO component) and gfit$gam (GAM components):

## Extract lasso estimates of linear terms:
coef(gfit$cv.glmnet, s=”lambda.min”)

## Plot the estimates of the smooth effects:
plot(gfit$gam, pages=1)

# See ?summary.gamlasso for an example fitting a binomial response model
# See ?predict.gamlasso for an example fitting a poisson response model
# See ?cumbasehaz for an example fitting a survival response model
```

gamlassoChecks *Checking data before fitting gamlasso*

Description

This function checks if the arguments entered for fitting a gamlasso model are compatible with each other. Not recommended to call directly. Only use if cleaning data prior to fitting `gamlassoFit`

Usage

```
gamlassoChecks(
  data,
  response.name,
  linear.name,
  smooth.name,
  family,
  linear.penalty,
  smooth.penalty,
  offset.name,
  weights.name,
  num.knots,
  num.iter,
  tolerance,
  seed,
  prompts
)
```

Arguments

`data` The training data for fitting the model

`response.name` The name of the response variable. Vector of two if `family = "binomial"`

`linear.name` The names of the variables to be used as linear predictors

`smooth.name` The names of the variables to be used as smoothers

`family` The family describing the error distribution and link function to be used in the model. A character string which can only be `"gaussian"` (default), `"binomial"`, `"poisson"` or `"cox"`. For `family = "binomial"`, `response` can be a vector of two and for `family="cox"`, `weights` must be provided (see details below).

`linear.penalty` The penalty used on the linear predictors. Can be 0, 1 or 2

`smooth.penalty` The penalty used on the smoothers. Can be 1 or 2

`offset.name` The name of the offset variable. `NULL` (default) if not provided

`weights.name` The name of the weights variable. `NULL` (default) if not provided. See `Details` of `gamlasso`.

`num.knots` Number of knots for each smoothers. Can be a single integer (recycled for each smoother variable) or a vector of integers the same length as the number of smoothers.

`num.iter` Number of iterations for the gamlasso loop

`tolerance` Tolerance for covergence of the gamlasso loop

`seed` The random seed can be specified for reproducibility. This is used for fitting the gam and lasso models, or fixed before each loop of gamlasso.

`prompts` logical. Should `gamlassoChecks` provide interactive user prompts for corrective action when needed.

Value

`gamlassoChecks` produces a series of logical values: `allcheck` indicating if the arguments passed all the checks, `fit.smoothgam` indicating if there aren't any linear predictors and a model with only smoothers should be fitted, `fit.glmnet` is the counterpart for smooth predictors. It also returns the cleaned (if needed) arguments as a list named `cleandata` who's elements are:

`train.data`	The training data with unnecessary columns deleted
`linear.name`, `smooth.name`, `num.knots`	The changed variable names and number of knots
`linear.penalty`, `smooth.penalty`	The changed penalties for linear and smooth terms. Reset to their default values o

Note

The arguments `offset.name`, `num.iter`, `tolerance` and `seed` are not currently not being used in testing.

Examples

```
## Usage similar to gamlassoFit
```

`gamlassoFit` *The function fitting a gamlasso model*

Description

This function is the workhorse for fitting a gamlasso model. Not recommended to call directly. It is slightly more efficient than `gamlasso.default` since it doesn’t perform any quality checks. Only use if the data has been cleaned and no errors are expected to occur.

Usage

```
gamlassoFit(
  data,
  formula = NULL,
  response.name = NULL,
  linear.name = NULL,
  smooth.name = NULL,
  family = "gaussian",
  linear.penalty = 0,
  smooth.penalty = 2,
  offset.name = NULL,
  weights.name = NULL,
  num.knots = 5,
  num.iter = 100,
  interactions = F,
  tolerance = 1e-04,
  seed = .Random.seed[1],
  verbose = TRUE
)
```

Arguments

`data` The training data for fitting the model

`formula` A formula describing the model to be fitted

`response.name` The name of the response variable. Vector of two if `family = "binomial"`

`linear.name` The names of the variables to be used as linear predictors

`smooth.name` The names of the variables to be used as smoothers

`family` The family describing the error distribution and link function to be used in the model. A character string which can only be `"gaussian"` (default), `"binomial"`, `"poisson"` or `"cox"`. For `family = "binomial"`, `response` can be a vector of two and for `family="cox"`, `weights` must be provided (see details below).

`linear.penalty` The penalty used on the linear predictors. Can be 0, 1 or 2

`smooth.penalty` The penalty used on the smoothers. Can be 1 or 2

`offset.name` The name of the offset variable. `NULL` (default) if not provided

`weights.name` The name of the weights variable. `NULL` (default) if not provided. See `Details` of `gamlasso`.

`num.knots` Number of knots for each smoothers. Can be a single integer (recycled for each smoother variable) or a vector of integers the same length as the number of smoothers.

`num.iter` Number of iterations for the gamlasso loop

`interactions` logical. Should interactions be included.

`tolerance` Tolerance for covergence of the gamlasso loop

`seed` The random seed can be specified for reproducibility. This is used for fitting the gam and lasso models, or fixed before each loop of gamlasso.

`verbose` logical. Should there be "progress reports" printed to the console while fitting the model.

Value

See `gamlasso`

Examples

```
## Not recommended to use directly. Please see examples of gamlasso
```

lasso_gam_loop *Internal Function*

Description

Undocumented function. Do not use directly

Usage

```
lasso_gam_loop(
  data,
  response.name,
  families,
  formulae,
  num.iter,
  tolerance,
  offset.name,
  weights,
  seed
)
```

Arguments

data	The data with value for all the linear and smooth predictors
response.name	The name of the response variable. Vector of two if family = "binomial"
families	List of two families as returned by find_family
formulae	List of formulae as returned by formula_setup
num.iter	Number of iterations for the gamlasso loop
tolerance	Tolerance for covergence of the gamlasso loop
offset.name	The name of the offset variable. NULL (default) if not provided
weights	Vector with values of the weights variable if it exists. NULL otherwise.
seed	The random seed can be specified for reproducibility. This is used for fitting the gam and lasso models, or fixed before each loop of gamlasso.

meandist *Internal Function*

Description

Undocumented function. Do not use directly

Usage

```
meandist(x, y)
```

Arguments

x, y	Vectors of the same length

nzeros *Internal Function*

Description

Undocumented function. Do not use directly

Usage

```
nzeros(x, y = NULL)
```

Arguments

x, y	Vectors of the same length

predict.gamlasso *Prediction from a fitted gamlasso model*

Description

Takes a fitted gamlasso object produced by gamlasso and returns predictions given a new set of values of the linear and smooth variables.

Usage

```
## S3 method for class 'gamlasso'
predict(
  object,
  newdata = NULL,
  type = "link",
  s = "lambda.min",
  new.event.times = NULL,
  ...
)
```

Arguments

`object` fitted model object of the class `gamlasso` as produced by `gamlasso`

`newdata` A data frame with the values of the linear and smooth variables for which predictions are to be made. If not provided then predictions corresponding to the original data used to fit `object` is returned. If provided then the variable names (column names) should match with the variable names used to fit `object`: the code throws an error if not.

`type` When this has the value `"link"` (default) then the linear predictor (with offset added if needed) is returned. When `type = "response"` predictions on the response scale is returned, depending on the family used while fitting `object`.

`s` Value of the lasso penalty parameter `lambda` at which predictions are required. Default is `"lambda.min"` but alternatively `"lambda.1se"` can be used.

`new.event.times`
A vector of new event times to be used for predicting survival times when `type = "response"` for a gamlasso object fitted with `family = "cox"`

`...` Other arguments

Details

Lasso models do not have standard errors so `predict.gamlasso` does not provide them either. The standard errors for the gam part of the model can be accesed by using `mgcv::predict.gam` with suitable options. Offsets are always included in the prediction if present in the original call to `gamlasso`. Also if `type` is anything other than `"link"` or `"response"` then the function throws an error.

Value

Returns a vector of the same length as `nrow(newdata)` with the values of the linear predictor or on the response scale depending on `type`. For `type = "link"` the value is simply the elementwise sum of the predictions from the gam and lasso models in `object`. For `type = "response"` the values are on the response scale, for example exponential of the linear response is returned if `object$inherit$family = "poisson"`

See Also

`gamlasso`, `predict.gam`, `predict.glmnet`.

Examples

```
library(plsmselect)

data(simData)

## Fit poisson gamlasso model using the term specification approach:
## (L2-penalty on linear terms & L2-penalty on smooth terms)
pfit = gamlasso(response="Yp",
                linear.terms=paste0("x",1:10),
                smooth.terms=paste0("z",1:4),
                data=simData,
                linear.penalty = "l2",
                smooth.penalty = "l2",
                family="poisson",
                num.knots = 5,
                seed=1)

## fitted values (of linear predictor):
fitted.values <- predict(pfit)

## predicted values on response scale:
pred.response <- predict(pfit, type="response", newdata=simData)

## For same model as above, but with L1-penalty on linear terms
## i.e. L1-penalty on the model matrix (X) we can use formula approach:
simData$X = model.matrix(~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10, data=simData)[,-1]

pfit = gamlasso(Yp ~ X +
                   s(z1, k=5) + # L2-penalty (bs="tp") is default (see ?mgcv::s)
                   s(z2, k=5) +
                   s(z3, k=5) +
                   s(z4, k=5),
                 family="poisson",
                 data = simData,
                 seed=1)

# See ?gamlasso for an example fitting a gaussian response model
# See ?summary.gamlasso for an example fitting a binomial response model
# See ?cumbasehaz for an example fitting a survival response model
```

print.gamlasso *Print a gamlasso object*

Description

The default print method for a gamlasso object

Usage

```
## S3 method for class 'gamlasso'
print(x, ...)
```

Arguments

x	fitted model object of the class gamlasso as produced by gamlasso
...	Other arguments

Details

Outputs a list of two. `lasso` prints the lasso model (the same output as `print(object$cv.glmnet$glmnet.fit)`) if it is non-null and `gam` prints the gam model (the same output as `print(object$gam)`) if it is non-null.

See Also

`gamlasso`, `summary.gamlasso`, `print.gam`, `print.glmnet`.

Examples

```
## Please see the examples in ?gamlasso
```

`readconfirm` *Internal Function*

Description

Internal Function

Usage

```
readconfirm()
```

`simData` *Simulated dataset to be used for gamlasso*

Description

The package includes a simulated dataset that we will use for the examples.

Usage

```
data(simData)
```

Format

A 100-by-23 data frame. There are 10 variables (`x1`,...,`x10`) corresponding to the linear predictors and 4 (`z1`,...,`z4`) corresponding to the smooth predictors. There are 7 response variables corresponding to the different models fitted -

- `Yg` for the Gaussian response
- `Yb` as Bernoulli and `success` and `failure` as Binomial count responses
- `Yp` as the Poisson response
- `time` and `status` as the survival model responses

The variables starting with `X` are the same as the linear predictors but are concatenated into a matrix `X` to be used for the formula implementation of `gamlasso`

Details

The code for creating this simulated dataset is included in the vignette of this package.

Examples

```
## Please see examples in ?gamlasso
```

---

summary.gamlasso *Summary for a gamlasso fit*

Description

Default sumary method for a `gamlasso` object

Usage

```
## S3 method for class 'gamlasso'
summary(object, s = "lambda.min", ...)
```

Arguments

`object`	fitted model object of the class `gamlasso` as produced by `gamlasso`
`s`	Value of the lasso penalty parameter `lambda` at which predictions are required. Default is `"lambda.min"` but alternatively `"lambda.1se"` can be used.
`...`	Other arguments

Details

Outputs a list of two. `gam` prints a summary of the gam model (the same output as `summary(object$gam)`) if it is non-null. Objects of the class `cv.glmnet` do not have a default summary method, so the list item `lasso` produces the coefficients of the cross-vaidated lasso fit corresponding to the lowest value of the λ used (the same output as `coef(object$cv.glmnet, s = "lambda.min")` if it is non-null).

See Also

`gamlasso`, `summary.gam`, `coef.cv.glmnet`.

Examples

```
library(plsmselect)

data(simData)

## Fit binomial gamlasso model using the term specification
## approach with binomial counts response
## (L2-penalty on linear terms & L1-penalty on smooth terms)
bfit = gamlasso(c("success","failure"),
                linear.terms=paste0("x",1:10),
                smooth.terms=paste0("z",1:4),
                data=simData,
                family = "binomial",
                linear.penalty = "l2",
                smooth.penalty = "l1",
                num.knots = 5,
                seed=1)

## Since the above model has linear.penalty = "l2" it is
## a pure GAM model (i.e. no LASSO component):
bfit$cv.glmnet

## Summary of model (here essentially the same as summary(bfit$gam)
## because there is no LASSO component, i.e. linear.penalty="l2")
summary(bfit)

## We could use the formula approach below to fit the same model as above:
simData$X = model.matrix(~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10, data=simData)[,-1]
bfit = gamlasso(cbind(success,failure) ~ X + s(z1, bs="ts") +
                 s(z2, bs="ts") + s(z3, bs="ts") + s(z4, bs="ts"),
                data = simData,
                family = "binomial",
                linear.penalty = "l2",
                smooth.penalty = "l1",
                seed=1)

## For a binary responses we only need one response variable in the formula
bfit2 = gamlasso(Yb ~ X + s(z1, bs="ts") + s(z2, bs="ts") + s(z3, bs="ts") + s(z4, bs="ts"),
                  data = simData,
                  family = "binomial",
                  seed=1)

# See ?gamlasso for an example fitting a gaussian response model
# See ?predict.gamlasso for an example fitting a poisson response model
# See ?cumbasehaz for an example fitting a survival response model
```

Index