Package 'sparseGAM'

October 14, 2022

Type Package

Title Sparse Generalized Additive Models

Version 1.0

Date 2021-05-29

Author Ray Bai

Maintainer Ray Bai <raybaistat@gmail.com>

Description Fits sparse frequentist GAMs (SF-GAM) for continuous and discrete responses in the exponential dispersion family with the group lasso, group smoothly clipped absolute deviation (SCAD), and group minimax concave (MCP) penalties <doi:10.1007/s11222-013-9424-2>. Also fits sparse Bayesian generalized additive models (SB-GAM) with the spike-and-slab group lasso (SSGL) penalty of Bai et al. (2021) <doi:10.1080/01621459.2020.1765784>. B-spline basis functions are used to model the sparse additive functions. Stand-alone functions for group-regularized negative binomial regression, group-regularized gamma regression, and group-regularized regression in the exponential dispersion family with the SSGL penalty are also provided.

License GPL-3

Depends R (>= 3.6.0)

Imports stats, splines, MASS, pracma, grpreg

NeedsCompilation yes

Repository CRAN

Date/Publication 2021-05-31 10:10:02 UTC

R topics documented:

`cv.grpreg.gamma` *Cross-validation for Group-regularized Gamma Regression*

Description

This function implements K-fold cross-validation for group-regularized gamma regression with a known shape parameter ν and the log link. For a description of group-regularized gamma regression, see the description for the `grpreg.gamma` function.

Our implementation is based on the least squares approximation approach of Wang and Leng (2007), and hence, the function does not allow the total number of covariates p to be greater than $\frac{K-1}{K} \times$ sample size, where K is the number of folds.

Usage

```
cv.grpreg.gamma(y, X, groups, gamma.shape=1, penalty=c("gLASSO","gSCAD","gMCP"),
                nfolds=10, weights, taper, nlambda=100, lambda, max.iter=10000,
                tol=1e-4)
```

Arguments

`y`	$n \times 1$ vector of responses.
`X`	$n \times p$ design matrix, where the jth column of `X` corresponds to the jth overall covariate.
`groups`	p-dimensional vector of group labels. The jth entry in `groups` should contain either the group number *or* the name of the factor level that the jth covariate belongs to. `groups` must be either a vector of integers or factors.
`gamma.shape`	known shape parameter ν in $Gamma(\mu_i, \nu)$ distribution for the responses. Default is `gamma.shape=1`.
`penalty`	group regularization method to use on the groups of coefficients. The options are `"gLASSO"`, `"gSCAD"`, and `"gMCP"`. To implement cross-validation for gamma regression with the SSGL penalty, use the `cv.SSGL` function.
`nfolds`	number of folds K to use in K-fold cross-validation. Default is `nfolds=10`.
`weights`	group-specific, nonnegative weights for the penalty. Default is to use the square roots of the group sizes.
`taper`	tapering term γ in group SCAD and group MCP controlling how rapidly the penalty tapers off. Default is `taper=4` for group SCAD and `taper=3` for group MCP. Ignored if `"gLASSO"` is specified as the penalty.
`nlambda`	number of regularization parameters L. Default is `nlambda=100`.
`lambda`	grid of L regularization parameters. The user may specify either a scalar or a vector. If the user does not provide this, the program chooses the grid automatically.

max.iter	maximum number of iterations in the algorithm. Default is max.iter=10000.
tol	convergence threshold for algorithm. Default is tol=1e-4.

Value

The function returns a list containing the following components:

lambda	$L \times 1$ vector of regularization parameters lambda used to fit the model. lambda is displayed in descending order.
cve	$L \times 1$ vector of mean cross-validation error across all K folds. The kth entry in cve corresponds to the kth regularization parameter in lambda.
cvse	$L \times 1$ vector of standard errors for cross-validation error across all K folds. The kth entry in cvse corresponds to the kth regularization parameter in lambda.
lambda.min	value of lambda that minimizes mean cross-validation error cve.

References

Breheny, P. and Huang, J. (2015). "Group descent algorithms for nonconvex penalized linear and logistic regression models with grouped predictors." *Statistics and Computing*, **25**:173-187.

Wang, H. and Leng, C. (2007). "Unified LASSO estimation by least squares approximation." *Journal of the American Statistical Association*, **102**:1039-1048.

Examples

```
## Generate data
set.seed(12345)
X = matrix(runif(100*11), nrow=100)
n = dim(X)[1]
groups = c(1,1,1,2,2,2,3,3,4,5,5)
true.beta = c(-1,1,1,0,0,0,0,0,0,1.5,-1.5)

## Generate responses from gamma regression with known shape parameter 1
eta = crossprod(t(X), true.beta)
shape = 1
y = rgamma(n, rate=shape/exp(eta), shape=shape)

## 10-fold cross-validation for group-regularized gamma regression
## with the group LASSO penalty
gamma.cv = cv.grpreg.gamma(y, X, groups, penalty="gLASSO")

## Plot cross-validation curve
plot(gamma.cv$lambda, gamma.cv$cve, type="l", xlab="lambda", ylab="CVE")
## lambda which minimizes mean CVE
gamma.cv$lambda.min
```

cv.grpreg.nb *Cross-validation for Group-regularized Negative Binomial Regression*

Description

This function implements K-fold cross-validation for group-regularized negative binomial regression with a known size parameter α and the log link. For a description of group-regularized negative binomial regression, see the description for the grpreg.nb function.

Our implementation is based on the least squares approximation approach of Wang and Leng (2007), and hence, the function does not allow the total number of covariates p to be greater than $\frac{K-1}{K}\times$ sample size, where K is the number of folds.

Usage

```
cv.grpreg.nb(y, X, groups, nb.size=1, penalty=c("gLASSO","gSCAD","gMCP"),
            nfolds=10, weights, taper, nlambda=100, lambda, max.iter=10000,
            tol=1e-4)
```

Arguments

y	$n \times 1$ vector of responses.
X	$n \times p$ design matrix, where the jth column of X corresponds to the jth overall covariate.
groups	p-dimensional vector of group labels. The jth entry in groups should contain either the group number *or* the name of the factor level that the jth covariate belongs to. groups must be either a vector of integers or factors.
nb.size	known size parameter α in $NB(\alpha, \mu_i)$ distribution for the responses. Default is nb.size=1.
penalty	group regularization method to use on the groups of coefficients. The options are "gLASSO", "gSCAD", and "gMCP". To implement cross-validation for negative binomoial regression with the SSGL penalty, use the cv.SSGL function.
nfolds	number of folds K to use in K-fold cross-validation. Default is nfolds=10.
weights	group-specific, nonnegative weights for the penalty. Default is to use the square roots of the group sizes.
taper	tapering term γ in group SCAD and group MCP controlling how rapidly the penalty tapers off. Default is taper=4 for group SCAD and taper=3 for group MCP. Ignored if "gLASSO" is specified as the penalty.
nlambda	number of regularization parameters L. Default is nlambda=100.
lambda	grid of L regularization parameters. The user may specify either a scalar or a vector. If the user does not provide this, the program chooses the grid automatically.
max.iter	maximum number of iterations in the algorithm. Default is max.iter=10000.
tol	convergence threshold for algorithm. Default is tol=1e-4.

Value

The function returns a list containing the following components:

`lambda`	$L \times 1$ vector of regularization parameters `lambda` used to fit the model. `lambda` is displayed in descending order.
`cve`	$L \times 1$ vector of mean cross-validation error across all K folds. The kth entry in `cve` corresponds to the kth regularization parameter in `lambda`.
`cvse`	$L \times 1$ vector of standard errors for cross-validation error across all K folds. The kth entry in `cvse` corresponds to the kth regularization parameter in `lambda`.
`lambda.min`	value of `lambda` that minimizes mean cross-validation error `cve`.

References

Breheny, P. and Huang, J. (2015). "Group descent algorithms for nonconvex penalized linear and logistic regression models with grouped predictors." *Statistics and Computing*, **25**:173-187.

Wang, H. and Leng, C. (2007). "Unified LASSO estimation by least squares approximation." *Journal of the American Statistical Association*, **102**:1039-1048.

Examples

```
## Generate data
set.seed(1234)
X = matrix(runif(100*16), nrow=100)
n = dim(X)[1]
groups = c(1,1,1,2,2,2,2,3,4,5,5,6,7,8,8,8)
true.beta = c(-2,2,2,0,0,0,0,0,0,1.5,-1.5,0,0,-2,2,2)

## Generate count responses from negative binomial regression
eta = crossprod(t(X), true.beta)
y = rnbinom(n,size=1, mu=exp(eta))

## 10-fold cross-validation for group-regularized negative binomial
## regression with the group SCAD penalty
nb.cv = cv.grpreg.nb(y,X,groups,penalty="gMCP")

## Plot cross-validation curve
plot(nb.cv$lambda, nb.cv$cve, type="l", xlab="lambda", ylab="CVE")
## lambda which minimizes mean CVE
nb.cv$lambda.min
```

`cv.SBGAM` *Cross-Validation for Sparse Bayesian Generalized Additive Model*

Description

This function implements K-fold cross-validation for sparse Bayesian generalized additive models (GAMs) with the spike-and-slab group lasso (SSGL) penalty. The identity link function is used for Gaussian GAMs, the logit link is used for binomial GAMs, and the log link is used for Poisson, negative binomial, and gamma GAMs.

Usage

```
cv.SBGAM(y, X, df=6,
         family=c("gaussian","binomial","poisson","negativebinomial","gamma"),
         nb.size=1, gamma.shape=1, nfolds=5, nlambda0=20, lambda0, lambda1,
         a, b, max.iter=100, tol = 1e-6, print.fold=TRUE)
```

Arguments

y	$n \times 1$ vector of responses.
X	$n \times p$ design matrix, where the jth column of X corresponds to the jth overall covariate.
df	number of B-spline basis functions to use in each basis expansion. Default is df=6, but the user may specify degrees of freedom as any integer greater than or equal to 3.
family	exponential dispersion family. Allows for "gaussian", "binomial", "poisson", "negativebinomial", and "gamma". Note that for "negativebinomial", the size parameter must be specified, while for "gamma", the shape parameter must be specified.
nb.size	known size parameter α in $NB(\alpha, \mu_i)$ distribution for negative binomial responses. Default is nb.size=1. Ignored if family is not "negativebinomial".
gamma.shape	known shape parameter ν in $Gamma(\mu_i, \nu)$ distribution for gamma responses. Default is gamma.shape=1. Ignored if family is not "gamma".
nfolds	number of folds K to use in K-fold cross-validation. Default is nfolds=5.
nlambda0	number of spike hyperparameter L. Default is nlambda0=20.
lambda0	grid of L spike hyperparameters λ_0. The user may specify either a scalar or a vector. If the user does not provide this, the program chooses the grid automatically.
lambda1	slab hyperparameter λ_1 in the SSGL prior. Default is lambda1=1.
a	shape hyperparameter for the $Beta(a, b)$ prior on the mixing proportion in the SSGL prior. Default is a=1.
b	shape hyperparameter for the $Beta(a, b)$ prior on the mixing proportion in the SSGL prior. Default is b=dim(X)[2].
max.iter	maximum number of iterations in the algorithm. Default is max.iter=100.
tol	convergence threshold for algorithm. Default is tol=1e-6.
print.fold	Boolean variable for whether or not to print the current fold in the algorithm. Default is print.fold=TRUE.

Value

The function returns a list containing the following components:

lambda0	$L \times 1$ vector of spike hyperparameters lambda0 used to fit the model. lambda0 is displayed in descending order.
cve	$L \times 1$ vector of mean cross-validation error across all K folds. The kth entry in cve corresponds to the kth regularization parameter in lambda0.

cvse	$L \times 1$ vector of standard errors for cross-validation error across all K folds. The kth entry in cvse corresponds to the kth regularization parameter in lambda0.
lambda0.min	value of lambda0 that minimizes mean cross-validation error cve.

References

Bai R. (2021). "Spike-and-slab group lasso for consistent Bayesian estimation and variable selection in non-Gaussian generalized additive models." *arXiv pre-print arXiv:2007.07021*.

Bai, R., Moran, G. E., Antonelli, J. L., Chen, Y., and Boland, M.R. (2021). "Spike-and-slab group lassos for grouped regression and sparse generalized additive models." *Journal of the American Statistical Association*, in press.

Examples

```
## Generate data
set.seed(12345)
X = matrix(runif(30*3), nrow=30)
n = dim(X)[1]
y = 2.5*sin(pi*X[,1]) + rnorm(n)

## K-fold cross-validation for 4 degrees of freedom and 4 values of lambda0
## Note that if user does not specify lambda0, cv.SBGAM chooses a grid automatically.

cv.mod = cv.SBGAM(y, X, df=4, family="gaussian", lambda0=seq(from=25,to=5,by=-10))

## Plot CVE curve
plot(cv.mod$lambda0, cv.mod$cve, type="l", xlab="lambda0", ylab="CVE")
## lambda which minimizes cross-validation error
cv.mod$lambda0.min
```

---

cv.SFGAM *Sparse Frequentist Generalized Additive Models*

Description

This function implements K-fold cross-validation for sparse frequentist generalized additive models (GAMs) with the group LASSO, group SCAD, and group MCP penalties. The identity link function is used for Gaussian GAMs, the logit link is used for binomial GAMs, and the log link is used for Poisson, negative binomial, and gamma GAMs.

Usage

```
cv.SFGAM(y, X, df=6,
        family=c("gaussian","binomial", "poisson", "negativebinomial","gamma"),
         nb.size=1, gamma.shape=1, penalty=c("gLASSO","gMCP","gSCAD"), taper,
         nfolds=10, nlambda=100, lambda, max.iter=10000, tol=1e-4)
```

Arguments

y	$n \times 1$ vector of responses.
X	$n \times p$ design matrix, where the jth column of X corresponds to the jth overall covariate.
df	number of B-spline basis functions to use in each basis expansion. Default is df=6, but the user may specify degrees of freedom as any integer greater than or equal to 3.
family	exponential dispersion family. Allows for "gaussian", "binomial", "poisson", "negativebinomial", and "gamma". Note that for "negativebinomial", the size parameter must be specified, while for "gamma", the shape parameter must be specified.
nb.size	known size parameter α in $NB(\alpha, \mu_i)$ distribution for negative binomial responses. Default is nb.size=1. Ignored if family is not "negativebinomial".
gamma.shape	known shape parameter ν in $Gamma(\mu_i, \nu)$ distribution for gamma responses. Default is gamma.shape=1. Ignored if family is not "gamma".
penalty	group regularization method to use on the groups of basis coefficients. The options are "gLASSO", "gSCAD", and "gMCP". To implement sparse GAMs with the SSGL penalty, use the SBGAM function.
taper	tapering term γ in group SCAD and group MCP controlling how rapidly the penalty tapers off. Default is taper=4 for group SCAD and taper=3 for group MCP. Ignored if "gLASSO" is specified as the penalty.
nfolds	number of folds K to use in K-fold cross-validation. Default is nfolds=10.
nlambda	number of regularization parameters L. Default is nlambda=100.
lambda	grid of L regularization parameters. The user may specify either a scalar or a vector. If the user does not provide this, the program chooses the grid automatically.
max.iter	maximum number of iterations in the algorithm. Default is max.iter=10000.
tol	convergence threshold for algorithm. Default is tol=1e-4.

Value

The function returns a list containing the following components:

lambda	$L \times 1$ vector of regularization parameters lambda used to fit the model. lambda is displayed in descending order.
cve	$L \times 1$ vector of mean cross-validation error across all K folds. The kth entry in cve corresponds to the kth regularization parameter in lambda.
cvse	$L \times 1$ vector of standard errors for cross-validation error across all K folds. The kth entry in cvse corresponds to the kth regularization parameter in lambda.
lambda.min	value of lambda that minimizes mean cross-validation error cve.

References

Breheny, P. and Huang, J. (2015). "Group descent algorithms for nonconvex penalized linear and logistic regression models with grouped predictors." *Statistics and Computing*, **25**:173-187.

Wang, H. and Leng, C. (2007). "Unified LASSO estimation by least squares approximation." *Journal of the American Statistical Association*, **102**:1039-1048.

Yuan, M. and Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **68**: 49-67.

Examples

```
## Generate data
set.seed(12345)
X = matrix(runif(100*20), nrow=100)
n = dim(X)[1]
y = 5*sin(2*pi*X[,1])-5*cos(2*pi*X[,2]) + rnorm(n)

## Test data with 50 observations
X.test = matrix(runif(50*20), nrow=50)

## Fit sparse Gaussian generalized additive model to data with the MCP penalty
gam.mod = SFGAM(y, X, X.test, family="gaussian", penalty="gMCP")

## The model corresponding to the 75th tuning parameter
gam.mod$lambda[75]
gam.mod$classifications[,75] ## The covariate index is listed first

## Plot first function f_1(x_1) in 75th model
x1 = X.test[,1]
## Estimates of all 20 function evaluations on test data
f.hat = gam.mod$f.pred[[75]]
## Extract estimates of f_1
f1.hat = f.hat[,1]

## Plot X_1 against f_1(x_1)
plot(x1[order(x1)], f1.hat[order(x1)], xlab=expression(x[1]),
     ylab=expression(f[1](x[1])))
```

`cv.SSGL` *Cross-Validation for Spike-and-Slab Group Lasso Regression*

Description

This function implements K-fold cross-validation for group-regularized regression in the exponential dispersion family with the spike-and-slab group lasso (SSGL) penalty. The identity link function is used for Gaussian regression, the logit link is used for binomial regression, and the log link is used for Poisson, negative binomial, and gamma regression.

Usage

```
cv.SSGL(y, X, groups,
        family=c("gaussian","binomial","poisson","negativebinomial","gamma"),
        nb.size=1, gamma.shape=1, weights, nfolds=5, nlambda0=20,
        lambda0, lambda1, a, b, max.iter=100, tol=1e-6, print.fold=TRUE)
```

Arguments

y	$n \times 1$ vector of responses.
X	$n \times p$ design matrix, where the jth column of X corresponds to the jth overall covariate.
groups	p-dimensional vector of group labels. The jth entry in groups should contain either the group number *or* the name of the factor level that the jth covariate belongs to. groups must be either a vector of integers or factors.
family	exponential dispersion family. Allows for "gaussian", "binomial", "poisson", "negativebinomial", and "gamma". Note that for "negativebinomial", the size parameter must be specified, while for "gamma", the shape parameter must be specified.
nb.size	known size parameter α in $NB(\alpha, \mu_i)$ distribution for negative binomial responses. Default is nb.size=1. Ignored if family is not "negativebinomial".
gamma.shape	known shape parameter ν in $Gamma(\mu_i, \nu)$ distribution for gamma responses. Default is gamma.shape=1. Ignored if family is not "gamma".
weights	group-specific, nonnegative weights for the penalty. Default is to use the square roots of the group sizes.
nfolds	number of folds K to use in K-fold cross-validation. Default is nfolds=5.
nlambda0	number of spike hyperparameters L. Default is nlambda0=20.
lambda0	grid of L spike hyperparameters λ_0. The user may specify either a scalar or a vector. If the user does not provide this, the program chooses the grid automatically.
lambda1	slab hyperparameter λ_1 in the SSGL prior. Default is lambda1=1.
a	shape hyperparameter for the $Beta(a, b)$ prior on the mixing proportion in the SSGL prior. Default is a=1.
b	shape hyperparameter for the $Beta(a, b)$ prior on the mixing proportion in the SSGL prior. Default is b=dim(X)[2].
max.iter	maximum number of iterations in the algorithm. Default is max.iter=100.
tol	convergence threshold for algorithm. Default is tol=1e-6.
print.fold	Boolean variable for whether or not to print the current fold in the algorithm. Default is print.fold=TRUE.

Value

The function returns a list containing the following components:

lambda0	$L \times 1$ vector of spike hyperparameters lambda0 used to fit the model. lambda0 is displayed in descending order.

cve	$L \times 1$ vector of mean cross-validation error across all K folds. The kth entry in cve corresponds to the kth regularization parameter in lambda0.
cvse	$L \times 1$ vector of standard errors for cross-validation error across all K folds. The kth entry in cvse corresponds to the kth regularization parameter in lambda0.
lambda0.min	value of lambda0 that minimizes mean cross-validation error cve.

References

Bai R. (2021). "Spike-and-slab group lasso for consistent Bayesian estimation and variable selection in non-Gaussian generalized additive models." *arXiv pre-print arXiv:2007.07021*.

Bai, R., Moran, G. E., Antonelli, J. L., Chen, Y., and Boland, M.R. (2021). "Spike-and-slab group lassos for grouped regression and sparse generalized additive models." *Journal of the American Statistical Association*, in press.

Examples

```
## Generate data
set.seed(12345)
X = matrix(runif(30*6), nrow=30)
n = dim(X)[1]
groups = c(1,1,1,2,2,3)
true.beta = c(-1.5,0.5,-1.5,0,0,0)

## Generate responses from Gaussian distribution
y = crossprod(t(X), true.beta) + rnorm(n)

## K-fold cross-validation for 3 choices of lambda0
## Note that if user does not specify lambda0, cv.SSGL chooses a grid automatically.

ssgl.mods = cv.SSGL(y, X, groups, family="gaussian", lambda0=seq(from=10,to=2,by=-4))

## Plot cross-validation curve
plot(ssgl.mods$lambda0, ssgl.mods$cve, type="l", xlab="lambda0", ylab="CVE")
## lambda which minimizes mean CVE
ssgl.mods$lambda0.min


## Example with Poisson regression

## Generate count responses
eta = crossprod(t(X), true.beta)
y = rpois(n,exp(eta))

## K-fold cross-validation with 4 choices of lambda0
## Note that if user does not specify lambda0, cv.SSGL chooses a grid automatically.

ssgl.poisson.mods = cv.SSGL(y, X, groups, family="poisson", lambda0=seq(from=8,to=2,by=-2))

## Plot cross-validation curve
plot(ssgl.poisson.mods$lambda0, ssgl.poisson.mods$cve, type="l", xlab="lambda0", ylab="CVE")
## lambda which minimizes mean CVE
```

```
ssgl.poisson.mods$lambda0.min
```

`grpreg.gamma` *Group-regularized Gamma Regression*

Description

This function implements group-regularized gamma regression with a known shape parameter ν and the log link. In gamma regression, we assume that $y_i \sim Gamma(\mu_i, \nu)$, where

$$f(y_i|\mu_i, \nu) = \frac{1}{\Gamma(\nu)}(\frac{\nu}{\mu_i})^\nu \exp(-\frac{\nu}{\mu_i}y_i)y_i^{\nu-1}, y > 0.$$

Then $E(y_i) = \mu_i$, and we relate μ_i to a set of p covariates x_i through the log link,

$$\log(\mu_i) = \beta_0 + x_i^T\beta, i = 1, ..., n$$

If the covariates in each x_i are grouped according to known groups $g = 1, ..., G$, then this function may estimate some of the G groups of coefficients as all zero, depending on the amount of regularization.

Our implementation for regularized gamma regression is based on the least squares approximation approach of Wang and Leng (2007), and hence, the function does not allow the total number of covariates p to be greater than sample size.

Usage

```
grpreg.gamma(y, X, X.test, groups, gamma.shape=1,
             penalty=c("gLASSO","gSCAD","gMCP"),
             weights, taper, nlambda=100, lambda, max.iter=10000, tol=1e-4)
```

Arguments

`y`	$n \times 1$ vector of responses for training data.
`X`	$n \times p$ design matrix for training data, where the jth column of `X` corresponds to the jth overall covariate.
`X.test`	$n_{test} \times p$ design matrix for test data to calculate predictions. `X.test` must have the *same* number of columns as `X`, but not necessarily the same number of rows. If *no* test data is provided or if in-sample predictions are desired, then the function automatically sets `X.test=X` in order to calculate *in-sample* predictions.
`groups`	p-dimensional vector of group labels. The jth entry in `groups` should contain either the group number *or* the name of the factor level that the jth covariate belongs to. `groups` must be either a vector of integers or factors.
`gamma.shape`	known shape parameter ν in $Gamma(\mu_i, \nu)$ distribution for the responses. Default is `gamma.shape=1`.

penalty	group regularization method to use on the groups of coefficients. The options are "gLASSO", "gSCAD", "gMCP". To implement gamma regression with the SSGL penalty, use the SSGL function.
weights	group-specific, nonnegative weights for the penalty. Default is to use the square roots of the group sizes.
taper	tapering term γ in group SCAD and group MCP controlling how rapidly the penalty tapers off. Default is taper=4 for group SCAD and taper=3 for group MCP. Ignored if "gLASSO" is specified as the penalty.
nlambda	number of regularization parameters L. Default is nlambda=100.
lambda	grid of L regularization parameters. The user may specify either a scalar or a vector. If the user does not provide this, the program chooses the grid automatically.
max.iter	maximum number of iterations in the algorithm. Default is max.iter=10000.
tol	convergence threshold for algorithm. Default is tol=1e-4.

Value

The function returns a list containing the following components:

lambda	$L \times 1$ vector of regularization parameters lambda used to fit the model. lambda is displayed in descending order.
beta0	$L \times 1$ vector of estimated intercepts. The kth entry in beta0 corresponds to the kth regularization parameter in lambda.
beta	$p \times L$ matrix of estimated regression coefficients. The kth column in beta corresponds to the kth regularization parameter in lambda.
mu.pred	$n_{test} \times L$ matrix of predicted mean response values $\mu_{test} = E(Y_{test})$ based on the *test* data in X.test (or training data X if no argument was specified for X.test). The kth column in mu.pred corresponds to the predictions for the kth regularization parameter in lambda.
classifications	$G \times L$ matrix of classifications, where G is the number of groups. An entry of "1" indicates that the group was classified as nonzero, and an entry of "0" indicates that the group was classified as zero. The kth column of classifications corresponds to the kth regularization parameter in lambda.
loss	$L \times 1$ vector of negative log-likelihood of the fitted models. The kth entry in loss corresponds to the kth regularization parameter in lambda.

References

Breheny, P. and Huang, J. (2015). "Group descent algorithms for nonconvex penalized linear and logistic regression models with grouped predictors." *Statistics and Computing*, **25**:173-187.

Wang, H. and Leng, C. (2007). "Unified LASSO estimation by least squares approximation." *Journal of the American Statistical Association*, **102**:1039-1048.

Examples

```
## Generate data
set.seed(12345)
X = matrix(runif(100*11), nrow=100)
n = dim(X)[1]
groups = c("a","a","a","b","b","b","c","c","d","e","e")
groups = as.factor(groups)
true.beta = c(-1,1,1,0,0,0,0,0,0,1.5,-1.5)

## Generate responses from gamma regression with known shape parameter 1
eta = crossprod(t(X), true.beta)
shape = 1
y = rgamma(n, rate=shape/exp(eta), shape=shape)

## Generate test data
n.test = 50
X.test = matrix(runif(n.test*11), nrow=n.test)

## Fit gamma regression models with the group LASSO penalty
gamma.mod = grpreg.gamma(y, X, X.test, groups, penalty="gLASSO")

## Tuning parameters used to fit models
gamma.mod$lambda

# Predicted n.test-dimensional vectors mu=E(Y.test) based on test data, X.test.
# The kth column of 'mu.pred' corresponds to the kth entry in 'lambda.'
gamma.mod$mu.pred

# Classifications of the 5 groups. The kth column of 'classifications'
# corresponds to the kth entry in 'lambda.'
gamma.mod$classifications
```

`grpreg.nb` *Group-regularized Negative Binomial Regression*

Description

This function implements group-regularized negative binomial regression with a known size parameter α and the log link. In negative binomial regression, we assume that $y_i \sim NB(\alpha, \mu_i)$, where

$$f(y_i|\alpha,\mu_i) = \frac{\Gamma(y+\alpha)}{y!\Gamma(\alpha)}(\frac{\mu_i}{\mu_i+\alpha})^y(\frac{\alpha}{\mu_i+\alpha})^\alpha, y = 0,1,2,...$$

Then $E(y_i) = \mu_i$, and we relate μ_i to a set of p covariates x_i through the log link,

$$\log(\mu_i) = \beta_0 + x_i^T\beta, i = 1,...,n$$

If the covariates in each x_i are grouped according to known groups $g = 1, ..., G$, then this function may estimate some of the G groups of coefficients as all zero, depending on the amount of regularization.

Our implementation for regularized negative binomial regression is based on the least squares approximation approach of Wang and Leng (2007), and hence, the function does not allow the total number of covariates p to be greater than sample size.

Usage

```
grpreg.nb(y, X, X.test, groups, nb.size=1, penalty=c("gLASSO","gSCAD","gMCP"),
          weights, taper, nlambda=100, lambda, max.iter=10000, tol=1e-4)
```

Arguments

`y`	$n \times 1$ vector of responses for training data.
`X`	$n \times p$ design matrix for training data, where the jth column of `X` corresponds to the jth overall covariate.
`X.test`	$n_{test} \times p$ design matrix for test data to calculate predictions. `X.test` must have the *same* number of columns as `X`, but not necessarily the same number of rows. If *no* test data is provided or if in-sample predictions are desired, then the function automatically sets `X.test=X` in order to calculate *in-sample* predictions.
`groups`	p-dimensional vector of group labels. The jth entry in `groups` should contain either the group number *or* the name of the factor level that the jth covariate belongs to. `groups` must be either a vector of integers or factors.
`nb.size`	known size parameter α in $NB(\alpha, \mu_i)$ distribution for the responses. Default is `nb.size=1`.
`penalty`	group regularization method to use on the groups of coefficients. The options are `"gLASSO"`, `"gSCAD"`, `"gMCP"`. To implement negative binomial regression with the SSGL penalty, use the `SSGL` function.
`weights`	group-specific, nonnegative weights for the penalty. Default is to use the square roots of the group sizes.
`taper`	tapering term γ in group SCAD and group MCP controlling how rapidly the penalty tapers off. Default is `taper=4` for group SCAD and `taper=3` for group MCP. Ignored if `"gLASSO"` is specified as the penalty.
`nlambda`	number of regularization parameters L. Default is `nlambda=100`.
`lambda`	grid of L regularization parameters. The user may specify either a scalar or a vector. If the user does not provide this, the program chooses the grid automatically.
`max.iter`	maximum number of iterations in the algorithm. Default is `max.iter=10000`.
`tol`	convergence threshold for algorithm. Default is `tol=1e-4`.

Value

The function returns a list containing the following components:

`lambda` $L \times 1$ vector of regularization parameters `lambda` used to fit the model. `lambda` is displayed in descending order.

`beta0` $L \times 1$ vector of estimated intercepts. The kth entry in `beta0` corresponds to the kth regularization parameter in `lambda`.

`beta` $p \times L$ matrix of estimated regression coefficients. The kth column in `beta` corresponds to the kth regularization parameter in `lambda`.

`mu.pred` $n_{test} \times L$ matrix of predicted mean response values $\mu_{test} = E(Y_{test})$ based on the *test* data in `X.test` (or training data `X` if no argument was specified for `X.test`). The kth column in `mu.pred` corresponds to the predictions for the kth regularization parameter in `lambda`.

`classifications` $G \times L$ matrix of classifications, where G is the number of groups. An entry of "1" indicates that the group was classified as nonzero, and an entry of "0" indicates that the group was classified as zero. The kth column of `classifications` corresponds to the kth regularization parameter in `lambda`.

`loss` $L \times 1$ vector of negative log-likelihood of the fitted models. The kth entry in `loss` corresponds to the kth regularization parameter in `lambda`.

References

Breheny, P. and Huang, J. (2015). "Group descent algorithms for nonconvex penalized linear and logistic regression models with grouped predictors." *Statistics and Computing*, **25**:173-187.

Wang, H. and Leng, C. (2007). "Unified LASSO estimation by least squares approximation." *Journal of the American Statistical Association*, **102**:1039-1048.

Examples

```
## Generate training data
set.seed(1234)
X = matrix(runif(100*16), nrow=100)
n = dim(X)[1]
groups = c("A","A","A","B","B","B","C","C","D","E","E","F","G","H","H","H")
groups = as.factor(groups)
true.beta = c(-2,2,2,0,0,0,0,0,0,1.5,-1.5,0,0,-2,2,2)

## Generate count responses from negative binomial regression
eta = crossprod(t(X), true.beta)
y = rnbinom(n,size=1, mu=exp(eta))

## Generate test data
n.test = 50
X.test = matrix(runif(n.test*16), nrow=n.test)

## Fit negative binomial regression models with the group SCAD penalty
nb.mod = grpreg.nb(y, X, X.test, groups, penalty="gSCAD")

## Tuning parameters used to fit models
nb.mod$lambda
```

```
# Predicted n.test-dimensional vectors mu=E(Y.test) based on test data, X.test.
# The kth column of 'mu.pred' corresponds to the kth entry in 'lambda.'
nb.mod$mu.pred

# Classifications of the 8 groups. The kth column of 'classifications'
# corresponds to the kth entry in lambda.
nb.mod$classifications
```

SBGAM *Sparse Bayesian Generalized Additive Models*

Description

This function implements sparse Bayesian generalized additive models (GAMs) with the spike-and-slab group lasso (SSGL) penalty. Let y_i denote the ith response and x_i denote a p-dimensional vector of covariates. GAMs are of the form,

$$g(E(y_i)) = \beta_0 + \sum_{j=1}^{p} f_j(x_{ij}), i = 1, ..., n,$$

where g is a monotone increasing link function. The identity link function is used for Gaussian regression, the logit link is used for binomial regression, and the log link is used for Poisson, negative binomial, and gamma regression. With the SSGL penalty, some of the univariate functions $f_j(x_j)$ will be estimated as $\hat{f}_j(x_j) = 0$, depending on the size of the spike hyperparameter λ_0 in the SSGL prior. The functions $f_j(x_j), j = 1, ..., p$, are modeled using B-spline basis expansions.

There is another implementation of sparse Gaussian GAMs with the SSGL penalty available at https://github.com/jantonelli111/SSGL, which uses natural cubic splines as the basis functions. This package `sparseGAM` uses B-spline basis functions and also implements sparse GAMs with the SSGL penalty for binomial, Poisson, negative binomial, and gamma regression.

For implementation of sparse *frequentist* GAMs with the group LASSO, group SCAD, and group MCP penalties, use the `SFGAM` function.

Usage

```
SBGAM(y, X, X.test, df=6,
      family=c("gaussian","binomial","poisson","negativebinomial","gamma"),
      nb.size=1, gamma.shape=1, nlambda0=20, lambda0, lambda1, a, b,
      max.iter=100, tol = 1e-6, print.iter=TRUE)
```

Arguments

`y`	$n \times 1$ vector of responses for training data.
`X`	$n \times p$ design matrix for training data, where the jth column of `X` corresponds to the jth overall covariate.

X.test	$n_{test} \times p$ design matrix for test data to calculate predictions. X.test must have the *same* number of columns as X, but not necessarily the same number of rows. If *no* test data is provided or if in-sample predictions are desired, then the function automatically sets X.test=X in order to calculate *in-sample* predictions.
df	number of B-spline basis functions to use in each basis expansion. Default is df=6, but the user may specify degrees of freedom as any integer greater than or equal to 3.
family	exponential dispersion family. Allows for "gaussian", "binomial", "poisson", "negativebinomial", and "gamma". Note that for "negativebinomial", the size parameter must be specified, while for "gamma", the shape parameter must be specified.
nb.size	known size parameter α in $NB(\alpha, \mu_i)$ distribution for negative binomial responses. Default is nb.size=1. Ignored if family is not "negativebinomial".
gamma.shape	known shape parameter ν in $Gamma(\mu_i, \nu)$ distribution for gamma responses. Default is gamma.shape=1. Ignored if family is not "gamma".
nlambda0	number of spike hyperparameter L. Default is nlambda0=20.
lambda0	grid of L spike hyperparameters λ_0. The user may specify either a scalar or a vector. If the user does not provide this, the program chooses the grid automatically.
lambda1	slab hyperparameter λ_1 in the SSGL prior. Default is lambda1=1.
a	shape hyperparameter for the $Beta(a, b)$ prior on the mixing proportion in the SSGL prior. Default is a=1.
b	shape hyperparameter for the $Beta(a, b)$ prior on the mixing proportion in the SSGL prior. Default is b=dim(X)[2].
max.iter	maximum number of iterations in the algorithm. Default is max.iter=100.
tol	convergence threshold for algorithm. Default is tol=1e-6.
print.iter	Boolean variable for whether or not to print the current nlambda0 in the algorithm. Default is print.iter=TRUE.

Value

The function returns a list containing the following components:

lambda0	$L \times 1$ vector of spike hyperparameters lambda0 used to fit the model. lambda0 is displayed in descending order.
f.pred	List of L $n_{test} \times p$ matrices, where the kth matrix in the list corresponds to the kth spike hyperparameter in lambda0. The jth column in each matrix in f.pred is the estimate of the jth function evaluated on the test data in X.test for the jth covariate (or training data X if X.test was not specified).
mu.pred	$n_{test} \times L$ matrix of predicted mean response values $\mu_{test} = E(Y_{test})$ based on the *test* data in X.test (or training data X if no argument was specified for X.test). The kth column in mu.pred corresponds to the predictions for the kth spike hyperparameter in lambda0.

`classifications`
: $p \times L$ matrix of classifications. An entry of "1" indicates that the corresponding function was classified as nonzero, and an entry of "0" indicates that the function was classified as zero. The kth column of `classifications` corresponds to the kth spike hyperparameter in `lambda0`.

`beta0`
: $L \times 1$ vector of estimated intercepts. The kth entry in `beta0` corresponds to the kth spike hyperparameter in `lambda0`.

`beta`
: $dp \times L$ matrix of estimated basis coefficients. The kth column in `beta` corresponds to the kth spike hyperparameter in `lambda0`.

`loss`
: vector of either the residual sum of squares (`"gaussian"`) or the negative log-likelihood (`"binomial"`, `"poisson"`, `"negativebinomial"`, `"gamma"`) of the fitted model. The kth entry in `loss` corresponds to the kth spike hyperparameter in `lambda0`.

References

Bai R. (2021). "Spike-and-slab group lasso for consistent Bayesian estimation and variable selection in non-Gaussian generalized additive models." *arXiv pre-print arXiv:2007.07021*.

Bai, R., Moran, G. E., Antonelli, J. L., Chen, Y., and Boland, M.R. (2021). "Spike-and-slab group lassos for grouped regression and sparse generalized additive models." *Journal of the American Statistical Association*, in press.

Examples

```
## Generate data
set.seed(12345)
X = matrix(runif(100*5), nrow=100)
n = dim(X)[1]
y = 3*sin(2*pi*X[,1])-3*cos(2*pi*X[,2]) + rnorm(n)

## Test data with 30 observations
X.test = matrix(runif(30*5), nrow=30)

## Fit sparse Bayesian generalized additive model to data with the SSGL penalty
## and 5 spike hyperparameters
SBGAM.mod = SBGAM(y, X, X.test, family="gaussian", lambda0=seq(from=50,to=10,by=-10))

## The model corresponding to the 1st spike hyperparameter
SBGAM.mod$lambda[1]
SBGAM.mod$classifications[,1]

## Plot first function f_1(x_1) in 2nd model
x1 = X.test[,1]
## Estimates of all 20 function evaluations on test data
f.hat = SBGAM.mod$f.pred[[1]]
## Extract estimates of f_1
f1.hat = f.hat[,1]

## Plot X_1 against f_1(x_1)
plot(x1[order(x1)], f1.hat[order(x1)], xlab=expression(x[1]),
```

```
ylab=expression(f[1](x[1])))
```

SFGAM *Sparse Frequentist Generalized Additive Models*

Description

This function implements sparse frequentist generalized additive models (GAMs) with the group LASSO, group SCAD, and group MCP penalties. Let y_i denote the ith response and x_i denote a p-dimensional vector of covariates. GAMs are of the form,

$$g(E(y_i)) = \beta_0 + \sum_{j=1}^{p} f_j(x_{ij}), i = 1, ..., n,$$

where g is a monotone increasing link function. The identity link function is used for Gaussian regression, the logit link is used for binomial regression, and the log link is used for Poisson, negative binomial, and gamma regression. The univariate functions are estimated using linear combinations of B-spline basis functions. Under group regularization of the basis coefficients, some of the univariate functions $f_j(x_j)$ will be estimated as $\hat{f}_j(x_j) = 0$, depending on the size of the regularization parameter λ.

For implementation of sparse *Bayesian* GAMs with the SSGL penalty, use the `SBGAM` function.

Usage

```
SFGAM(y, X, X.test, df=6,
      family=c("gaussian","binomial", "poisson", "negativebinomial","gamma"),
      nb.size=1, gamma.shape=1, penalty=c("gLASSO","gMCP","gSCAD"), taper,
      nlambda=100, lambda, max.iter=10000, tol=1e-4)
```

Arguments

`y`	$n \times 1$ vector of responses for training data.
`X`	$n \times p$ design matrix for training data, where the jth column of `X` corresponds to the jth overall covariate.
`X.test`	$n_{test} \times p$ design matrix for test data to calculate predictions. `X.test` must have the *same* number of columns as `X`, but not necessarily the same number of rows. If *no* test data is provided or if in-sample predictions are desired, then the function automatically sets `X.test=X` in order to calculate *in-sample* predictions.
`df`	number of B-spline basis functions to use in each basis expansion. Default is `df=6`, but the user may specify degrees of freedom as any integer greater than or equal to 3.
`family`	exponential dispersion family. Allows for `"gaussian"`, `"binomial"`, `"poisson"`, `"negativebinomial"`, and `"gamma"`. Note that for `"negativebinomial"`, the size parameter must be specified, while for `"gamma"`, the shape parameter must be specified.

nb.size	known size parameter α in $NB(\alpha, \mu_i)$ distribution for negative binomial responses. Default is nb.size=1. Ignored if family is not "negativebinomial".
gamma.shape	known shape parameter ν in $Gamma(\mu_i, \nu)$ distribution for gamma responses. Default is gamma.shape=1. Ignored if family is not "gamma".
penalty	group regularization method to use on the groups of basis coefficients. The options are "gLASSO", "gSCAD", and "gMCP". To implement sparse GAMs with the SSGL penalty, use the SBGAM function.
taper	tapering term γ in group SCAD and group MCP controlling how rapidly the penalty tapers off. Default is taper=4 for group SCAD and taper=3 for group MCP. Ignored if "gLASSO" is specified as the penalty.
nlambda	number of regularization parameters L. Default is nlambda=100.
lambda	grid of L regularization parameters. The user may specify either a scalar or a vector. If the user does not provide this, the program chooses the grid automatically.
max.iter	maximum number of iterations in the algorithm. Default is max.iter=10000.
tol	convergence threshold for algorithm. Default is tol=1e-4.

Value

The function returns a list containing the following components:

lambda	$L \times 1$ vector of regularization parameters lambda used to fit the model. lambda is displayed in descending order.
f.pred	List of L $n_{test} \times p$ matrices, where the kth matrix in the list corresponds to the kth regularization parameter in lambda. The jth column in each matrix in f.pred is the estimate of the jth function evaluated on the test data in X.test for the jth covariate (or training data X if X.test was not specified).
mu.pred	$n_{test} \times L$ matrix of predicted mean response values $\mu_{test} = E(Y_{test})$ based on the *test* data in X.test (or training data X if no argument was specified for X.test). The kth column in mu.pred corresponds to the predictions for the kth regularization parameter in lambda.
classifications	$p \times L$ matrix of classifications. An entry of "1" indicates that the corresponding function was classified as nonzero, and an entry of "0" indicates that the function was classified as zero. The kth column of classifications corresponds to the kth regularization parameter in lambda.
beta0	$L \times 1$ vector of estimated intercepts. The kth entry in beta0 corresponds to the kth regularization parameter in lambda.
beta	$dp \times L$ matrix of estimated basis coefficients. The kth column in beta corresponds to the kth regularization parameter in lambda.
loss	vector of either the residual sum of squares ("gaussian") or the negative log-likelihood ("binomial", "poisson", "negativebinomial", "gamma") of the fitted model. The kth entry in loss corresponds to the kth regularization parameter in lambda.

References

Breheny, P. and Huang, J. (2015). "Group descent algorithms for nonconvex penalized linear and logistic regression models with grouped predictors." *Statistics and Computing*, **25**:173-187.

Wang, H. and Leng, C. (2007). "Unified LASSO estimation by least squares approximation." *Journal of the American Statistical Association*, **102**:1039-1048.

Yuan, M. and Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **68**: 49-67.

Examples

```
## Generate data
set.seed(12345)
X = matrix(runif(100*20), nrow=100)
n = dim(X)[1]
y = 5*sin(2*pi*X[,1])-5*cos(2*pi*X[,2]) + rnorm(n)

## Test data with 50 observations
X.test = matrix(runif(50*20), nrow=50)

## K-fold cross-validation with group MCP penalty
cv.mod = cv.SFGAM(y, X, family="gaussian", penalty="gMCP")
## Plot CVE curve
plot(cv.mod$lambda, cv.mod$cve, type="l", xlab="lambda", ylab="CVE")
## lambda which minimizes cross-validation error
lambda.opt = cv.mod$lambda.min

## Fit a single model with lambda.opt
SFGAM.mod = SFGAM(y, X, X.test, penalty="gMCP", lambda=lambda.opt)

## Classifications
SFGAM.mod$classifications
## Predicted function evaluations on test data
f.pred = SFGAM.mod$f.pred

## Plot estimated first function
x1 = X.test[,1]
f1.hat = f.pred[,1]

## Plot x_1 against f_1(x_1)
plot(x1[order(x1)], f1.hat[order(x1)], xlab=expression(x[1]),
     ylab=expression(f[1](x[1])))
```

SSGL *Spike-and-Slab Group Lasso Regression*

Description

This is a stand-alone function for group-regularized regression models in the exponential dispersion family with the spike-and-slab group lasso (SSGL) penalty. Let y_i denote the ith response and x_i denote a p-dimensional vector of covariates. We fit models of the form,

$$g(E(y_i)) = \beta_0 + x_i^T \beta, i = 1, ..., n,$$

where g is a monotone increasing link function. The identity link function is used for Gaussian regression, the logit link is used for binomial regression, and the log link is used for Poisson, negative binomial, and gamma regression.

If the covariates in each x_i are grouped according to known groups $g = 1, ..., G$, then this function may estimate some of the G groups of coefficients as all zero, depending on the amount of regularization.

Another implementation of the SSGL model for Gaussian regression models is available on Github at https://github.com/jantonelli111/SSGL. This package `sparseGAM` also implements the SSGL model for binomial, Poisson, negative binomial, and gamma regression.

Usage

```
SSGL(y, X, X.test, groups,
     family=c("gaussian","binomial","poisson","negativebinomial","gamma"),
     nb.size=1, gamma.shape=1, weights, nlambda0=20, lambda0, lambda1, a, b,
     max.iter=100, tol = 1e-6, print.iter=TRUE)
```

Arguments

`y`	$n \times 1$ vector of responses for training data.
`X`	$n \times p$ design matrix for training data, where the jth column of `X` corresponds to the jth overall covariate.
`X.test`	$n_{test} \times p$ design matrix for test data to calculate predictions. `X.test` must have the *same* number of columns as `X`, but not necessarily the same number of rows. If *no* test data is provided or if in-sample predictions are desired, then the function automatically sets `X.test=X` in order to calculate *in-sample* predictions.
`groups`	p-dimensional vector of group labels. The jth entry in `groups` should contain either the group number *or* the name of the factor level that the jth covariate belongs to. `groups` must be either a vector of integers or factors.
`family`	exponential dispersion family. Allows for `"gaussian"`, `"binomial"`, `"poisson"`, `"negativebinomial"`, and `"gamma"`. Note that for `"negativebinomial"`, the size parameter must be specified, while for `"gamma"`, the shape parameter must be specified.
`nb.size`	known size parameter α in $NB(\alpha, \mu_i)$ distribution for the negative binomial responses. Default is `nb.size=1`. Ignored if `family` is not `"negativebinomial"`.
`gamma.shape`	known shape parameter ν in $Gamma(\mu_i, \nu)$ distribution for gamma responses. Default is `gamma.shape=1`. Ignored if `family` is not `"gamma"`.
`weights`	group-specific, nonnegative weights for the penalty. Default is to use the square roots of the group sizes.

`nlambda0`	number of spike hyperparameters L. Default is `nlambda0=20`.
`lambda0`	grid of L spike hyperparameters λ_0. The user may specify either a scalar or a vector. If the user does not provide this, the program chooses the grid automatically.
`lambda1`	slab hyperparameter λ_1 in the SSGL prior. Default is `lambda1=1`.
`a`	shape hyperparameter for the $Beta(a, b)$ prior on the mixing proportion in the SSGL prior. Default is `a=1`.
`b`	shape hyperparameter for the $Beta(a, b)$ prior on the mixing proportion in the SSGL prior. Default is `b=dim(X)[2]`.
`max.iter`	maximum number of iterations in the algorithm. Default is `max.iter=100`.
`tol`	convergence threshold for algorithm. Default is `tol=1e-6`.
`print.iter`	Boolean variable for whether or not to print the current `nlambda0` in the algorithm. Default is `print.iter=TRUE`.

Value

The function returns a list containing the following components:

`lambda0`	$L \times 1$ vector of spike hyperpameters `lambda0` used to fit the model. `lambda0` is displayed in descending order.
`beta0`	$L \times 1$ vector of estimated intercepts. The kth entry in `beta0` corresponds to the kth spike hyperparameter in `lambda0`.
`beta`	$p \times L$ matrix of estimated regression coefficients. The kth column in `beta` corresponds to the kth spike hyperparameter in `lambda0`.
`mu.pred`	$n_{test} \times L$ matrix of predicted mean response values $\mu_{test} = E(Y_{test})$ based on the *test* data in `X.test` (or training data `X` if no argument was specified for`X.test`). The kth column in `mu.pred` corresponds to the predictions for the kth spike hyperparameter in `lambda0`.
`classifications`	$G \times L$ matrix of classifications, where G is the number of groups. An entry of "1" indicates that the group was classified as nonzero, and an entry of "0" indicates that the group was classified as zero. The kth column of `classifications` corresponds to the kth spike hyperparameter in `lambda0`.
`loss`	vector of either the residual sum of squares (`"gaussian"`) or the negative log-likelihood (`"binomial"`, `"poisson"`, `"negativebinomial"`, `"gamma"`) of the fitted model. The kth entry in `loss` corresponds to the kth spike hyperparameter in `lambda0`.

References

Bai R. (2021). "Spike-and-slab group lasso for consistent Bayesian estimation and variable selection in non-Gaussian generalized additive models." *arXiv pre-print arXiv:2007.07021*.

Bai, R., Moran, G. E., Antonelli, J. L., Chen, Y., and Boland, M.R. (2021). "Spike-and-slab group lassos for grouped regression and sparse generalized additive models." *Journal of the American Statistical Association*, in press.

Examples

```
## Generate data
set.seed(12345)
X = matrix(runif(100*10), nrow=100)
n = dim(X)[1]
groups = c("A","A","A","B","B","B","C","C","D","D")
groups = as.factor(groups)
true.beta = c(-2.5,1.5,1.5,0,0,0,2,-2,0,0)

## Generate responses from Gaussian distribution
y = crossprod(t(X),true.beta) + rnorm(n)

## Generate test data
n.test = 50
X.test = matrix(runif(n.test*10), nrow=n.test)

## Fit SSGL model with 10 spike hyperparameters
## Note that if user does not specify lambda0, the SSGL function chooses a grid automatically.

SSGL.mod = SSGL(y, X, X.test, groups, family="gaussian", lambda0=seq(from=50,to=5,by=-5))

## Regression coefficient estimates
SSGL.mod$beta

# Predicted n.test-dimensional vectors mu=E(Y.test) based on test data, X.test.
# The kth column of 'mu.pred' corresponds to the kth entry in 'lambda.'
SSGL.mod$mu.pred

# Classifications of the 8 groups. The kth column of 'classifications'
# corresponds to the kth entry in 'lambda.'
SSGL.mod$classifications


## Example with binomial regression

## Generate binary responses
eta = crossprod(t(X), true.beta)
y = rbinom(n, size=1, prob=1/(1+exp(-eta)))

## Fit SSGL model with 10 spike hyperparameters
## Note that if user does not specify lambda0, the SSGL function chooses a grid automatically.

SSGL.mod = SSGL(y, X, X.test, groups, family="binomial",
lambda0=seq(from=10,to=1,by=-1))

## Predicted probabilities of success mu=E(Y.test) based on test data, X.test
SSGL.mod$mu.pred

## Classifications of the 8 groups.
SSGL.mod$classifications
```

Index