Package 'hqreg'

February 16, 2017

Type Package

Title Regularization Paths for Lasso or Elastic-Net Penalized Huber Loss Regression and Quantile Regression

Version 1.4

Date 2017-2-15

Author Congrui Yi

Maintainer Congrui Yi <congrui-yi@uiowa.edu>

Description Efficient algorithms for fitting regularization paths for lasso or elastic-net penalized regression models with Huber loss, quantile loss or squared loss.

License GPL-3

URL http://arxiv.org/abs/1509.02957

Imports parallel

NeedsCompilation yes

Repository CRAN

Date/Publication 2017-02-16 07:13:26

R topics documented:

`hqreg-package` *Regularization Paths for Lasso or Elastic-net Penalized Huber Loss Regression and Quantile Regression*

Description

Efficient algorithms for fitting regularization paths for lasso or elastic-net penalized regression models with Huber loss, quantile loss or squared loss.

Details

Package:	hqreg
Type:	Package
Version:	1.4
Date:	2017-2-15
License:	GPL-3

Very simple to use. Accepts `X,y` data for regression models, and produces the regularization path over a grid of values for the tuning parameter `lambda`. Also provides functions for plotting, prediction and parallelized cross-validation.

Author(s)

Congrui Yi <congrui-yi@uiowa.edu>

References

Yi, C. and Huang, J. (2016) *Semismooth Newton Coordinate Descent Algorithm for Elastic-Net Penalized Huber Loss Regression and Quantile Regression*, https://arxiv.org/abs/1509.02957
Journal of Computational and Graphical Statistics, accepted in Nov 2016
http://www.tandfonline.com/doi/full/10.1080/10618600.2016.1256816

Examples

```
X = matrix(rnorm(1000*100), 1000, 100)
beta = rnorm(10)
eps = 4*rnorm(1000)
y = drop(X[,1:10] %*% beta + eps)

# Huber loss
fit1 = hqreg(X, y)
coef(fit1, 0.01)
predict(fit1, X[1:5,], lambda = c(0.02, 0.01))
cv.fit1 = cv.hqreg(X, y)
plot(cv.fit1)
```

```
# Quantile loss
fit2 = hqreg(X, y, method = "quantile", tau = 0.2)
plot(fit2)

# Squared loss
fit3 = hqreg(X, y, method = "ls", preprocess = "rescale")
plot(fit3, xvar = "norm")
```

`cv.hqreg` *Cross-validation for hqreg*

Description

Perform k-fold cross validation for elastic-net penalized Huber loss regression and quantile regression over a sequence of lambda values and find an optimal lambda.

Usage

```
cv.hqreg(X, y, ..., FUN = c("hqreg", "hqreg_raw"), ncores = 1, nfolds = 10, fold.id,
         type.measure = c("deviance", "mse", "mae"), seed)
```

Arguments

`X`	The input matrix.
`y`	The response vector.
`...`	Additional arguments to `FUN`.
`FUN`	Model fitting function. The default is "hqreg" which preprocesses the data internally. The other option is "hqreg_raw" which uses the raw data as is.
`ncores`	`cv.hqreg` can be run in parallel across a cluster using the `parallel` package. If `ncores > 1`,a cluster is created to run `cv.hqreg` in parallel. The code is run sequentially if `ncores = 1` (the default). A message is printed if `ncores` is larger than the total number of available cores, and all available cores will be used.
`nfolds`	The number of cross-validation folds. Default is 10.
`fold.id`	(Optional) a vector of values between 1 and nfold indicating which fold each observation belongs to. If supplied, nfolds can be missing. By default the observations are randomly assigned by `cv.hqreg`.
`type.measure`	The default is "deviance", which uses the chosen loss function of the model. Other options include "mse" for mean squared error and "mae" for mean absolute error.
`seed`	(Optional) Seed for the random number generator in order to obtain reproducible results.

Details

The function randomly partitions the data in `nfolds`. It calls `hqreg` `nfolds+1` times, the first to obtain the `lambda` sequence, and the remainder to fit with each of the folds left out once for validation. The cross-validation error is the average of validation errors for the `nfolds` fits.

Note that `cv.hqreg` does not search for values of `alpha`, `gamma` or `tau`. Specific values should be supplied, otherwise the default ones for `hqreg` are used. If users would like to cross-validate `alpha`, `gamma` or `tau` as well, they should call `cv.hqreg` for each combination of these parameters and use the same "seed" in these calls so that the partitioning remains the same.

Value

The function returns an object of S3 class `"cv.hqreg"`, which is a list containing:

`cve`	The error for each value of `lambda`, averaged across the cross-validation folds.
`cvse`	The estimated standard error associated with each value of `cve`.
`type.measure`	Same as above.
`lambda`	The values of `lambda` used in the cross-validation fits.
`fit`	The fitted `hqreg` object for the whole data.
`lambda.1se`	The largest `lambda` such that the error is within 1 standard error of the minimum.
`lambda.min`	The value of `lambda` with the minimum cross-validation error.

Author(s)

Congrui Yi <congrui-yi@uiowa.edu>

References

Yi, C. and Huang, J. (2016) *Semismooth Newton Coordinate Descent Algorithm for Elastic-Net Penalized Huber Loss Regression and Quantile Regression*, https://arxiv.org/abs/1509.02957
Journal of Computational and Graphical Statistics, accepted in Nov 2016
http://www.tandfonline.com/doi/full/10.1080/10618600.2016.1256816

See Also

`hqreg`, `plot.cv.hqreg`

Examples

```
X = matrix(rnorm(1000*100), 1000, 100)
beta = rnorm(10)
eps = 4*rnorm(1000)
y = drop(X[,1:10] %*% beta + eps)
cv = cv.hqreg(X, y, seed = 123)
plot(cv)

cv_raw = cv.hqreg(X, y, FUN = "hqreg_raw", seed = 321)
predict(cv_raw, X[1:5,])
```

```
# parallel cross validation
## Not run:
cv_parallel = cv.hqreg(X, y, ncores = 5)
plot(cv_parallel)

## End(Not run)
```

`hqreg` *Fit a robust regression model with Huber or quantile loss penalized by lasso or elasti-net*

Description

Fit solution paths for Huber loss regression or quantile regression penalized by lasso or elastic-net over a grid of values for the regularization parameter lambda.

Usage

```
hqreg(X, y, method = c("huber", "quantile", "ls"),
   gamma = IQR(y)/10, tau = 0.5, alpha = 1, nlambda = 100, lambda.min = 0.05, lambda,
    preprocess = c("standardize", "rescale"), screen = c("ASR", "SR", "none"),
   max.iter = 10000, eps = 1e-7, dfmax = ncol(X)+1, penalty.factor = rep(1, ncol(X)),
    message = FALSE)
```

Arguments

`X`	Input matrix.
`y`	Response vector.
`method`	The loss function to be used in the model. Either "huber" (default), "quantile", or "ls" for least squares (see `Details`).
`gamma`	The tuning parameter of Huber loss, with no effect for the other loss functions. Huber loss is quadratic for absolute values less than gamma and linear for those greater than gamma. The default value is IQR(y)/10.
`tau`	The tuning parameter of the quantile loss, with no effect for the other loss functions. It represents the conditional quantile of the response to be estimated, so must be a number between 0 and 1. It includes the absolute loss when tau = 0.5 (default).
`alpha`	The elastic-net mixing parameter that controls the relative contribution from the lasso and the ridge penalty. It must be a number between 0 and 1. `alpha=1` is the lasso penalty and `alpha=0` the ridge penalty.
`nlambda`	The number of lambda values. Default is 100.
`lambda.min`	The smallest value for lambda, as a fraction of lambda.max, the data derived entry value. Default is 0.05.

lambda	A user-specified sequence of lambda values. Typical usage is to leave blank and have the program automatically compute a lambda sequence based on nlambda and lambda.min. Specifying lambda overrides this. This argument should be used with care and supplied with a decreasing sequence instead of a single value. To get coefficients for a single lambda, use coef or predict instead after fitting the solution path with hqreg or performing k-fold CV with cv.hqreg.
preprocess	Preprocessing technique to be applied to the input. Either "standardize" (default) or "rescale"(see Details). The coefficients are always returned on the original scale.
screen	Screening rule to be applied at each lambda that discards variables for speed. Either "ASR" (default), "SR" or "none". "SR" stands for the strong rule, and "ASR" for the adaptive strong rule. Using "ASR" typically requires fewer iterations to converge than "SR", but the computing time are generally close. Note that the option "none" is used mainly for debugging, which may lead to much longer computing time.
max.iter	Maximum number of iterations. Default is 10000.
eps	Convergence threshold. The algorithms continue until the maximum change in the objective after any coefficient update is less than eps times the null deviance. Default is 1E-7.
dfmax	Upper bound for the number of nonzero coefficients. The algorithm exits and returns a partial path if dfmax is reached. Useful for very large dimensions.
penalty.factor	A numeric vector of length equal to the number of variables. Each component multiplies lambda to allow differential penalization. Can be 0 for some variables, in which case the variable is always in the model without penalization. Default is 1 for all variables.
message	If set to TRUE, hqreg will inform the user of its progress. This argument is kept for debugging. Default is FALSE.

Details

The sequence of models indexed by the regularization parameter lambda is fit using a semismooth Newton coordinate descent algorithm. The objective function is defined to be

$$\frac{1}{n}\sum loss_i + \lambda \text{penalty}.$$

For method = "huber",

$$loss(t) = \frac{t^2}{2\gamma} I(|t| \le \gamma) + (|t| - \frac{\gamma}{2};) I(|t| > \gamma)$$

for method = "quantile",

$$loss(t) = t(\tau - I(t < 0));$$

for method = "ls",

$$loss(t) = \frac{t^2}{2}$$

In the model, "t" is replaced by residuals.

The program supports different types of preprocessing techniques. They are applied to each column of the input matrix `X`. Let x be a column of `X`. For `preprocess = "standardize"`, the formula is

$$x' = \frac{x - mean(x)}{sd(x)};$$

for `preprocess = "rescale"`,

$$x' = \frac{x - min(x)}{max(x) - min(x)}.$$

The models are fit with preprocessed input, then the coefficients are transformed back to the original scale via some algebra. To fit a model for raw data with no preprocessing, use `hqreg_raw`.

Value

The function returns an object of S3 class `"hqreg"`, which is a list containing:

`call`	The call that produced this object.
`beta`	The fitted matrix of coefficients. The number of rows is equal to the number of coefficients, and the number of columns is equal to `nlambda`. An intercept is included.
`iter`	A vector of length `nlambda` containing the number of iterations until convergence at each value of `lambda`.
`saturated`	A logical flag for whether the number of nonzero coefficients has reached `dfmax`.
`lambda`	The sequence of regularization parameter values in the path.
`alpha`	Same as above.
`gamma`	Same as above. `NULL` except when `method = "huber"`.
`tau`	Same as above. `NULL` except when `method = "quantile"`.
`penalty.factor`	Same as above.
`method`	Same as above.
`nv`	The variable screening rules are accompanied with checks of optimality conditions. When violations occur, the program adds in violating variables and re-runs the inner loop until convergence. `nv` is the number of violations.

Author(s)

Congrui Yi <congrui-yi@uiowa.edu>

References

Yi, C. and Huang, J. (2016) *Semismooth Newton Coordinate Descent Algorithm for Elastic-Net Penalized Huber Loss Regression and Quantile Regression*, https://arxiv.org/abs/1509.02957
Journal of Computational and Graphical Statistics, accepted in Nov 2016
http://www.tandfonline.com/doi/full/10.1080/10618600.2016.1256816

See Also

`plot.hqreg`, `cv.hqreg`

Examples

```
X = matrix(rnorm(1000*100), 1000, 100)
beta = rnorm(10)
eps = 4*rnorm(1000)
y = drop(X[,1:10] %*% beta + eps)

# Huber loss
fit1 = hqreg(X, y)
coef(fit1, 0.01)
predict(fit1, X[1:5,], lambda = c(0.02, 0.01))

# Quantile loss
fit2 = hqreg(X, y, method = "quantile", tau = 0.2)
plot(fit2)

# Squared loss
fit3 = hqreg(X, y, method = "ls", preprocess = "rescale")
plot(fit3, xvar = "norm")
```

`hqreg_raw` *Fit a robust regression model on raw data with Huber or quantile loss penalized by lasso or elasti-net*

Description

On raw data without internal data preprocessing, fit solution paths for Huber loss regression or quantile regression penalized by lasso or elastic-net over a grid of values for the regularization parameter lambda.

Usage

```
hqreg_raw(X, y, method = c("huber", "quantile", "ls"),
   gamma = IQR(y)/10, tau = 0.5, alpha = 1, nlambda = 100, lambda.min = 0.05, lambda,
    intercept = TRUE, screen = c("ASR", "SR", "none"),
   max.iter = 10000, eps = 1e-7, dfmax = ncol(X)+1, penalty.factor = rep(1, ncol(X)),
    message = FALSE)
```

Arguments

`X`	Input matrix.
`y`	Response vector.
`method`	The loss function to be used in the model. Either "huber" (default), "quantile", or "ls" for least squares (see `Details`).
`gamma`	The tuning parameter of Huber loss, with no effect for the other loss functions. Huber loss is quadratic for absolute values less than gamma and linear for those greater than gamma. The default value is IQR(y)/10.

tau	The tuning parameter of the quantile loss, with no effect for the other loss functions. It represents the conditional quantile of the response to be estimated, so must be a number between 0 and 1. It includes the absolute loss when tau = 0.5 (default).
alpha	The elastic-net mixing parameter that controls the relative contribution from the lasso and the ridge penalty. It must be a number between 0 and 1. `alpha=1` is the lasso penalty and `alpha=0` the ridge penalty.
nlambda	The number of lambda values. Default is 100.
lambda.min	The smallest value for lambda, as a fraction of lambda.max, the data derived entry value. Default is 0.05.
lambda	A user-specified sequence of lambda values. Typical usage is to leave blank and have the program automatically compute a `lambda` sequence based on `nlambda` and `lambda.min`. Specifying `lambda` overrides this. This argument should be used with care and supplied with a decreasing sequence instead of a single value. To get coefficients for a single `lambda`, use `coef` or `predict` instead after fitting the solution path with `hqreg` or performing k-fold CV with `cv.hqreg`.
intercept	Should an intercept be included? Default is TRUE.
screen	Screening rule to be applied at each `lambda` that discards variables for speed. Either "ASR" (default), "SR" or "none". "SR" stands for the strong rule, and "ASR" for the adaptive strong rule. Using "ASR" typically requires fewer iterations to converge than "SR", but the computing time are generally close. Note that the option "none" is used mainly for debugging, which may lead to much longer computing time.
max.iter	Maximum number of iterations. Default is 10000.
eps	Convergence threshold. The algorithms continue until the maximum change in the objective after any coefficient update is less than `eps` times the null deviance. Default is `1E-7`.
dfmax	Upper bound for the number of nonzero coefficients. The algorithm exits and returns a partial path if `dfmax` is reached. Useful for very large dimensions.
penalty.factor	A numeric vector of length equal to the number of variables. Each component multiplies `lambda` to allow differential penalization. Can be 0 for some variables, in which case the variable is always in the model without penalization. Default is 1 for all variables.
message	If set to TRUE, hqreg will inform the user of its progress. This argument is kept for debugging. Default is FALSE.

Details

The sequence of models indexed by the regularization parameter `lambda` is fit using a semismooth Newton coordinate descent algorithm. The objective function is defined to be

$$\frac{1}{n}\sum loss_i + \lambda \text{penalty}.$$

For `method = "huber"`,

$$loss(t) = \frac{t^2}{2\gamma} I(|t| \le \gamma) + (|t| - \frac{\gamma}{2};) I(|t| > \gamma)$$

for `method = "quantile"`,

$$loss(t) = t(\tau - I(t < 0));$$

for `method = "ls"`,

$$loss(t) = \frac{t^2}{2}$$

In the model, "t" is replaced by residuals.

Value

The function returns an object of S3 class `"hqreg"`, which is a list containing:

`call`	The call that produced this object.
`beta`	The fitted matrix of coefficients. The number of rows is equal to the number of coefficients, and the number of columns is equal to `nlambda`. An intercept is included.
`iter`	A vector of length `nlambda` containing the number of iterations until convergence at each value of `lambda`.
`saturated`	A logical flag for whether the number of nonzero coefficients has reached `dfmax`.
`lambda`	The sequence of regularization parameter values in the path.
`alpha`	Same as above.
`gamma`	Same as above. `NULL` except when `method = "huber"`.
`tau`	Same as above. `NULL` except when `method = "quantile"`.
`penalty.factor`	Same as above.
`method`	Same as above.
`nv`	The variable screening rules are accompanied with checks of optimality conditions. When violations occur, the program adds in violating variables and re-runs the inner loop until convergence. `nv` is the number of violations.

Author(s)

Congrui Yi <congrui-yi@uiowa.edu>

References

Yi, C. and Huang, J. (2016) *Semismooth Newton Coordinate Descent Algorithm for Elastic-Net Penalized Huber Loss Regression and Quantile Regression*, https://arxiv.org/abs/1509.02957
Journal of Computational and Graphical Statistics, accepted in Nov 2016
http://www.tandfonline.com/doi/full/10.1080/10618600.2016.1256816

See Also

`plot.hqreg`, `cv.hqreg`

Examples

```
X = matrix(rnorm(1000*100), 1000, 100)
beta = rnorm(10)
eps = 4*rnorm(1000)
y = drop(X[,1:10] %*% beta) + eps

# Huber loss
# include an intercept by default
fit1 = hqreg_raw(X, y)
coef(fit1, 0.01)
predict(fit1, X[1:5,], lambda = c(0.02, 0.01))

# no intercept
fit2 = hqreg_raw(X, y, intercept = FALSE)
plot(fit2)
```

`plot.cv.hqreg` *Plot the cross-validation curve for a "cv.hqreg" object*

Description

Plot the cross-validation curve for a "cv.hqreg" object against the `lambda` values used, along with standard error bars.

Usage

```
## S3 method for class 'cv.hqreg'
plot(x, log.l = TRUE, nvars = TRUE, ...)
```

Arguments

`x`	A `"cv.hqreg"` object.
`log.l`	Should `log(lambda)` be used instead of `lambda` for X-axis? Default is TRUE.
`nvars`	If `TRUE` (the default), places an axis on top of the plot denoting the number of variables with nonzero coefficients at each `lambda`.
`...`	Other graphical parameters to `plot`

Details

Produces a plot of mean cv errors at each `lambda` along with upper and lower standard error bars.

Author(s)

Congrui Yi <congrui-yi@uiowa.edu>

References

Yi, C. and Huang, J. (2016) *Semismooth Newton Coordinate Descent Algorithm for Elastic-Net Penalized Huber Loss Regression and Quantile Regression*, https://arxiv.org/abs/1509.02957 *Journal of Computational and Graphical Statistics, accepted in Nov 2016* http://www.tandfonline.com/doi/full/10.1080/10618600.2016.1256816

See Also

hqreg, cv.hqreg

Examples

```
X = matrix(rnorm(1000*100), 1000, 100)
beta = rnorm(10)
eps = 4*rnorm(1000)
y = drop(X[,1:10] %*% beta + eps)
cv = cv.hqreg(X, y, seed = 123)
plot(cv)
```

---

plot.hqreg *Plot coefficients from a "hqreg" object*

Description

Produce a plot of the coefficient paths for a fitted "hqreg" object.

Usage

```
## S3 method for class 'hqreg'
plot(x, xvar = c("lambda", "norm"), log.l= TRUE, nvars = TRUE,
    alpha = 1, ...)
```

Arguments

x	A hqreg object.
xvar	What is on the X-axis. "lambda" plots against the lambda sequence, "norm" against the L1-norm of the coefficients. Default is "lambda".
log.l	Should log(lambda) be used instead of lambda when xvar = "lambda"? Default is TRUE. It has no effect for "norm".
nvars	If TRUE (the default), places an axis on top of the plot denoting the number of variables with nonzero coefficients at each lambda.
alpha	A value between 0 and 1 for alpha transparency channel(0 means transparent and 1 means opaque), helpful when the number of variables is large.
...	Other graphical parameters to plot.

Author(s)

Congrui Yi <congrui-yi@uiowa.edu>

References

Yi, C. and Huang, J. (2016) *Semismooth Newton Coordinate Descent Algorithm for Elastic-Net Penalized Huber Loss Regression and Quantile Regression*, https://arxiv.org/abs/1509.02957 *Journal of Computational and Graphical Statistics, accepted in Nov 2016* http://www.tandfonline.com/doi/full/10.1080/10618600.2016.1256816

See Also

hqreg

Examples

```
X = matrix(rnorm(1000*100), 1000, 100)
beta = rnorm(10)
eps = 4*rnorm(1000)
y = drop(X[,1:10] %*% beta + eps)
fit = hqreg(X, y)
par(mfrow = c(2,2))
plot(fit)
plot(fit, nvars = FALSE, alpha = 0.5)
plot(fit, xvar = "norm")
```

---

predict.cv.hqreg *Model predictions based on "cv.hqreg" object.*

Description

This function makes predictions from a cross-validated hqreg model, using the stored fit and the optimal value chosen for `lambda`.

Usage

```
## S3 method for class 'cv.hqreg'
predict(object, X, lambda = c("lambda.1se","lambda.min"),
    type = c("response","coefficients","nvars"), ...)
## S3 method for class 'cv.hqreg'
coef(object, lambda = c("lambda.1se","lambda.min"), ...)
```

Arguments

`object`	Fitted `"hqreg"` model object.
`X`	Matrix of values at which predictions are to be made. Used only for `type = "response"`.

lambda	Values of the regularization parameter lambda at which predictions are requested. Default is the value "lambda.1se" stored on the CV object. Alternatively "lambda.min" can be used. If lambda is numeric, it is taken as the value(s) of lambda to be used.
type	Type of prediction. "response" returns the fitted values; "coefficients" returns the coefficients; "nvars" returns the number of nonzero coefficients at each value of lambda.
...	Not used. Other arguments to predict.

Value

The object returned depends on type.

Author(s)

Congrui Yi <congrui-yi@uiowa.edu>

References

Yi, C. and Huang, J. (2016) *Semismooth Newton Coordinate Descent Algorithm for Elastic-Net Penalized Huber Loss Regression and Quantile Regression*, https://arxiv.org/abs/1509.02957 *Journal of Computational and Graphical Statistics, accepted in Nov 2016* http://www.tandfonline.com/doi/full/10.1080/10618600.2016.1256816

See Also

hqreg cv.hqreg

Examples

```
X = matrix(rnorm(1000*100), 1000, 100)
beta = rnorm(10)
eps = 4*rnorm(1000)
y = drop(X[,1:10] %*% beta + eps)
cv = cv.hqreg(X, y, seed = 1011)
predict(cv, X[1:5,])
predict(cv, X[1:5,], lambda = "lambda.min")
predict(cv, X[1:5,], lambda = 0.05)
```

---

predict.hqreg *Model predictions based on "hqreg" object.*

Description

This function returns fitted values, coefficients and more from a fitted "hqreg" object.

Usage

```
## S3 method for class 'hqreg'
predict(object, X, lambda, type = c("response","coefficients","nvars"),
    exact = FALSE, ...)
## S3 method for class 'hqreg'
coef(object, lambda, exact = FALSE, ...)
```

Arguments

object	Fitted "hqreg" model object.
X	Matrix of values at which predictions are to be made. Used only for type = "response".
lambda	Values of the regularization parameter lambda at which predictions are requested. Default is the entire sequence used to create the model.
type	Type of prediction. "response" returns the fitted values; "coefficients" returns the coefficients; "nvars" returns the number of nonzero coefficients at each value of lambda.
exact	If exact=FALSE (default), then the function uses linear interpolation to make predictions for values of lambda that do not coincide with those used to fit the model. If exact=TRUE, and predictions are requested at values of lambda not included in the original fit, the model is refit on a lambda sequence consisting object$lambda and the new ones before predictions are made.
...	Not used. Other arguments to predict.

Value

The object returned depends on type.

Author(s)

Congrui Yi <congrui-yi@uiowa.edu>

References

Yi, C. and Huang, J. (2016) *Semismooth Newton Coordinate Descent Algorithm for Elastic-Net Penalized Huber Loss Regression and Quantile Regression*, https://arxiv.org/abs/1509.02957
Journal of Computational and Graphical Statistics, accepted in Nov 2016
http://www.tandfonline.com/doi/full/10.1080/10618600.2016.1256816

See Also

hqreg

Examples

```
X = matrix(rnorm(1000*100), 1000, 100)
beta = rnorm(10)
eps = 4*rnorm(1000)
y = drop(X[,1:10] %*% beta + eps)
```

```
fit = hqreg(X, y, method = "quantile", tau = 0.7)
predict(fit, X[1:5,], lambda = c(0.05, 0.01))
predict(fit, X[1:5,], lambda = 0.05, exact = TRUE)
predict(fit, X[1:5,], lambda = 0.05, type = "nvars")
coef(fit, lambda = 0.05)
```

Index