Package ‘penalized’

July 23, 2025

Version 0.9-52

Date 2022-04-23

Title L1 (Lasso and Fused Lasso) and L2 (Ridge) Penalized Estimation in GLMs and in the Cox Model

Author Jelle Goeman, Rosa Meijer, Nimisha Chaturvedi, Matthew Lueder

Maintainer Jelle Goeman `<j.j.goeman@lumc.nl>`

Depends R (>= 2.10.0), survival, methods

Imports Rcpp

LinkingTo Rcpp, RcppArmadillo

Suggests globaltest

Description Fitting possibly high dimensional penalized regression models. The penalty structure can be any combination of an L1 penalty (lasso and fused lasso), an L2 penalty (ridge) and a positivity constraint on the regression coefficients. The supported regression models are linear, logistic and Poisson regression and the Cox Proportional Hazards model. Cross-validation routines allow optimization of the tuning parameters.

License GPL (>= 2)

NeedsCompilation yes

Collate onattach.R penfit.R breslow.R penalized.R core.R checkinput.R cvl.R contrasts.R Brent.R plotpath.R cox.R logit.R linear.R poisson.R

LazyLoad yes

Repository CRAN

Date/Publication 2022-04-23 14:10:02 UTC

RoxygenNote 7.1.1

Contents

`breslow object` *Breslow estimator object*

Description

Stores one or more survival stepfunctions of the Kaplan-Meier and Breslow type.

Details

Breslow objects are generally created by the `penalized` function and related functions.

Slots

`time`**:** Object of class "vector". The time points at which the function starts, jumps or ends.

`curves`**:** Object of class "matrix". The values of the curve at each time point. Note that the function is left-continuous. Curves are stored as a matrix of dimension (\# curves) x (\# timepoints).

Methods

"[" (breslow): Coerces the object to a matrix (as as.matrix) and selects rows and/or columns.

"[[" (breslow): Selects a subset of the curves.

as.data.frame (breslow): Returns the "curves" slot together with the "time" slot in a `data.frame`.

as.list (breslow): Coerces the object to a list.

as.matrix (breslow): Returns the "curves" slot with the "time" slot as column names.

plot (breslow): Straightforward plotting (all curves in one plot).

show (breslow): Summarizes the dimensions of the object.

survival (breslow): Takes a second argument (`time`). Extracts the value of the survival curve(s) at any time point.

Author(s)

Jelle Goeman: `<j.j.goeman@lumc.nl>`

See Also

`penalized`.

`Cross-validation in penalized generalized linear models`
Cross-validated penalized regression

Description

Cross-validating generalized linear models with L1 (lasso or fused lasso) and/or L2 (ridge) penalties, using likelihood cross-validation.

Usage

```
cvl (response, penalized, unpenalized, lambda1 = 0, lambda2= 0, positive = FALSE,
    fusedl = FALSE, data, model = c("cox", "logistic", "linear", "poisson"),
    startbeta, startgamma, fold, epsilon = 1e-10, maxiter, standardize = FALSE,
    trace = TRUE, approximate = FALSE)

optL1 (response, penalized, unpenalized, minlambda1, maxlambda1, base1, lambda2 = 0,
  fusedl = FALSE, positive = FALSE, data,
  model = c("cox", "logistic", "linear", "poisson"), startbeta, startgamma, fold,
  epsilon = 1e-10, maxiter = Inf, standardize = FALSE, tol = .Machine$double.eps^0.25,
  trace = TRUE)

optL2 (response, penalized, unpenalized, lambda1 = 0, minlambda2, maxlambda2, base2,
fusedl = FALSE ,positive = FALSE, data,
  model = c("cox", "logistic", "linear", "poisson"), startbeta, startgamma,
  fold, epsilon = 1e-10, maxiter, standardize = FALSE, tol = .Machine$double.eps^0.25,
  trace = TRUE, approximate = FALSE)

profL1 (response, penalized, unpenalized, minlambda1, maxlambda1, base1, lambda2 = 0,
  fusedl = FALSE,positive = FALSE, data,
  model = c("cox", "logistic", "linear", "poisson"), startbeta, startgamma, fold,
  epsilon = 1e-10, maxiter = Inf, standardize = FALSE, steps = 100, minsteps = steps/3,
  log = FALSE, save.predictions = FALSE, trace = TRUE, plot = FALSE)

profL2 (response, penalized, unpenalized, lambda1 = 0, minlambda2, maxlambda2, base2,
  fusedl = FALSE,positive = FALSE, data,
  model = c("cox", "logistic", "linear", "poisson"), startbeta, startgamma, fold,
  epsilon = 1e-10, maxiter, standardize = FALSE, steps = 100, minsteps = steps/2,
  log = TRUE, save.predictions = FALSE, trace = TRUE, plot = FALSE, approximate = FALSE)
```

Arguments

`response`	The response variable (vector). This should be a numeric vector for linear regression, a `Surv` object for Cox regression and `factor` or a vector of 0/1 values for logistic regression.

`penalized`	The penalized covariates. These may be specified either as a matrix or as a (one-sided) `formula` object. See also under `data`.
`unpenalized`	Additional unpenalized covariates. Specified as under `penalized`. Note that an unpenalized intercept is included in the model by default (except in the cox model). This can be suppressed by specifying `unpenalized = ~0`.
`lambda1`, `lambda2`	The fixed values of the tuning parameters for L1 and L2 penalization. Each must be either a single positive numbers or a vector with length equal to the number of covariates in `penalized` argument. In the latter case, each covariate is given its own penalty weight.
`minlambda1`, `minlambda2`, `maxlambda1`, `maxlambda2`	The values of the tuning parameters for L1 or L2 penalization between which the cross-validated likelihood is to be profiled or optimized. For fused lasso penalty `minlambda2` and `maxlambda2` are the tuning parameters for L1 penalty on the differences of the coefficients between which the cross-validated likelihood is to be optimized.
`base1`, `base2`	An optional vector of length equal to the number of covariates in penalized. If supplied, profiling or optimization is performed between `base1*minlambda1` and `base1*maxlambda1`; analogous for `base2`.
`fusedl`	If `TRUE` or a vector, the penalization method used is fused lasso. The value for `lambda1` is used as the tuning parameter for L1 penalization on the coefficients and the value for `lambda2` is used as the tuning parameter for L1 penalization on the differences of the coefficients. Default value is `FALSE`.
`positive`	If `TRUE`, constrains the estimated regression coefficients of all penalized covariates to be non-negative. If a logical vector with the length of the number of covariates in `penalized`, constrains the estimated regression coefficients of a subset of the penalized covariates to be non-negative.
`data`	A `data.frame` used to evaluate `response`, and the terms of `penalized` or `unpenalized` when these have been specified as a `formula` object.
`model`	The model to be used. If missing, the model will be guessed from the `response` input.
`startbeta`	Starting values for the regression coefficients of the penalized covariates. These starting values will be used only for the first values of `lambda1` and `lambda2`.
`startgamma`	Starting values for the regression coefficients of the unpenalized covariates. These starting values will be used only for the first values of `lambda1` and `lambda2`.
`fold`	The fold for cross-validation. May be supplied as a single number (between 2 and n) giving the number of folds, or, alternatively, as a length `n` vector with values in `1:fold`, specifying exactly which subjects are assigned to which fold. The default is `fold = 1:n`, resulting in leave-one-out (n-fold) cross-validation.
`epsilon`	The convergence criterion. As in `glm`. Convergence is judged separately on the likelihood and on the penalty.
`maxiter`	The maximum number of iterations allowed in each fitting of the model. Set by default at 25 when only an L2 penalty is present, infinite otherwise.

`standardize`	If `TRUE`, standardizes all penalized covariates to unit central L2-norm before applying penalization.
`steps`	The maximum number of steps between `minlambda1` and `maxlambda1` or `minlambda2` and `maxlambda2` at which the cross-validated likelihood is to be calculated.
`minsteps`	The minimum number of steps between `minlambda1` and `maxlambda1` or `minlambda2` and `maxlambda2` at which the cross-validated likelihood is to be calculated. If `minsteps` is smaller than `steps`, the algorithm will automatically stop when the cross-validated likelihood drops below the cross-validated likelihood of the null model, provided it has done at least `minsteps` steps.
`log`	If `FALSE`, the steps between `minlambda1` and `maxlambda1` or `minlambda2` and `maxlambda2` are equidistant on a linear scale, if `TRUE` on a logarithmic scale. Please note the different default between `optL1` (`FALSE`) and `optL2` (`TRUE`).
`tol`	The tolerance of the Brent algorithm used for minimization. See also `optimize`.
`save.predictions`	Controls whether or not to save cross-validated predictions for all values of lambda.
`trace`	If `TRUE`, prints progress information. Note that setting `trace=TRUE` may slow down the algorithm (but it often feels quicker)
`approximate`	If `TRUE`, the cross-validated likelihood values are approximated rather than fully calculated. Note that this option is only available for ridge models.
`plot`	If `TRUE`, makes a plot of cross-validated likelihood versus lambda.

Details

All five functions return a list with the following named elements:

`lambda`**:** For `optL1` and `optL2` `lambda` gives the optimal value of the tuning parameters found. For `profL1` and `profL2` `lambda` is the vector of values of the tuning parameter for which the cross-validated likelihood has been calculated. Absent in the output of `cvl`.

`cvl`**:** The value(s) of the cross-validated likelihood. For `optL1`, `optL2` this is the cross-validated likelihood at the optimal value of the tuning parameter.

`fold`**:** Returns the precise allocation of the subjects into the cross-validation folds. Note that the same allocation is used for all cross-validated likelihood calculations in each call to `optL1`, `optL2`, `profL1`, `profL2`.

`predictions`**:** The cross-validated predictions for the left-out samples. The precise format of the cross-validated predictions depends on the type of generalized linear model (see `breslow` for survival models. The functions `profL1` and `profL2` return a list here (only if `save.predictions = TRUE`), whereas `optL1`, `optL2` return the predictions for the optimal value of the tuning parameter only.

`fullfit`**:** The fitted model on the full data. The functions `profL1` and `profL2` return a list of `penfit` objects here, whereas `optL1`, `optL2` return the full data fit (a single `penfit` object) for the optimal value of the tuning parameter only.

Value

A named list. See details.

Note

The `optL1` and `optL2` functions use Brent's algorithm for minimization without derivatives (see also `optimize`). There is a risk that these functions converge to a local instead of to a global optimum. This is especially the case for `optL1`, as the cross-validated likelihood as a function of `lambda1` quite often has local optima. It is recommended to use `optL1` in combination with `profL1` to check whether `optL1` has converged to the right optimum.

See also the notes under `penalized`.

Author(s)

Jelle Goeman: `<j.j.goeman@lumc.nl>`

References

Goeman J.J. (2010). L-1 Penalized Estimation in the Cox Proportional Hazards Model. Biometrical Journal 52 (1) 70-84.

See Also

`penalized`, `penfit`, `plotpath`.

Examples

```
# More examples in the package vignette:
#  type vignette("penalized")

data(nki70)
attach(nki70)

# Finding an optimal cross-validated likelihood
opt <- optL1(Surv(time, event), penalized = nki70[,8:77], fold = 5)
coefficients(opt$fullfit)
plot(opt$predictions)

# Plotting the profile of the cross-validated likelihood
prof <- profL1(Surv(time, event), penalized = nki70[,8:77],
    fold = opt$fold, steps=10)
plot(prof$lambda, prof$cvl, type="l")
plotpath(prof$fullfit)
```

`Netherlands Cancer Institute 70 gene signature`

The 70 gene signature of the Netherlands Cancer Institute for prediction of metastasis-free survival, measured on 144 independent lymph node positive patients.

Description

A `data.frame` of 144 lymph node positive breast cancer patients on metastasis-free survival, 5 clinical risk factors, and gene expression measurements of 70 genes found to be prognostic for metastasis-free survival in an earlier study. The included variables are

time Metastasis-free follow-up time (months).

event Event indicator. 1 = metastasis or death; 0 = censored.

Diam Diameter of the tumor (two levels).

N Number of affected lymph nodes (two levels).

ER Estrogen receptor status (two levels).

Grade Grade of the tumor (three ordered levels).

Age Patient age at diagnosis (years).

TSPYL5...C20orf46 Gene expression measurements of 70 prognostic genes.

Usage

```
data(nki70)
```

Format

A `data.frame`.

Note

The 70 gene signature was trained on lymph node negative patients only.

References

M.J. van de Vijver, Y.D. He, L.J. van 't Veer, H. Dai, A.A.M. Hart, D.W. Voskuil, G.J. Schreiber, J.L. Peterse, C. Roberts, M.J. Marton, M. Parrish, D. Atsma, A. Witteveen, A. Glas, L. Delahaye, T. van der Velde, H. Bartelink, S. Rodenhuis, E.T. Rutgers, S.H. Friend, and R. Bernards (2002). A gene-expression signature as a predictor of survival in breast cancer. New England Journal of Medicine 347 (25), 1999–2009.

`Penalized generalized linear models`

Penalized regression

Description

Fitting generalized linear models with L1 (lasso and fused lasso) and/or L2 (ridge) penalties, or a combination of the two.

Usage

```
penalized (response, penalized, unpenalized, lambda1=0,
  lambda2=0, positive = FALSE, data, fusedl=FALSE,
  model = c("cox", "logistic", "linear", "poisson"),
  startbeta, startgamma, steps =1, epsilon = 1e-10,
  maxiter, standardize = FALSE, trace = TRUE)
```

Arguments

`response`	The response variable (vector). This should be a numeric vector for linear regression, a `Surv` object for Cox regression and `factor` or a vector of 0/1 values for logistic regression.
`penalized`	The penalized covariates. These may be specified either as a matrix or as a (one-sided) `formula` object. See also under `data`.
`unpenalized`	Additional unpenalized covariates. Specified as under `penalized`. Note that an unpenalized intercept is included in the model by default (except in the Cox model). This can be suppressed by specifying `unpenalized = ~0`.
`lambda1, lambda2`	The tuning parameters for L1 and L2 penalization. Each must be either a single positive numbers or a vector with length equal to the number of covariates in `penalized` argument. In the latter case, each covariate is given its own penalty weight.
`positive`	If `TRUE`, constrains the estimated regression coefficients of all penalized covariates to be non-negative. If a logical vector with the length of the number of covariates in `penalized`, constrains the estimated regression coefficients of a subset of the penalized covariates to be non-negative.
`data`	A `data.frame` used to evaluate `response`, and the terms of `penalized` or `unpenalized` when these have been specified as a `formula` object.
`fusedl`	If `TRUE` or a vector, the penalization method used is fused lasso. The value for `lambda1` is used as the tuning parameter for L1 penalization on the coefficients and the value for `lambda2` is used as the tuning parameter for L1 penalization on the differences of the coefficients. Default value is `FALSE`.
`model`	The model to be used. If missing, the model will be guessed from the `response` input.
`startbeta`	Starting values for the regression coefficients of the penalized covariates.
`startgamma`	Starting values for the regression coefficients of the unpenalized covariates.
`steps`	If greater than 1, the algorithm will fit the model for a range of `steps` `lambda1`-values, starting from the maximal value down to the value of `lambda1` specified. This is useful for making plots as in `plotpath`. With `steps = "Park"` it is possible to choose the steps in such a way that they are at the approximate value at which the active set changes, following Park and Haste (2007).
`epsilon`	The convergence criterion. As in `glm`. Convergence is judged separately on the likelihood and on the penalty.

`maxiter`	The maximum number of iterations allowed. Set by default at 25 when only an L2 penalty is present, infinite otherwise.
`standardize`	If `TRUE`, standardizes all penalized covariates to unit central L2-norm before applying penalization.
`trace`	If `TRUE`, prints progress information. Note that setting `trace=TRUE` may slow down the algorithm up to 30 percent (but it often feels quicker)

Details

The `penalized` function fits regression models for a given combination of L1 and L2 penalty parameters.

Value

`penalized` returns a `penfit` object when `steps = 1` or a list of such objects if `steps > 1`.

Note

The `response` argument of the function also accepts formula input as in `lm` and related functions. In that case, the right hand side of the `response` formula is used as the `penalized` argument or, if that is already given, as the `unpenalized` argument. For example, the input `penalized(y~x)` is equivalent to `penalized(y, ~x)` and `penalized(y~x, ~z)` to `penalized(y, ~z, ~x)`.

In case of tied survival times, the function uses Breslow's version of the partial likelihood.

Author(s)

Jelle Goeman: `<j.j.goeman@lumc.nl>`

References

Goeman J.J. (2010). L-1 Penalized Estimation in the Cox Proportional Hazards Model. Biometrical Journal 52 (1) 70-84.

See Also

`penfit` for the `penfit` object returned, `plotpath` for plotting the solution path, and `cvl` for cross-validation and optimizing the tuning parameters.

Examples

```
# More examples in the package vignette:
#  type vignette("penalized")

data(nki70)

# A single lasso fit predicting survival
pen <- penalized(Surv(time, event), penalized = nki70[,8:77],
    unpenalized = ~ER+Age+Diam+N+Grade, data = nki70, lambda1 = 10)
show(pen)
coefficients(pen)
```

```
coefficients(pen, "penalized")
basehaz(pen)

# A single lasso fit using the clinical risk factors
pen <- penalized(Surv(time, event), penalized = ~ER+Age+Diam+N+Grade,
    data = nki70, lambda1=10, standardize=TRUE)

# using steps
pen <- penalized(Surv(time, event), penalized = nki70[,8:77],
    data = nki70, lambda1 = 1,steps = 20)
plotpath(pen)


# A fused lasso fit predicting survival
pen <- penalized(Surv(time, event), penalized = nki70[,8:77], data = nki70,
     lambda1 = 1, lambda2 = 2, fusedl = TRUE)
plot(coefficients(pen, "all"),type="l",xlab = "probes",ylab = "coefficient value")
plot(predict(pen,penalized=nki70[,8:77]))
```

`Penalized regression contrasts`

Contrast functions for penalized regression

Description

Contrast functions for factors that are appropriate for penalized regression.

Usage

```
contr.none(n, contrasts)

contr.diff(n, contrasts = TRUE)
```

Arguments

`n`	A vector of levels for a factor, or the number of levels.
`contrasts`	A logical indicating whether contrasts should be computed. This argument is ignored in `contr.none`.

Details

These functions are used for creating contrast matrices for use in fitting penalized analysis of variance and regression models. The columns of the resulting matrices contain contrasts which can be used for coding a factor with n levels. The returned value contains the computed contrasts.

`contr.none` returns an identity matrix except when the number of levels is 2, in which case it returns a single contrast. `contr.none` ensures that all levels of an unordered factor are treated symmetrically in a penalized regression model.

`contr.diff` returns a lower triangular matrix of ones if `contrasts=FALSE` and the same matrix without its first column if `contrasts=TRUE`. This makes sure that penalization is done on the difference between successive levels of an ordered factor. It is not appropriate for unordered factors.

Value

`contr.diff` returns a matrix with `n` rows and `k` columns, with `k=n-1` if contrasts is `TRUE` and `k=n` if contrasts is `FALSE`.

`contr.none` returns a matrix with `n` rows and `n` columns, except when `n=2` when it returns a matrix with 2 rows and one column.

Author(s)

Jelle Goeman: <`j.j.goeman@lumc.nl`>

See Also

`penalized`, `contr.treatment`, `contr.poly`, `contr.helmert`, `contr.SAS`, `contr.sum`.

Examples

```
# Three levels
levels <- LETTERS[1:3]
contr.none(levels)
contr.diff(levels)

# Two levels
levels <- LETTERS[1:2]
contr.none(levels)
```

`penfit object` *Penalized regression object*

Description

Stores the result of a call to `penalized`.

Slots

- `penalized`**:** Object of class "vector". Regression coefficients for the penalized covariates.
- `unpenalized`**:** Object of class "vector". Regression coefficients for the unpenalized covariates.
- `residuals`**:** Object of class "vector". Unstandardized residuals for each subject in the fitted model. Martingale residuals are given for the cox model.
- `fitted`**:** Object of class "vector". Fitted values (means) for each subject in the fitted model. In the cox model, this slot holds the relative risks.
- `lin.pred`**:** Object of class "vector". Linear predictors for each subject in the fitted model.

`loglik`**:** Object of class "numeric". Log likelihood of the fitted model. For the Cox model, reports the full likelihood rather than the partial likelihood.

`penalty`**:** Object of class "vector". L1 and L2 penalties of the fitted model.

`iterations`**:** Object of class "numeric". Number of iterations used in the fitting process.

`converged`**:** Object of class "logical". Whether the fitting process was judged to be converged.

`model`**:** Object of class "character". The name of the generalized linear model used.

`lambda1`**:** Object of class "vector". The lambda1 parameter(s) used.

`lambda2`**:** Object of class "vector". The lambda2 parameter(s) used.

`nuisance`**:** Object of class "list". The maximum likelihood estimates of any nuisance parameters in the model.

`weights`**:** Object of class "vector". The weights of the covariates used for standardization.

`formula`**:** Object of class "list". A named list containing the unpenalized and penalized formula objects, if present.

Methods

basehaz (penfit): Returns the baseline hazard (a `data.frame`) if a cox model was fitted, `NULL` otherwise. An additional argument `center` (default (`TRUE`) can be used to give the survival curve at the covariate mean (`center = TRUE`) rather than at zero.

basesurv (penfit): Returns the baseline survival curve (a `breslow` object) if a cox model was fitted, `NULL` otherwise. An additional argument `center` (default (`TRUE`) can be used to give the survival curve at the covariate mean (`center = TRUE`) rather than at zero.

coef (penfit): Returns the regression coefficients. Accepts a second argument "which", that takes values "nonzero" (the default), "all", "penalized" or "unpenalized" for extracting only the nonzero, the penalized or the unpenalized regression coefficients. A third argument "standardize" (default FALSE) can be used to let the method return the regression coefficients for the standardized covariates.

coefficients (penfit): synonym for `coef` above.

fitted (penfit): Returns the fitted values for each subject (i.e. the predicted means). In the Cox model, this method returns the relative risks for each individual.

fitted.values (penfit): synonym for `fitted` above.

linear.predictors (penfit): Returns the linear predictors for each subject.

loglik (penfit): Returns the log likelihood of the fitted model.

penalty (penfit): Returns the L1 and L2 penalties of the fitted model.

residuals (penfit): Returns the residuals.

show (penfit): Summarizes the fitted model.

weights (penfit): Returns the weights used for standardization.

predict (penfit): Calculates predictions for new subjects. See `predict`.

Author(s)

Jelle Goeman: `<j.j.goeman@lumc.nl>`

See Also

`penalized`.

`Plotting the LASSO path`

Plotting the LASSO path

Description

Plotting a LASSO path fitted using `penalized` with `steps` > `1`.

Usage

```
plotpath(object, labelsize = 0.6, standardize = FALSE, ...)
```

Arguments

`object`	A list of `link{penfit}` objects calculated for the same data but different `lambda1` values. This object can be created using the `link{penalized}` function using the `steps` argument.
`labelsize`	Sets the size of the variable labels in the plot. Set to zero for no variable labels.
`standardize`	If `TRUE`, plots the regression coefficients for the covariates standardize to unit second central moment. See `penalized` for details on standardization. Note that the standardization in `plotpath` can also be used if standarization was not used in `penalized`.
`...`	Any other arguments will be forwarded to the plot function.

Author(s)

Jelle Goeman: `<j.j.goeman@lumc.nl>`

See Also

`penalized`, `penfit`.

Examples

```
data(nki70)

# Fit the model with the steps argument and plot
pen <- penalized(Surv(time, event), penalized = nki70[,8:77],
    data = nki70, lambda1=1, steps = 20)

plotpath(pen)
```

`Prediction from penalized models` *Prediction based on penfit objects*

Description

Predicting a response for new subjects based on a fitted penalized regression model.

Usage

```
## S4 method for signature 'penfit'
predict(object, penalized, unpenalized, data)
```

Arguments

`object`	The fitted model (a `penfit` object).
`penalized`	The matrix of penalized covariates for the new subjects.
`unpenalized`	The unpenalized covariates for the new subjects.
`data`	A `data.frame` used to evaluate the terms of `penalized` or `unpenalized` when these have been specified as a `formula` object.

Details

The user need only supply those terms from the original call that are different relative to the original call that produced the `penfit` object. In particular, if penalized and/or unpenalized was specified in matrix form, a matrix must be given with the new subjects' data. The columns of these matrices must be exactly the same as in the matrices supplied in the original call that produced the `penfit` object. If either penalized or unpenalized was given as a `formula` in the original call, the user of `predict` must supply a new `data` argument. As with matrices, the new `data` argument must have a similar make-up as the `data` argument in the original call that produced the `penfit` object. In particular, any factors in `data` must have the same levels.

Value

The predictions, either as a `vector` (logistic and Poisson models), a `matrix` (linear model), or a `breslow` object (Cox model).

Examples

```
data(nki70)

pen <- penalized(Surv(time, event), penalized = nki70[1:50,8:77],
    unpenalized = ~ER+Age+Diam+N+Grade, data = nki70[1:50,], lambda1 = 10)

predict(pen, nki70[51:52,8:77], data = nki70[51:52,])
```

Index