Package ‘hdqr’

February 12, 2025

Type Package

Title Fast Algorithm for Penalized Quantile Regression

Version 1.0.1

Date 2025-02-05

Maintainer Qian Tang <qian-tang@uiowa.edu>

Description Implements an efficient algorithm to fit and tune penalized quantile regression models using the generalized coordinate descent algorithm. Designed to handle high-dimensional datasets effectively, with emphasis on precision and computational efficiency. This package implements the algorithms proposed in Tang, Q., Zhang, Y., & Wang, B. (2022) <https://openreview.net/pdf?id=RvwMTDYTOb>.

License GPL-2

Encoding UTF-8

Depends R (>= 3.5.0)

Imports stats, Matrix, methods

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation yes

RoxygenNote 7.2.3

Author Qian Tang [aut, cre],
Yikai Zhang [aut],
Boxiang Wang [aut]

Repository CRAN

Date/Publication 2025-02-12 18:20:02 UTC

Contents

`coef.cv.hdqr` *Extract Coefficients from a ‘cv.hdqr‘ Object*

Description

Retrieves coefficients from a cross-validated ‘hdqr()‘ model, using the stored ‘"hdqr.fit"‘ object and the optimal ‘lambda‘ value determined during cross-validation.

Usage

```
## S3 method for class 'cv.hdqr'
coef(object, s = c("lambda.1se", "lambda.min"), ...)
```

Arguments

`object`	A fitted ‘cv.hdqr()‘ object from which coefficients are to be extracted.
`s`	Specifies the value(s) of the penalty parameter ‘lambda‘ for which coefficients are desired. The default is ‘s = "lambda.1se"‘, which corresponds to the largest value of ‘lambda‘ such that the cross-validation error estimate is within one standard error of the minimum. Alternatively, ‘s = "lambda.min"‘ can be used, corresponding to the minimum of the cross-validation error estimate. If ‘s‘ is numeric, these are taken as the actual values of ‘lambda‘ to use.
`...`	Not used.

Value

Returns the coefficients at the specified ‘lambda‘ values.

See Also

`cv.hdqr`, `predict.cv.hdqr`

Examples

```
set.seed(315)
n <- 100
p <- 400
x <- matrix(data = rnorm(n * p, mean = 0, sd = 1), nrow = n, ncol = p)
beta_star <- c(c(2, 1.5, 0.8, 1, 1.75, 0.75, 0.3), rep(0, (p - 7)))
eps <- rnorm(n, mean = 0, sd = 1)
y <- x %*% beta_star + eps
tau <- 0.5
lam2 <- 0.01
cv.fit <- cv.hdqr(x = x, y = y, tau = tau, lam2 = lam2)
coef(cv.fit, s = c(0.02, 0.03))
```

`coef.cv.nc.hdqr` *Extract Coefficients from a ‘cv.nc.hdqr‘ Object*

Description

Retrieves coefficients at specified values of ‘lambda‘ from a fitted ‘cv.nc.hdqr()‘ model. Utilizes the stored ‘"nchdqr.fit"‘ object and the optimal ‘lambda‘ values determined during the cross-validation process.

Usage

```
## S3 method for class 'cv.nc.hdqr'
coef(object, s = c("lambda.1se", "lambda.min"), ...)
```

Arguments

`object`	A fitted ‘cv.nc.hdqr()‘ object from which coefficients are to be extracted.
`s`	Specifies the ‘lambda‘ values at which coefficients are requested. The default is ‘s = "lambda.1se"‘, representing the largest ‘lambda‘ such that the cross-validation error estimate is within one standard error of the minimum. Alternatively, ‘s = "lambda.min"‘ corresponds to the ‘lambda‘ yielding the minimum cross-validation error. If ‘s‘ is numeric, these values are directly used as the ‘lambda‘ values for coefficient extraction.
`...`	Not used.

Value

Returns a vector or matrix of coefficients corresponding to the specified ‘lambda‘ values.

See Also

`cv.nc.hdqr`, `predict.cv.nc.hdqr`

Examples

```
set.seed(315)
n <- 100
p <- 400
x <- matrix(data = rnorm(n * p, mean = 0, sd = 1), nrow = n, ncol = p)
beta_star <- c(c(2, 1.5, 0.8, 1, 1.75, 0.75, 0.3), rep(0, (p - 7)))
eps <- rnorm(n, mean = 0, sd = 1)
y <- x %*% beta_star + eps
tau <- 0.5
lam2 <- 0.01
lambda <- 10^(seq(1,-4, length.out=30))
cv.nc.fit <- cv.nc.hdqr(x = x, y = y, tau = tau, lambda = lambda, lam2 = lam2)
coef(cv.nc.fit, s = c(0.02, 0.03))
```

---

coef.hdqr *Extract Model Coefficients from a ‘hdqr‘ Object*

Description

Retrieves the coefficients at specified values of ‘lambda‘ from a fitted ‘hdqr()‘ model.

Usage

```
## S3 method for class 'hdqr'
coef(object, s = NULL, type = c("coefficients", "nonzero"), ...)
```

Arguments

object	Fitted ‘hdqr()‘ object.
s	Values of the penalty parameter ‘lambda‘ for which coefficients are requested. Defaults to the entire sequence used during the model fit.
type	Type of prediction required. Type ‘"coefficients"‘ computes the coefficients at the requested values for ‘s‘. Type ‘"nonzero"‘ returns a list of the indices of the nonzero coefficients for each value of s.
...	Not used.

Details

This function extracts coefficients for specified ‘lambda‘ values from a ‘hdqr()‘ object. If ‘s‘, the vector of ‘lambda‘ values, contains values not originally used in the model fitting, the ‘coef‘ function employs linear interpolation between the closest ‘lambda‘ values from the original sequence to estimate coefficients at the new ‘lambda‘ values.

Value

Returns a matrix or vector of coefficients corresponding to the specified ‘lambda‘ values.

See Also

hdqr, predict.hdqr

Examples

```
set.seed(315)
n <- 100
p <- 400
x <- matrix(data = rnorm(n * p, mean = 0, sd = 1), nrow = n, ncol = p)
beta_star <- c(c(2, 1.5, 0.8, 1, 1.75, 0.75, 0.3), rep(0, (p - 7)))
eps <- rnorm(n, mean = 0, sd = 1)
y <- x %*% beta_star + eps
tau <- 0.5
lam2 <- 0.01
fit <- hdqr(x = x, y = y, tau = tau, lam2 = lam2)
coefs <- coef(fit, s = fit$lambda[3:5])
```

---

coef.nc.hdqr *Extract Model Coefficients from a ‘nc.hdqr‘ Object*

Description

Retrieves the coefficients at specified values of ‘lambda‘ from a fitted ‘nc.hdqr()‘ model.

Usage

```
## S3 method for class 'nc.hdqr'
coef(object, s = NULL, type = c("coefficients", "nonzero"), ...)
```

Arguments

`object`	Fitted ‘nc.hdqr()‘ object.
`s`	Values of the penalty parameter ‘lambda‘ for which coefficients are requested. Defaults to the entire sequence used during the model fit.
`type`	Type of prediction required. Type ‘"coefficients"‘ computes the coefficients at the requested values for ‘s‘. Type ‘"nonzero"‘ returns a list of the indices of the nonzero coefficients for each value of `s`.
`...`	Not used.

Details

This function extracts coefficients for specified ‘lambda‘ values from a ‘nc.hdqr()‘ object. If ‘s‘, the vector of ‘lambda‘ values, contains values not originally used in the model fitting, the ‘coef‘ function employs linear interpolation between the closest ‘lambda‘ values from the original sequence to estimate coefficients at the new ‘lambda‘ values.

Value

Returns a matrix or vector of coefficients corresponding to the specified ‘lambda‘ values.

See Also

`nc.hdqr`, `predict.nc.hdqr`

Examples

```
set.seed(315)
n <- 100
p <- 400
x <- matrix(data = rnorm(n * p, mean = 0, sd = 1), nrow = n, ncol = p)
beta_star <- c(c(2, 1.5, 0.8, 1, 1.75, 0.75, 0.3), rep(0, (p - 7)))
eps <- rnorm(n, mean = 0, sd = 1)
y <- x %*% beta_star + eps
tau <- 0.5
lam2 <- 0.01
lambda <- 10^(seq(1,-4, length.out=30))
nc.fit <- nc.hdqr(x=x, y=y, tau=tau, lambda=lambda, lam2=lam2, pen="scad")
nc.coefs <- coef(nc.fit, s = nc.fit$lambda[3:5])
```

`cv.hdqr` *Cross-validation for Selecting the Tuning Parameter in Penalized Quantile Regression*

Description

Performs k-fold cross-validation for `hdqr`.

Usage

```
cv.hdqr(x, y, lambda = NULL, tau, nfolds = 5L, foldid, ...)
```

Arguments

`x`	A numerical matrix with n rows (observations) and p columns (variables).
`y`	Response variable.
`lambda`	Optional; a user-supplied sequence of `lambda` values. If `NULL`, `hdqr` selects its own sequence.
`tau`	Quantile level (`tau`) used in the loss function.
`nfolds`	Number of folds for cross-validation. Defaults to 5.
`foldid`	Optional vector specifying the indices of observations in each fold. If provided, it overrides `nfolds`.
`...`	Additional arguments passed to `hdqr`.

Details

This function computes the average cross-validation error and provides the standard error.

Value

An object with S3 class `cv.hdqr` consisting of

`lambda`	Candidate `lambda` values.
`cvm`	Mean cross-validation error.
`cvsd`	Standard error of the mean cross-validation error.
`cvup`	Upper confidence curve: `cvm` + `cvsd`.
`cvlo`	Lower confidence curve: `cvm` - `cvsd`.
`lambda.min`	`lambda` achieving the minimum cross-validation error.
`lambda.1se`	Largest `lambda` within one standard error of the minimum error.
`cv.min`	Cross-validation error at `lambda.min`.
`cv.1se`	Cross-validation error at `lambda.1se`.
`hdqr.fit`	a fitted `hdqr` object for the full data.
`nzero`	Number of non-zero coefficients at each `lambda`.

Examples

```
set.seed(315)
n <- 100
p <- 400
x <- matrix(data = rnorm(n * p, mean = 0, sd = 1), nrow = n, ncol = p)
beta_star <- c(c(2, 1.5, 0.8, 1, 1.75, 0.75, 0.3), rep(0, (p - 7)))
eps <- rnorm(n, mean = 0, sd = 1)
y <- x %*% beta_star + eps
tau <- 0.5
cv.fit <- cv.hdqr(x = x, y = y, tau = tau)
```

`cv.nc.hdqr` *Cross-validation for Selecting the Tuning Parameter of Nonconvex Penalized Quantile Regression*

Description

Conducts k-fold cross-validation for the ‘nc.hdqr()‘ function.

Usage

```
cv.nc.hdqr(x, y, lambda = NULL, tau, nfolds = 5L, foldid, ...)
```

Arguments

`x`	A numerical matrix with dimensions (n rows and p columns), where each row represents an observation.
`y`	Response variable.
`lambda`	Optional user-supplied sequence of `lambda` values.
`tau`	The quantile level (`tau`) used in the error calculation. Default value is typically 0.5 unless specified.
`nfolds`	Number of folds in the cross-validation, default is 5.
`foldid`	An optional vector that assigns each observation to a specific fold. If provided, this parameter overrides `nfolds`.
`...`	Additional arguments passed to `nc.hdqr`.

Details

This function estimates the average cross-validation error and its standard error across folds. It is primarily used to identify the optimal `lambda` value for fitting nonconvex penalized quantile regression models.

Value

An object of class `cv.nc.hdqr` is returned, which is a list with the ingredients of the cross-validated fit.

`lambda`	the values of `lambda` used in the fits.
`cvm`	the mean cross-validated error - a vector of length `length(lambda)`.
`cvsd`	estimate of standard error of `cvm`.
`cvupper`	upper curve = `cvm+cvsd`.
`cvlower`	lower curve = `cvm-cvsd`.
`nzero`	number of non-zero coefficients at each `lambda`.
`name`	a text string indicating type of measure (for plotting purposes).
`nchdqr.fit`	a fitted `nc.hdqr` object for the full data.
`lambda.min`	The optimal value of `lambda` that gives minimum cross validation error `cvm`.
`lambda.1se`	The largest value of `lambda` such that error is within 1 standard error of the minimum.

Examples

```
set.seed(315)
n <- 100
p <- 400
x <- matrix(data = rnorm(n * p, mean = 0, sd = 1), nrow = n, ncol = p)
beta_star <- c(c(2, 1.5, 0.8, 1, 1.75, 0.75, 0.3), rep(0, (p - 7)))
eps <- rnorm(n, mean = 0, sd = 1)
y <- x %*% beta_star + eps
tau <- 0.5
```

```
lam2 <- 0.01
lambda <- 10^(seq(1,-4, length.out=10))
cv.nc.fit <- cv.nc.hdqr(y=y, x=x, tau=tau, lambda=lambda, lam2=lam2, pen="scad")
```

hdqr — *Solve the linear quantile regression. The solution path is computed at a grid of values of tuning parameter* `lambda`.

Description

Solve the linear quantile regression. The solution path is computed at a grid of values of tuning parameter `lambda`.

Usage

```
hdqr(
  x,
  y,
  tau,
  nlambda = 100,
  lambda.factor = ifelse(nobs < nvars, 0.01, 1e-04),
  lambda = NULL,
  lam2 = 0.01,
  hval = 0.125,
  pf = rep(1, nvars),
  pf2 = rep(1, nvars),
  exclude,
  dfmax = nvars + 1,
  pmax = min(dfmax * 1.2, nvars),
  standardize = TRUE,
  eps = 1e-08,
  maxit = 1e+06,
  sigma = 0.05,
  is_exact = FALSE
)
```

Arguments

`x`	Matrix of predictors, of dimension (nobs * nvars); each row is an observation.
`y`	Response variable. The length is n.
`tau`	The quantile level τ. The value must be in (0,1). Default is 0.5.
`nlambda`	The number of `lambda` values (default is 100).
`lambda.factor`	The factor for getting the minimal value in the `lambda` sequence, where `min(lambda) = lambda.factor * max(lambda)` and `max(lambda)` is the smallest value of `lambda` for which all coefficients (except the intercept when it is present) are

	penalized to zero. The default depends on the relationship between n (the number of rows in the design matrix) and p (the number of predictors). If $n < p$, it defaults to `0.05`. If $n > p$, the default is `0.001`, closer to zero. A very small value of `lambda.factor` will lead to a saturated fit. The argument takes no effect if there is a user-supplied `lambda` sequence.
`lambda`	A user-supplied `lambda` sequence. Typically, by leaving this option unspecified, users can have the program compute its own `lambda` sequence based on `nlambda` and `lambda.factor`. It is better to supply, if necessary, a decreasing sequence of `lambda` values than a single (small) value. The program will ensure that the user-supplied `lambda` sequence is sorted in decreasing order before fitting the model to take advanage of the warm-start technique.
`lam2`	Regularization parameter `lambda2` for the quadratic penalty of the coefficients. Unlike `lambda`, only one value of `lambda2` is used for each fitting process.
`hval`	The smoothing index for `method='huber'`. Default is 0.125.
`pf`	L1 penalty factor of length p used for the adaptive LASSO or adaptive elastic net. Separate L1 penalty weights can be applied to each coefficient to allow different L1 shrinkage. Can be 0 for some variables (but not all), which imposes no shrinkage, and results in that variable always being included in the model. Default is 1 for all variables (and implicitly infinity for variables in the `exclude` list).
`pf2`	L2 penalty factor of length p used for adaptive elastic net. Separate L2 penalty weights can be applied to each coefficient to allow different L2 shrinkage. Can be 0 for some variables, which imposes no shrinkage. Default is 1 for all variables.
`exclude`	Indices of variables to be excluded from the model. Default is none. Equivalent to an infinite penalty factor.
`dfmax`	The maximum number of variables allowed in the model. Useful for very large p when a partial path is desired. Default is $p + 1$.
`pmax`	The maximum number of coefficients allowed ever to be nonzero along the solution path. For example, once β enters the model, no matter how many times it exits or re-enters the model through the path, it will be counted only once. Default is `min(dfmax*1.2, p)`.
`standardize`	Logical flag for variable standardization, prior to fitting the model sequence. The coefficients are always returned to the original scale. Default is `TRUE`.
`eps`	Stopping criterion.
`maxit`	Maximum number of iterates.
`sigma`	Penalty parameter appearing in the quadratic term of the augmented Lagrangian function. Must be positive.
`is_exact`	Exact or approximated solutions. Default is `FALSE`.

Details

Note that the objective function in the penalized quantile regression is

$$1'\rho_\tau(y - X\beta - b_0))/N + \lambda_1 \cdot |pf_1 \circ \beta|_1 + 0.5 * \lambda_2 \cdot |\sqrt{pf_2} \circ \beta|^2,$$

where ρ_τ the quantile or check loss and the penalty is a combination of weighted L1 and L2 terms and $\circ$ denotes the Hadmamard product.

For faster computation, if the algorithm is not converging or running slow, consider increasing `eps`, increasing `sigma`, decreasing `nlambda`, or increasing `lambda.factor` before increasing `maxit`.

Value

An object with S3 class `hdqr` consisting of

`call`	the call that produced this object
`b0`	intercept sequence of length `length(lambda)`
`beta`	a `p*length(lambda)` matrix of coefficients, stored as a sparse matrix (`dgCMatrix` class, the standard class for sparse numeric matrices in the `Matrix` package.). To convert it into normal type matrix, use `as.matrix()`.
`lambda`	the actual sequence of `lambda` values used
`df`	the number of nonzero coefficients for each value of `lambda`.
`npasses`	the number of iterations for every lambda value
`jerr`	error flag, for warnings and errors, 0 if no error.

Examples

```
set.seed(315)
n <- 100
p <- 400
x <- matrix(data = rnorm(n * p, mean = 0, sd = 1), nrow = n, ncol = p)
beta_star <- c(c(2, 1.5, 0.8, 1, 1.75, 0.75, 0.3), rep(0, (p - 7)))
eps <- rnorm(n, mean = 0, sd = 1)
y <- x %*% beta_star + eps
tau <- 0.5
lam2 <- 0.01
fit <- hdqr(x = x, y = y, tau = tau, lam2 = lam2)
```

`nc.hdqr` *Solve the Penalized Quantile Regression with Nonconvex Penalties*

Description

This function fits the penalized quantile regression model using nonconvex penalties such as SCAD or MCP. It allows for flexible control over the regularization parameters and offers advanced options for initializing and optimizing the fit.

Usage

```
nc.hdqr(
  x,
  y,
  tau,
  lambda,
  pen = "scad",
  aval = NULL,
  lam2 = 1,
  ini_beta = NULL,
  lla_step = 3,
  ...
)
```

Arguments

`x`	Matrix of predictors, with dimensions (nobs * nvars); each row represents an observation.
`y`	Response variable, with length n.
`tau`	The quantile level τ, which must be in the range (0,1). Default is 0.5.
`lambda`	Optional user-supplied sequence of `lambda` values. If unspecified, the program calculates its own sequence based on `nlambda` and `lambda.factor`. Supplying a decreasing sequence of `lambda` values is advisable to leverage the warm-start optimization.
`pen`	Specifies the type of nonconvex penalty: "SCAD" or "MCP".
`aval`	The parameter value for the SCAD or MCP penalty. Default is 3.7 for SCAD and 2 for MCP.
`lam2`	Regularization parameter `lambda2` for the quadratic penalty on the coefficients. Only one value of `lambda2` is used per fit.
`ini_beta`	Optional initial coefficients to start the fitting process.
`lla_step`	Number of Local Linear Approximation (LLA) steps. Default is 3.
`...`	Additional arguments passed to `hdqr`.

Value

An object with S3 class `nc.hdqr` consisting of

`call`	the call that produced this object
`b0`	intercept sequence of length `length(lambda)`
`beta`	a `p*length(lambda)` matrix of coefficients, stored as a sparse matrix (`dgCMatrix` class, the standard class for sparse numeric matrices in the `Matrix` package.). To convert it into normal type matrix, use `as.matrix()`.
`lambda`	the actual sequence of `lambda` values used
`df`	the number of nonzero coefficients for each value of `lambda`.
`npasses`	the number of iterations for every lambda value

`jerr`	error flag, for warnings and errors, 0 if no error.

#'

Examples

```
set.seed(315)
n <- 100
p <- 400
x <- matrix(data = rnorm(n * p, mean = 0, sd = 1), nrow = n, ncol = p)
beta_star <- c(c(2, 1.5, 0.8, 1, 1.75, 0.75, 0.3), rep(0, (p - 7)))
eps <- rnorm(n, mean = 0, sd = 1)
y <- x %*% beta_star + eps
tau <- 0.5
lam2 <- 0.01
lambda <- 10^(seq(1,-4, length.out=30))
nc.fit <- nc.hdqr(x=x, y=y, tau=tau, lambda=lambda, lam2=lam2, pen="scad")
```

`predict.cv.hdqr` *Make Predictions from a ‘cv.hdqr‘ Object*

Description

Generates predictions using a fitted ‘cv.hdqr()‘ object. This function utilizes the stored ‘hdqr.fit‘ object and an optimal value of ‘lambda‘ determined during the cross-validation process.

Usage

```
## S3 method for class 'cv.hdqr'
predict(object, newx, s = c("lambda.1se", "lambda.min"), ...)
```

Arguments

`object`	A fitted ‘cv.hdqr()‘ object from which predictions are to be made.
`newx`	Matrix of new predictor values for which predictions are desired. This must be a matrix and is a required argument.
`s`	Specifies the value(s) of the penalty parameter ‘lambda‘ at which predictions are desired. The default is ‘s = "lambda.1se"‘, representing the largest value of ‘lambda‘ such that the cross-validation error estimate is within one standard error of the minimum. Alternatively, ‘s = "lambda.min"‘ can be used, corresponding to the minimum of the cross-validation error estimate. If ‘s‘ is numeric, these are taken as the actual values of ‘lambda‘ to use for predictions.
`...`	Not used.

Value

Returns a matrix or vector of predicted values corresponding to the specified ‘lambda‘ values.

See Also

cv.hdqr, coef.cv.hdqr

Examples

```
set.seed(315)
n <- 100
p <- 400
x <- matrix(data = rnorm(n * p, mean = 0, sd = 1), nrow = n, ncol = p)
beta_star <- c(c(2, 1.5, 0.8, 1, 1.75, 0.75, 0.3), rep(0, (p - 7)))
eps <- rnorm(n, mean = 0, sd = 1)
y <- x %*% beta_star + eps
tau <- 0.5
lam2 <- 0.01
cv.fit <- cv.hdqr(x = x, y = y, tau = tau, lam2 = lam2)
predict(cv.fit, newx = x[50:60, ], s = "lambda.min")
```

---

predict.cv.nc.hdqr *Make Predictions from a ‘cv.nc.hdqr‘ Object*

Description

Generates predictions using a fitted ‘cv.nc.hdqr()‘ object. This function utilizes the stored ‘nchdqr.fit‘ object and an optimal value of ‘lambda‘ determined during the cross-validation process.

Usage

```
## S3 method for class 'cv.nc.hdqr'
predict(object, newx, s = c("lambda.1se", "lambda.min"), ...)
```

Arguments

object	A fitted ‘cv.nc.hdqr()‘ object from which predictions are to be made.
newx	Matrix of new predictor values for which predictions are desired. This must be a matrix and is a required argument.
s	Specifies the value(s) of the penalty parameter ‘lambda‘ at which predictions are desired. The default is ‘s = "lambda.1se"‘, representing the largest value of ‘lambda‘ such that the cross-validation error estimate is within one standard error of the minimum. Alternatively, ‘s = "lambda.min"‘ can be used, corresponding to the minimum of the cross-validation error estimate. If ‘s‘ is numeric, these are taken as the actual values of ‘lambda‘ to use for predictions.
...	Not used.

Value

Returns a matrix or vector of predicted values corresponding to the specified ‘lambda‘ values.

See Also

cv.nc.hdqr, predict.cv.nc.hdqr

Examples

```
set.seed(315)
n <- 100
p <- 400
x <- matrix(data = rnorm(n * p, mean = 0, sd = 1), nrow = n, ncol = p)
beta_star <- c(c(2, 1.5, 0.8, 1, 1.75, 0.75, 0.3), rep(0, (p - 7)))
eps <- rnorm(n, mean = 0, sd = 1)
y <- x %*% beta_star + eps
tau <- 0.5
lam2 <- 0.01
lambda <- 10^(seq(1,-4, length.out=10))
cv.nc.fit <- cv.nc.hdqr(x = x, y = y, tau = tau, lambda = lambda, lam2 = lam2)
predict(cv.nc.fit, newx = x[50:60, ], s = "lambda.min")
```

`predict.hdqr` *Make Predictions from a 'hdqr' Object*

Description

Produces fitted values for new predictor data using a fitted 'hdqr()' object.

Usage

```
## S3 method for class 'hdqr'
predict(object, newx, s = NULL, ...)
```

Arguments

`object`	Fitted 'hdqr()' object from which predictions are to be derived.
`newx`	Matrix of new predictor values for which predictions are desired. This must be a matrix and is a required argument.
`s`	Values of the penalty parameter 'lambda' for which predictions are requested. Defaults to the entire sequence used during the model fit.
`...`	Not used.

Details

This function generates predictions at specified 'lambda' values from a fitted 'hdqr()' object. It is essential to provide a new matrix of predictor values ('newx') at which these predictions are to be made.

Value

Returns a vector or matrix of predicted values corresponding to the specified 'lambda' values.

See Also

hdqr, coef.hdqr

Examples

```
set.seed(315)
n <- 100
p <- 400
x <- matrix(data = rnorm(n * p, mean = 0, sd = 1), nrow = n, ncol = p)
beta_star <- c(c(2, 1.5, 0.8, 1, 1.75, 0.75, 0.3), rep(0, (p - 7)))
eps <- rnorm(n, mean = 0, sd = 1)
y <- x %*% beta_star + eps
tau <- 0.5
lam2 <- 0.01
fit <- hdqr(x = x, y = y, tau = tau, lam2 = lam2)
preds <- predict(fit, newx = tail(x), s = fit$lambda[3:5])
```

---

predict.nc.hdqr *Make Predictions from a ‘nc.hdqr‘ Object*

Description

Produces fitted values for new predictor data using a fitted ‘nc.hdqr()‘ object.

Usage

```
## S3 method for class 'nc.hdqr'
predict(object, newx, s = NULL, ...)
```

Arguments

object	Fitted ‘nc.hdqr()‘ object from which predictions are to be derived.
newx	Matrix of new predictor values for which predictions are desired. This must be a matrix and is a required argument.
s	Values of the penalty parameter ‘lambda‘ for which predictions are requested. Defaults to the entire sequence used during the model fit.
...	Not used.

Details

This function generates predictions at specified ‘lambda‘ values from a fitted ‘nc.hdqr()‘ object. It is essential to provide a new matrix of predictor values (‘newx‘) at which these predictions are to be made.

Value

Returns a vector or matrix of predicted values corresponding to the specified ‘lambda‘ values.

See Also

nc.hdqr, coef.nc.hdqr

Examples

```
set.seed(315)
n <- 100
p <- 400
x <- matrix(data = rnorm(n * p, mean = 0, sd = 1), nrow = n, ncol = p)
beta_star <- c(c(2, 1.5, 0.8, 1, 1.75, 0.75, 0.3), rep(0, (p - 7)))
eps <- rnorm(n, mean = 0, sd = 1)
y <- x %*% beta_star + eps
tau <- 0.5
lam2 <- 0.01
lambda <- 10^(seq(1,-4, length.out=30))
nc.fit <- nc.hdqr(x=x, y=y, tau=tau, lambda=lambda, lam2=lam2, pen="scad")
nc.preds <- predict(nc.fit, newx = tail(x), s = nc.fit$lambda[3:5])
```

Index