Package: fastkqr (via r-universe)

October 24, 2026

Type Package

Title A Fast Algorithm for Kernel Quantile Regression

Version 1.0.1

Date 2026-06-30

Author Qian Tang [aut, cre], Yuwen Gu [aut], Boxiang Wang [aut]

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

Description Implements fast algorithms for kernel quantile regression and related models, including non-crossing kernel quantile regression and regularized linear quantile regression. The methods are described in Tang, Gu and Wang (2026) <doi:10.1080/10618600.2025.2541004>.

Depends R (>= 3.5.0)

Imports stats, parallel

License GPL-2

NeedsCompilation yes

Encoding UTF-8

Suggests knitr, rmarkdown

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

RoxygenNote 7.2.3

Repository https://cran.r-universe.dev

Date/Publication 2026-07-01 22:30:22 UTC

RemoteUrl https://github.com/cran/fastkqr

RemoteRef HEAD

RemoteSha f3fea14acafecaaeb94d8ffd5711fa7c33ef6c1b

Contents

coef.cv.fastlqr . . . 2
coef.fastlqr . . . 3
coef.kqr . . . 3
coef.nckqr . . . 4
cv.kqr . . . 5
cv.nckqr . . . 7
cv.qr . . . 8
kqr . . . 10
nckqr . . . 11
predict.cv.fastlqr . . . 13
predict.fastlqr . . . 13
predict.kqr . . . 14
predict.nckqr . . . 15
qr . . . 16

Index **18**

`coef.cv.fastlqr` *Extract Coefficients from a cv.fastlqr Object*

Description

Extract Coefficients from a cv.fastlqr Object

Usage

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

Arguments

`object`	A fitted `cv.fastlqr` object.
`s`	`"lambda.1se"`, `"lambda.min"`, or numeric lambda value(s).
`...`	Passed to `coef.fastlqr()`.

Value

A coefficient matrix.

coef.fastlqr *Extract Coefficients from a fastlqr Object*

Description

Extract Coefficients from a fastlqr Object

Usage

```
## S3 method for class 'fastlqr'
coef(object, s = NULL, ...)
```

Arguments

object	A fitted fastlqr object.
s	Optional lambda value(s). If omitted, all coefficients are returned.
...	Not used.

Value

A coefficient matrix with rows (Intercept) and predictors.

coef.kqr *Extract model coefficients from a* kqr *object.*

Description

Computes the coefficients at the requested value(s) for lambda from a kqr() object.

Usage

```
## S3 method for class 'kqr'
coef(object, s = NULL, ...)
```

Arguments

object	Fitted kqr() object.
s	Value(s) of the penalty parameter lambda at which coefficients are required. Default is the entire sequence.
...	Not used.

Details

s is the new vector of lambda values at which predictions are requested. If s is not in the lambda sequence used for fitting the model, the coef function will use linear interpolation to make predictions. The new values are interpolated using a fraction of coefficients from both left and right lambda indices.

Value

The coefficients at the requested values for lambda.

See Also

kqr() and predict.kqr().

Examples

```
set.seed(1)
x <- matrix(rnorm(60), 30, 2)
y <- x[, 1] + rnorm(30)
lambda <- 10^seq(0, -2, length.out = 3)
fit <- kqr(x, y, lambda = lambda, tau = 0.5)
coef(fit)
```

---

coef.nckqr *Extract model coefficients from a* nckqr *object.*

Description

Computes the coefficients at the requested value(s) for lambda1 for a given lambda2 from a nckqr() object.

Usage

```
## S3 method for class 'nckqr'
coef(object, s1 = NULL, s2, ...)
```

Arguments

object	A fitted nckqr object.
s1	Value(s) of the penalty parameter lambda1 at which coefficients are required. Default is the entire sequence used to create the model.
s2	Value of the penalty parameter lambda2 at which coefficients are required.
...	Not used.

Details

s1 is the new vector of lambda1 values at which predictions are requested. If s1 is not in the lambda sequence used for fitting the model, the coef function will use linear interpolation to make predictions. The new values are interpolated using a fraction of coefficients from both left and right lambda indices.

Value

The coefficients for the non-crossing kernel quantile regression model.

See Also

nckqr() and predict.nckqr().

Examples

```
set.seed(1)
x <- matrix(rnorm(60), 30, 2)
y <- x[, 1] + rnorm(30)
tau <- c(0.25, 0.5, 0.75)
lambda1 <- 1
lambda2 <- 10^seq(0, -2, length.out = 3)
fit <- nckqr(x, y, lambda1 = lambda1, lambda2 = lambda2, tau = tau)
coef(fit, s1 = lambda1, s2 = lambda2[1])
```

---

cv.kqr *cross-validation for selecting the tuning parameter of kernel quantile regression*

Description

Performs k-fold cross-validation for kqr().

Usage

```
cv.kqr(
  x,
  y,
  tau,
  lambda = NULL,
  sigma = NULL,
  kernel = "rbfdot",
  nfolds = 5L,
  foldid,
  parallel = FALSE,
  ncores = max(1L, parallel::detectCores() - 1L),
  Kfull = NULL,
  ...
)
```

Arguments

x	A numerical input matrix. The dimension is n rows and p columns.
y	Response variable.
tau	A user-supplied tau value for a quantile level.
lambda	A user-supplied lambda sequence.
sigma	Kernel bandwidth parameter used by "rbfdot" and "laplacedot".
kernel	Kernel function. Supported values are "rbfdot", and "laplacedot". Default is "rbfdot".
nfolds	The number of folds in cross-validation. Default is 5.
foldid	An optional vector which indexed the observations into each cross-validation fold. If supplied, nfolds is overridden.
parallel	If TRUE, fit cross-validation folds in parallel. Default is FALSE.
ncores	Number of worker processes used when parallel=TRUE.
Kfull	Optional precomputed full n by n kernel matrix. If supplied, fold-specific training and validation kernels are sliced from this matrix.
...	Additional arguments passed into kqr

Details

The function computes the average cross-validation error and reports the standard error.

Value

An object of class cv.kqr is returned, which is a list with the components describing the cross-validation error.

lambda	The lambda candidate values.
cvm	Mean cross-validation error.
cvsd	Estimates of standard error of cross-validation error.
cvup	The upper curve: cvm + cvsd.
cvlo	The lower curve: cvm - cvsd.
lambda.min	The lambda incurring the minimum cross-validation error.
lambda.1se	The largest lambda whose error is within one standard error of the minimum.
cv.min	The cross-validation error at lambda.min.
cv.1se	The cross-validation error at lambda.1se.

Examples

```
set.seed(1)
x <- matrix(rnorm(60), 30, 2)
y <- x[, 1] + rnorm(30)
lambda <- 10^seq(0, -2, length.out = 3)
foldid <- rep(1:3, length.out = nrow(x))
cv.fit <- cv.kqr(x, y, lambda = lambda, tau = 0.5, foldid = foldid)
```

cv.nckqr *cross-validation for selecting the tuning parameter 'lambda2' of non-crossing kernel quantile regression*

Description

Performs k-fold cross-validation for nckqr().

Usage

```
cv.nckqr(
  x,
  y,
  tau,
  lambda1 = NULL,
  lambda2 = NULL,
  sigma = NULL,
  kernel = "rbfdot",
  nfolds = 5L,
  foldid,
  parallel = FALSE,
  ncores = max(1L, parallel::detectCores() - 1L),
  Kfull = NULL,
  ...
)
```

Arguments

x	A numerical input matrix. The dimension is n rows and p columns.
y	Response variable.
tau	A user-supplied tau sequence.
lambda1	A user-supplied lambda1 value.
lambda2	A user-supplied lambda2 sequence.
sigma	Kernel bandwidth parameter used by "rbfdot" and "laplacedot".
kernel	Kernel function. Supported values are "rbfdot", and "laplacedot". Default is "rbfdot".
nfolds	The number of folds in cross-validation. Default is 5.
foldid	An optional vector which indexed the observations into each cross-validation fold. If supplied, nfolds is overridden.
parallel	If TRUE, fit cross-validation folds in parallel. Default is FALSE.
ncores	Number of worker processes used when parallel=TRUE.
Kfull	Optional precomputed full n by n kernel matrix. If supplied, fold-specific training and validation kernels are sliced from this matrix.
...	Additional arguments passed into nckqr

Details

The function computes the average cross-validation error and reports the standard error.

Value

An object of class `cv.nckqr` is returned, which is a list with the components describing the cross-validation error.

`lambda2`	The `lambda2` candidate values.
`cvm`	Mean cross-validation error.
`cvsd`	Estimates of standard error of cross-validation error.
`cvup`	The upper curve: `cvm` + `cvsd`.
`cvlo`	The lower curve: `cvm` - `cvsd`.
`lambda.min`	The `lambda2` incurring the minimum cross-validation error.
`lambda.1se`	The largest `lambda` whose error is within one standard error of the minimum.
`cv.min`	The cross-validation error at `lambda.min`.
`cv.1se`	The cross-validation error at `lambda.1se`.

Examples

```
set.seed(1)
x <- matrix(rnorm(60), 30, 2)
y <- x[, 1] + rnorm(30)
lambda <- 10^seq(0, -2, length.out = 3)
foldid <- rep(1:3, length.out = nrow(x))
cvres <- cv.nckqr(x, y, tau = c(0.25, 0.5, 0.75),
  lambda1 = 1, lambda2 = lambda, foldid = foldid)
```

`cv.qr` *Cross-Validation for Regularized Linear Quantile Regression*

Description

Selects the ridge penalty for `qr()` by K-fold cross-validation.

Usage

```
cv.qr(
  x,
  y,
  lambda,
  tau = 0.5,
  nfolds = 5L,
  foldid = NULL,
  delta = 0.125,
```

```
  eps = 1e-05,
  maxit = 1e+06,
  gam = 1e-07,
  is_exact = FALSE,
  standardize = FALSE,
  keep.x = FALSE,
  parallel = FALSE,
  ncores = max(1L, parallel::detectCores() - 1L),
  ...
)
```

Arguments

x	Numeric predictor matrix.
y	Numeric response vector.
lambda	Positive ridge penalty sequence.
tau	Single quantile level in $(0, 1)$.
nfolds	Number of folds.
foldid	Optional fold assignments.
delta	Smoothing parameter passed to qr().
eps	Stopping criterion passed to qr().
maxit	Maximum iterations passed to qr().
gam	Numerical ridge passed to qr().
is_exact	Whether to use exact projection in qr().
standardize	Standardization is not implemented; must be FALSE.
keep.x	Whether to store training x in the final model.
parallel	Parallel CV is not implemented for this first version.
ncores	Reserved for future parallel CV support.
...	Compatibility arguments. method="fast_lqr" is accepted; removed linear QR methods are rejected.

Value

An object of class cv.fastlqr.

kqr *Solve the kernel quantile regression. The solution path is computed at a grid of values of tuning parameter* lambda.

Description

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

Usage

```
kqr(
  x,
  y,
  lambda,
  tau,
  delta = 0.125,
  eps = 1e-05,
  maxit = 1e+06,
  gam = 1e-07,
  sigma = NULL,
  kernel = "rbfdot",
  Kmat = NULL,
  is_exact = FALSE
)
```

Arguments

x	A numerical input matrix. The dimension is n rows and p columns.
y	Response variable. The length is n.
lambda	A user-supplied lambda sequence.
tau	A user-supplied tau value for a quantile level.
delta	The smoothing index for method='huber'. Default is 0.125.
eps	Stopping criterion.
maxit	Maximum number of iterates.
gam	A small number for numerical stability.
sigma	Kernel bandwidth parameter used by "rbfdot" and "laplacedot".
kernel	Kernel function. Supported values are "rbfdot", and "laplacedot". Default is "rbfdot".
Kmat	Optional precomputed training kernel matrix. If supplied, it is reused instead of recomputing kernelMat(x, x, sigma, kernel).
is_exact	Exact or approximated solutions. Default is FALSE.

Details

The function implements an accelerated proximal gradient descent to solve kernel quantile regression.

Value

An object with S3 class kqr

alpha	An $n + 1$ by L matrix of coefficients, where n is the number of observations and L is the number of tuning parameters. The first row of alpha contains the intercepts.
lambda	The lambda sequence that was actually used.
delta	The smoothing index.
npass	The total number of iterates used to train the classifier.
jerr	Warnings and errors; 0 if none.
info	A list includes some settings used to fit this object: eps, maxit

.

Examples

```
set.seed(1)
x <- matrix(rnorm(60), 30, 2)
y <- x[, 1] + rnorm(30)
lambda <- 10^seq(0, -2, length.out = 3)
fit <- kqr(x, y, lambda = lambda, tau = 0.5)
```

---

nckqr *Solve the non-crossing kernel quantile regression*

Description

Trains the kernel quantile regression

Usage

```
nckqr(
  x,
  y,
  lambda1,
  lambda2,
  tau,
  delta = 0.125,
  eps = 1e-08,
  maxit = 5e+06,
  gam = 1e-07,
```

```
  sigma = NULL,
  kernel = "rbfdot",
  Kmat = NULL,
  is_exact = FALSE
)
```

Arguments

x	A numerical input matrix with n rows and p columns.
y	Response variable. The length is n.
lambda1	A user-supplied lambda1 sequence. The length is $L1$.
lambda2	A user-supplied lambda2 sequence. The length is $L2$.
tau	A user-supplied tau sequence for quantile levels. The length is $ntau$.
delta	The smoothing index for method='huber'. Default is 0.125.
eps	Stopping criterion.
maxit	Maximum number of iterates.
gam	A small number for numerical stability.
sigma	Kernel bandwidth parameter used by "rbfdot" and "laplacedot".
kernel	Kernel function. Supported values are "rbfdot", and "laplacedot". Default is "rbfdot".
Kmat	Optional precomputed training kernel matrix. If supplied, it is reused instead of recomputing kernelMat(x, x, sigma, kernel).
is_exact	Exact or approximated solutions.

Details

The function implements the majorization-minimization method to solve non-crossing kernel quantile regression.

Value

An object with S3 class nckqr

alpha	An $n+1$ by L matrix of coefficients, where n represents the number of observations, $ntau$ represents the number of quantile levels, and L denotes the number of tuning parameters.
tau	The tau sequence that was actually used.
lambda1	The lambda1 sequence that was actually used.
lambda2	The lambda2 sequence that was actually used.
delta	The smoothing index.
npass	The total number of iterates used to train the classifier.
jerr	Warnings and errors; 0 if none.
info	A list includes some settings used to fit this object: eps, maxit

.

Examples

```
set.seed(1)
x <- matrix(rnorm(60), 30, 2)
y <- x[, 1] + rnorm(30)
tau <- c(0.25, 0.5, 0.75)
lambda1 <- 1
lambda2 <- 10^seq(0, -2, length.out = 3)
fit <- nckqr(x, y, lambda1 = lambda1, lambda2 = lambda2, tau = tau)
```

---

predict.cv.fastlqr *Predict from a cv.fastlqr Object*

Description

Predict from a cv.fastlqr Object

Usage

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

Arguments

object	A fitted cv.fastlqr object.
newx	New predictor matrix.
s	"lambda.1se", "lambda.min", or numeric lambda value(s).
...	Passed to predict.fastlqr().

Value

A matrix of fitted linear quantiles.

---

predict.fastlqr *Predict from a fastlqr Object*

Description

Predict from a fastlqr Object

Usage

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

Arguments

object	A fitted fastlqr object.
newx	New predictor matrix. If omitted, training data are used when the model was fit with keep.x=TRUE.
s	Optional lambda value(s).
...	Not used.

Value

A matrix of fitted linear quantiles.

---

predict.kqr *Predict the fitted values for a* kqr *object.*

Description

Predict the fitted values for a kqr object.

Usage

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

Arguments

object	A fitted kqr object.
x	The predictor matrix, i.e., the x matrix used when fitting the kqr object.
newx	A matrix of new values for x at which predictions are to be made. Note that newx must be of a matrix form, predict function does not accept a vector or other formats of newx.
s	Value(s) of the penalty parameter lambda at which predictions are required. Default is the entire sequence used to create the model.
...	Not used.

Details

The result is $\beta_0 + K_i'\alpha$ where β_0 and α are from the kqr object and K_i is the ith row of the kernel matrix.

Value

Returns the fitted values.

Examples

```
set.seed(1)
x <- matrix(rnorm(60), 30, 2)
y <- x[, 1] + rnorm(30)
lambda <- 10^seq(0, -2, length.out = 3)
fit <- kqr(x, y, lambda = lambda, tau = 0.5)
predict(fit, x, x[1:5, , drop = FALSE])
```

---

predict.nckqr *Predict the fitted values for a* nckqr *object.*

Description

Predict the fitted values for a nckqr object.

Usage

```
## S3 method for class 'nckqr'
predict(object, x, newx = NULL, s2, s1 = NULL, ...)
```

Arguments

object	A fitted nckqr object.
x	The predictor matrix, i.e., the x matrix used when fitting the nckqr object.
newx	A matrix of new values for x at which predictions are to be made. Note that newx must be of a matrix form, predict function does not accept a vector or other formats of newx.
s2	Value of the penalty parameter lambda2 at which predictions are required.
s1	Value(s) of the penalty parameter lambda1 at which predictions are required. Default is the entire sequence used to create the model.
...	Not used.

Value

Returns the fitted values for the non-crossing kernel quantile regression model.

Examples

```
set.seed(1)
x <- matrix(rnorm(60), 30, 2)
y <- x[, 1] + rnorm(30)
tau <- c(0.25, 0.5, 0.75)
lambda1 <- 1
lambda2 <- 10^seq(0, -2, length.out = 3)
fit <- nckqr(x, y, lambda1 = lambda1, lambda2 = lambda2, tau = tau)
predict(fit, x, x[1:5, , drop = FALSE], s1 = lambda1, s2 = lambda2[1])
```

qr *Regularized Linear Quantile Regression*

Description

Fits a parametric linear quantile regression path $Q_\tau(Y|X) = \beta_0 + X\beta$.

Usage

```
qr(
  x,
  y,
  lambda,
  tau = 0.5,
  delta = 0.125,
  eps = 1e-05,
  maxit = 1e+06,
  gam = 1e-07,
  is_exact = FALSE,
  standardize = FALSE,
  intercept = TRUE,
  keep.x = FALSE,
  ...
)
```

Arguments

x	Numeric predictor matrix.
y	Numeric response vector.
lambda	Positive ridge penalty sequence.
tau	Single quantile level in $(0, 1)$.
delta	Smoothing parameter used by the Fortran solver.
eps	Stopping criterion.
maxit	Maximum number of iterations.
gam	Small ridge added internally for numerical stability.
is_exact	Whether to run the exact projection path.
standardize	Standardization is not implemented; must be FALSE.
intercept	Intercept is required by fast_lqr; must be TRUE.
keep.x	If TRUE, store the training matrix for later prediction.
...	Compatibility arguments. method="fast_lqr" is accepted; removed linear QR methods are rejected.

Details

The objective is

$$n^{-1} \sum_i \rho_\tau(y_i - \beta_0 - x_i^T \beta) + \lambda ||\beta||_2^2 / 2.$$

The intercept is not penalized. This function masks `base::qr()`; use `base::qr()` when QR decomposition is needed. This parametric linear model is the supported path for linear quantile regression; kernel methods `kqr()` and `nckqr()` no longer support `kernel = "lineardot"`.

Value

An object of class `fastlqr`.

Examples

```
set.seed(1)
x <- matrix(rnorm(60), 30, 2)
y <- x[, 1] + rnorm(30)
lambda <- 10^seq(0, -2, length.out = 3)
fit <- qr(x, y, lambda = lambda, tau = 0.5)
predict(fit, x[1:5, , drop = FALSE])
```

Index

∗ **classification**
    predict.kqr, 14
∗ **kernel**
    cv.kqr, 5
    cv.nckqr, 7
    predict.kqr, 14
    predict.nckqr, 15
∗ **quantile**
    cv.kqr, 5
    cv.nckqr, 7
    kqr, 10
    nckqr, 11
∗ **regression**
    cv.kqr, 5
    cv.nckqr, 7
    kqr, 10
    nckqr, 11
    predict.nckqr, 15

coef.cv.fastlqr, 2
coef.fastlqr, 3
coef.kqr, 3
coef.nckqr, 4
cv.kqr, 5
cv.nckqr, 7
cv.qr, 8

kqr, 10

nckqr, 11

predict.cv.fastlqr, 13
predict.fastlqr, 13
predict.kqr, 14
predict.nckqr, 15

qr, 16