Package ‘qrcmNP’

January 23, 2024

Type Package

Title Nonlinear and Penalized Quantile Regression Coefficients Modeling

Version 0.2.1

Author Gianluca Sottile [aut, cre]

Maintainer Gianluca Sottile <gianluca.sottile@unipa.it>

Description Nonlinear and Penalized parametric modeling of quantile regression coefficient functions. Sottile G, Frumento P, Chiodi M and Bottai M (2020) <doi:10.1177/1471082X19825523>.

Imports stats, graphics, grDevices, utils

Depends survival (>= 2.4.1), qrcm (>= 3.0)

License GPL-2

RoxygenNote 7.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2024-01-23 09:40:03 UTC

R topics documented:

qrcmNP-package *Nonlinear and Penalized Quantile Regression Coefficients Modeling*

Description

This package implements a nonlinear Frumento and Bottai's (2016) method for quantile regression coefficient modeling (qrcm), in which quantile regression coefficients are described by (flexible) parametric functions of the order of the quantile. In the classical qrcm framework the linearity in $\beta(p)$ and/or in θ could be relaxed at a cost of more complicated expressions for the ojective and the gradient functions. Here, we propose an efficiently algorithm to use more flexible structures for the regression coefficients. With respect to the most famous function nlrq (quantreg package) our main function niqr implements the integrated quantile regression idea of Frumento and Bottai's (2016) for nonlinear functions. As already known, this practice allows to estimate quantiles all at one time and not one at a time. This package also implements a penalized Frumento and Bottai's (2015) method for quantile regression coefficient modeling (qrcm). This package fits lasso qrcm using pathwise coordinate descent algorithm. With respect to some other packages which implements the L1-quantile regression (e.g. quantreg, rqPen) estimating quantiles one at a time our proposal allows to estimate the conditional quantile function parametrically estimating quantiles all at one and to do variable selction in the meanwhile.

Details

Package:	qrcmNP
Type:	Package
Version:	0.2.1
Date:	2024-01-22
License:	GPL-2

The function `niqr` permits specifying nonlinear basis for each variables. The function `testfit.niqr` permits to do goodness of fit. The auxiliary functions `summary.niqr`, `predict.niqr`, and `plot.niqr` can be used to extract information from the fitted model. The function `piqr` permits specifying the lasso regression model. The function `gof.piqr` permits to select the best tuning parameter through AIC, BIC, GIC and GCV criteria. The auxiliary functions `summary.piqr`, `predict.piqr`, and `plot.piqr` can be used to extract information from the fitted model.

Author(s)

Gianluca Sottile

Maintainer: Gianluca Sottile <gianluca.sottile@unipa.it>

References

Sottile G, Frumento P, Chiodi M, Bottai M. (2020). *A penalized approach to covariate selection through quantile regression coefficient models*. Statistical Modelling, 20(4), pp 369-385. doi:10.1177/1471082X19825523.

Frumento, P., and Bottai, M. (2016). *Parametric modeling of quantile regression coefficient functions*. Biometrics, 72(1), pp 74-84, doi:10.1111/biom.12410.

Friedman, J., Hastie, T. and Tibshirani, R. (2008). *Regularization Paths for Generalized Linear Models via Coordinate Descent*. Journal of Statistical Software, Vol. 33(1), pp 1-22 Feb 2010.

Examples

```
# use simulated data

n <- 300
x <- runif(n)
fun <- function(theta, p){
  beta0 <- theta[1] + exp(theta[2]*p)
  beta1 <- theta[3] + theta[4]*p
  cbind(beta0, beta1)}
beta <- fun(c(1,1,1,1), runif(n))
y <- beta[, 1] + beta[, 2]*x
model <- niqr(fun=fun, x0=rep(0, 4), X=cbind(1,x), y=y)

# use simulated data

set.seed(1234)
n <- 300
x1 <- rexp(n)
x2 <- runif(n, 0, 5)
x <- cbind(x1,x2)

b <- function(p){matrix(cbind(1, qnorm(p), slp(p, 2)), nrow=4, byrow=TRUE)}
theta <- matrix(0, nrow=3, ncol=4); theta[, 1] <- 1; theta[1,2] <- 1; theta[2:3,3] <- 2
qy <- function(p, theta, b, x){rowSums(x * t(theta %*% b(p)))}

y <- qy(runif(n), theta, b, cbind(1, x))

s <- matrix(1, nrow=3, ncol=4); s[1,3:4] <- 0
obj <- piqr(y ~ x1 + x2, formula.p = ~ I(qnorm(p)) + slp(p, 2), s=s, nlambda=50)

best <- gof.piqr(obj, method="AIC", plot=FALSE)
best2 <- gof.piqr(obj, method="BIC", plot=FALSE)

summary(obj, best$posMinLambda)
summary(obj, best2$posMinLambda)
```

---

gof.piqr — *Goodness of Fit of Penalized Quantile Regression Coefficients Modeling*

Description

Goodness of Fit of an object of class “`piqr`”, usefull to select the best tuning parameter.

Usage

```
gof.piqr(object, method=c("BIC","AIC"), Cn="1", plot=TRUE, df.new=TRUE, logi=TRUE, ...)
```

Arguments

`object`	an object of class "`piqr`", the result of a call to `piqr`.
`method`	a method to evaluate the goodness of fit and select the best value of the tuning parameter.
`Cn`	It is some positive constant that diverges to infinity as n increase. It is used by the BIC criterion and if Cn = 1 the classical BIC is used.
`df.new`	if TRUE degrees of freedom are evaluated as the number of $$\beta_j(p\|\theta)! = 0, j = 1, \ldots, q$$.
`logi`	if TRUE the loss function is log-transformed.
`plot`	if TRUE the chosen method is plotted - default is TRUE.
`...`	additional arguments.

Details

The best value of lambda is chosen minimizing the criterion, i.e., AIC and BIC.

Value

`minLambda`	the best value of lambda.
`dfMinLambda`	the number of nonzero parameters associated to the best lambda.
`betaMin`	the parameters associated to the best lambda.
`posMinLambda`	the position of the best lambda along the sequence of lambda.
`call`	the matched call.

Author(s)

Gianluca Sottile <gianluca.sottile@unipa.it>

See Also

`piqr`, for model fitting; `summary.piqr` and `plot.piqr`, for summarizing and plotting `piqr` objects.

Examples

```
# using simulated data

set.seed(1234)
n <- 300
x1 <- rexp(n)
```

```
x2 <- runif(n, 0, 5)
x <- cbind(x1,x2)

b <- function(p){matrix(cbind(1, qnorm(p), slp(p, 2)), nrow=4, byrow=TRUE)}
theta <- matrix(0, nrow=3, ncol=4); theta[, 1] <- 1; theta[1,2] <- 1; theta[2:3,3] <- 2
qy <- function(p, theta, b, x){rowSums(x * t(theta %*% b(p)))}

y <- qy(runif(n), theta, b, cbind(1, x))

s <- matrix(1, nrow=3, ncol=4); s[1,3:4] <- 0
obj <- piqr(y ~ x1 + x2, formula.p = ~ I(qnorm(p)) + slp(p, 2), s=s, nlambda=50)

par(mfrow=c(1,2))
best <- gof.piqr(obj, method="AIC", plot=TRUE)
best2 <- gof.piqr(obj, method="BIC", plot=TRUE)
```

---

niqr *Nonlinear Quantile Regression Coefficients Modeling*

Description

This package implements a nonlinear Frumento and Bottai's (2015) method for quantile regression coefficient modeling (qrcm), in which quantile regression coefficients are described by (flexible) parametric functions of the order of the quantile.

Usage

```
niqr(fun, fun2, x0, X, y, control=list())
```

Arguments

fun	a function of theta and p describing the beta functions.
fun2	a function of beta and X describing the whole quantile process.
x0	starting values to search minimum.
X	a design matrix containing the intercept.
y	the response variable.
control	a list of control parameters. See 'Details'.

Details

Quantile regression permits modeling conditional quantiles of a response variabile, given a set of covariates.

Assume that each coefficient can be expressed as a parametric function of θ, p of the form:

$$\beta(\theta, p) = b_0(\theta_0, p) + b_1(\theta_1, p) + b_2(\theta_2, p) + \ldots.$$

Users are required to specify a function of θ and p and to provide starting points for the minimization, the design matrix (with intercept) and the response variable. Some control paramters such as, tol=1e-6, $\alpha = .1$, $\beta = .5$, maxit=200, maxitstart=20, cluster=NULL, display=FALSE, $\epsilon = 1e - 12$, a1=.001, h1=1e-4, meth="2", lowp=.01, upp=.99, np=100 could be modified from their default. α and β are parameters for line search, tol, epsilon, maxit, and a1 are parameters for quasi Newthod approach, maxit start is the maximum number of iteration for guessing the best start values, h1 and meth (method) are parameters for the gradient (method="2" is centered formula, it is possible to select "1" for right and "3" for five points stencil), lowp, upp and np are parameters used in the integral formula, and cluster if not NULL is a vector of ID to compute standard errors in longitudinal data. fun_prime_theta and fun_prime_beta are the gradient functions of the quantile function with respect to θ and β

Value

An object of class "`niqr`", a list containing the following items:

`call`	the matched call.
`x`	the optimal θ values.
`se`	standard errors for θ.
`fx`	the value of the minimized integrated loss function.
`gx`	the gradient calculated in the optimal points.
`Hx`	the hessian of quasi newthon method
`omega`	gradient matrix used in the sandwich formaula to compute standard errors.
`covar`	variance covariance matrix $H^(-1)\Omega H^(-1)$.
`p.star.y`	the CDF calculated in the optimal x values.
`code`	the convergence code, 0 for convergence, 1 for maxit reached, 2 no more step in the gradient.
`internal`	a list containing some initial and control object.

Author(s)

Gianluca Sottile <gianluca.sottile@unipa.it>

References

Frumento, P., and Bottai, M. (2015). *Parametric modeling of quantile regression coefficient functions*. Biometrics, doi: 10.1111/biom.12410.

See Also

`summary.niqr`, `plot.niqr`, `predict.niqr`, for summary, plotting, and prediction. `testfit.niqr` for goodness of fit.

Examples

```
set.seed(1234)
n <- 300
x <- runif(n)
fun <- function(theta, p){
  beta0 <- theta[1] + exp(theta[2]*p)
  beta1 <- theta[3] + theta[4]*p
  cbind(beta0, beta1)}
beta <- fun(c(1,1,1,1), runif(n))
y <- beta[, 1] + beta[, 2]*x
model <- niqr(fun=fun, x0=rep(.5, 4), X=cbind(1,x), y=y)

## Not run:
# NOT RUN---qgamma function
set.seed(1234)
n <- 1000
x <- runif(n)
fun2 <- function(theta, p){
  beta0 <- theta[1] + qgamma(p, exp(theta[2]), exp(theta[3]))
  beta1 <- theta[4] + theta[5]*p
  cbind(beta0, beta1)
}
beta <- fun2(c(1,2,2,1,1), runif(n))
y <- beta[, 1] + beta[, 2]*x
model <- niqr(fun=fun2, x0=rep(.5, 5), X=cbind(1,x), y=y)

# NOT RUN---qbeta function
set.seed(1234)
n <- 1000
x <- runif(n)
fun3 <- function(theta, p){
  beta0 <- theta[1] + theta[2]*qbeta(p, exp(theta[3]), exp(theta[4]))
  beta1 <- theta[5] + theta[6]*p
  cbind(beta0, beta1)
}
beta <- fun3(c(1,1.5,.5,.2,1,1), runif(n))
y <- beta[, 1] + beta[, 2]*x
model <- niqr(fun=fun3, x0=rep(.5, 6), X=cbind(1,x), y=y)

# NOT RUN---qt function
set.seed(1234)
n <- 1000
x <- runif(n)
fun4 <- function(theta, p){
  beta0 <- theta[1] + exp(theta[2])*qt(p, 1+exp(theta[3]), exp(theta[4]))
  beta1 <- theta[5] + theta[6]*p
  cbind(beta0, beta1)
}
beta <- fun4(c(1,.5,.3,.2,1,1), runif(n))
y <- beta[, 1] + beta[, 2]*x
model <- niqr(fun=fun4, x0=rep(.5, 6), X=cbind(1,x), y=y)
```

```
## End(Not run)

# see the documentation for 'summary.piqr', and 'plot.piqr'
```

---

piqr *Penalized Quantile Regression Coefficients Modeling*

Description

This package implements a penalized Frumento and Bottai's (2016) method for quantile regression coefficient modeling (qrcm), in which quantile regression coefficients are described by (flexible) parametric functions of the order of the quantile. This package fits lasso qrcm using pathwise coordinate descent algorithm.

Usage

```
piqr(formula, formula.p = ~ slp(p, 3), weights, data, s, nlambda=100,
     lambda.min.ratio=ifelse(nobs<nvars, 0.01, 0.0001), lambda,
     tol=1e-6, maxit=100, display=TRUE)
```

Arguments

`formula`	a two-sided formula of the form `y ~ x1 + x2 + ...`: a symbolic description of the quantile regression model.
`formula.p`	a one-sided formula of the form `~ b1(p, ...) + b2(p, ...) + ...`, describing how quantile regression coefficients depend on `p`, the order of the quantile.
`weights`	an optional vector of weights to be used in the fitting process.
`data`	an optional data frame, list or environment containing the variables in `formula`.
`s`	an optional 0/1 matrix that permits excluding some model coefficients (see 'Examples').
`nlambda`	the number of lambda values - default is 100.
`lambda.min.ratio`	Smallest value for lambda, as a fraction of lambda.max. The default depends on the sample size nobs relative to the number of variables nvars. If nobs > nvars, the default is 0.0001, close to zero. If nobs < nvars, the default is 0.01.
`lambda`	A user supplied lambda sequence.
`display`	if TRUE something is printed - default is TRUE.
`tol`	convergence criterion for numerical optimization - default is 1e-6.
`maxit`	maximum number of iterations - default is 100.

Details

Quantile regression permits modeling conditional quantiles of a response variabile, given a set of covariates. A linear model is used to describe the conditional quantile function:

$$Q(p|x) = \beta_0(p) + \beta_1(p)x_1 + \beta_2(p)x_2 + \ldots .$$

The model coefficients $\beta(p)$ describe the effect of covariates on the p-th quantile of the response variable. Usually, one or more quantiles are estimated, corresponding to different values of p.

Assume that each coefficient can be expressed as a parametric function of p of the form:

$$\beta(p|\theta) = \theta_0 + \theta_1 b_1(p) + \theta_2 b_2(p) + \ldots$$

where $b_1(p), b_2(p, \ldots)$ are known functions of p. If q is the dimension of $x = (1, x_1, x_2, \ldots)$ and k is that of $b(p) = (1, b_1(p), b_2(p), \ldots)$, the entire conditional quantile function is described by a $q \times k$ matrix θ of model parameters.

Users are required to specify two formulas: `formula` describes the regression model, while `formula.p` identifies the 'basis' $b(p)$. By default, `formula.p = ~ slp(p, k = 3)`, a 3rd-degree shifted Legendre polynomial (see `slp`). Any user-defined function $b(p, \ldots)$ can be used, see 'Examples'.

Estimation of penalized θ is carried out by minimizing a penalized integrated loss function, corresponding to the integral, over p, of the penalized loss function of standard quantile regression. This motivates the acronym `piqr` (penalized integrated quantile regression).

See details in `iqr`

Value

An object of class "`piqr`", a list containing the following items:

`call`	the matched call.
`lambda`	The actual sequence of lambda values used.
`coefficients`	a list of estimated model parameters describing the fitted quantile function along the path.
`minimum`	the value of the minimized integrated loss function for each value of lambda.
`dl`	a matrix of gradient values along the path.
`df`	The number of nonzero coefficients for each value of lambda.
`seqS`	a list containg each matrix s for each value of lambda.
`internal`	a list containing some initial object.

Note

By expressing quantile regression coefficients as functions of p, a parametric model for the conditional quantile function is specified. The induced `PDF` and `CDF` can be used as diagnostic tools. Negative values of `PDF` indicate quantile crossing, i.e., the conditional quantile function is not monotonically increasing. Null values of `PDF` indicate observations that lie outside the estimated support of the data, defined by quantiles of order 0 and 1. If null or negative `PDF` values occur for a relatively large proportion of data, the model is probably misspecified or ill-defined. If the model is correct, the fitted `CDF` should approximately follow a Uniform(0,1) distribution. This idea is used to implement a goodness-of-fit test, see `summary.iqr` and `test.fit`.

The intercept can be excluded from formula, e.g., iqr(y ~ -1 + x). This, however, implies that when x = 0, y is always 0. See example 5 in 'Examples'. The intercept can also be removed from formula.p. This is recommended if the data are bounded. For example, for strictly positive data, use iqr(y ~ 1, formula.p = -1 + slp(p,3)) to force the smallest quantile to be zero. See example 6 in 'Examples'.

Author(s)

Gianluca Sottile <gianluca.sottile@unipa.it>

References

Sottile G, Frumento P, Chiodi M, Bottai M. (2020). *A penalized approach to covariate selection through quantile regression coefficient models*. Statistical Modelling, 20(4), pp 369-385. doi:10.1177/1471082X19825523.

Frumento, P., and Bottai, M. (2016). *Parametric modeling of quantile regression coefficient functions*. Biometrics, 72(1), pp 74-84, doi:10.1111/biom.12410.

Friedman, J., Hastie, T. and Tibshirani, R. (2008). *Regularization Paths for Generalized Linear Models via Coordinate Descent*. Journal of Statistical Software, Vol. 33(1), pp 1-22 Feb 2010.

See Also

summary.piqr, plot.piqr, predict.piqr, for summary, plotting, and prediction. gof.piqr to select the best value of the tuning parameter though AIC, BIC, GIC, GCV criteria.

Examples

```
##### Using simulated data in all examples

##### Example 1
set.seed(1234)
n <- 300
x1 <- rexp(n)
x2 <- runif(n, 0, 5)
x <- cbind(x1,x2)

b <- function(p){matrix(cbind(1, qnorm(p), slp(p, 2)), nrow=4, byrow=TRUE)}
theta <- matrix(0, nrow=3, ncol=4); theta[, 1] <- 1; theta[1,2] <- 1; theta[2:3,3] <- 2
qy <- function(p, theta, b, x){rowSums(x * t(theta %*% b(p)))}

y <- qy(runif(n), theta, b, cbind(1, x))

s <- matrix(1, nrow=3, ncol=4); s[1,3:4] <- 0; s[2:3, 2] <- 0
obj <- piqr(y ~ x1 + x2, formula.p = ~ I(qnorm(p)) + slp(p, 2), s=s, nlambda=50)

best <- gof.piqr(obj, method="AIC", plot=FALSE)
best2 <- gof.piqr(obj, method="BIC", plot=FALSE)

summary(obj, best$posMinLambda)
summary(obj, best2$posMinLambda)
```

```
## Not run:
##### other examples
set.seed(1234)
n <- 1000
q <- 5
k <- 3
X <- matrix(abs(rnorm(n*q)), n, q)
rownames(X) <- 1:n
colnames(X) <- paste0("X", 1:q)
theta <- matrix(c(3, 1.5, 1, 1,
                  2, 1, 1, 1,
                  0, 0, 0, 0,
                  0, 0, 0, 0,
                  1.5, 1, 1, 1,
                  0, 0, 0, 0),
                ncol=(k+1), byrow=TRUE)
rownames(theta) <- c("(intercept)", paste0("X", 1:q))
colnames(theta) <- c("(intercept)", "slp(p,1)", "slp(p,2)", "slp(p,3)")
B <- function(p, k){matrix(cbind(1, slp(p, k)), nrow=(k+1), byrow=TRUE)}
Q <- function(p, theta, B, k, X){rowSums(X * t(theta %*% B(p, k)))}

pp <- runif(n)
y <- Q(p=pp, theta=theta, B=B, k=k, X=cbind(1, X))
m1 <- piqr(y ~ X, formula.p = ~ slp(p, k))
best1 <- gof.piqr(m1, method="AIC", plot=FALSE)
best2 <- gof.piqr(m1, method="BIC", plot=FALSE)
summary(m1, best1$posMinLambda)
summary(m1, best2$posMinLambda)
par(mfrow = c(1,3)); plot(m1, xvar="lambda");
                     plot(m1, xvar="objective"); plot(m1, xvar="grad")

set.seed(1234)
n <- 1000
q <- 6
k <- 4
# x <- runif(n)
X <- matrix(abs(rnorm(n*q)), n, q)
rownames(X) <- 1:n
colnames(X) <- paste0("X", 1:q)
theta <- matrix(c(1, 2, 0, 0, 0,
                  2, 0, 1, 0, 0,
                  0, 0, 0, 0, 0,
                  1, 0, 0, 1, -1.2,
                  0, 0, 0, 0, 0,
                  1.5, 0, .5, 0, 0,
                  0, 0, 0, 0, 0),
                ncol=(k+1), byrow=TRUE)
rownames(theta) <- c("(intercept)", paste0("X", 1:q))
colnames(theta) <- c("(intercept)", "qnorm(p)", "p", "log(p)", "log(1-p)")
B <- function(p, k){matrix(cbind(1, qnorm(p), p, log(p), log(1-p)), nrow=(k+1), byrow=TRUE)}
Q <- function(p, theta, B, k, X){rowSums(X * t(theta %*% B(p, k)))}
```

```
pp <- runif(n)
y <- Q(p=pp, theta=theta, B=B, k=k, X=cbind(1, X))
s <- matrix(1, q+1, k+1); s[2:(q+1), 2] <- 0; s[1, 3:(k+1)] <- 0; s[2:3, 4:5] <- 0
s[4:5, 3] <- 0; s[6:7, 4:5] <- 0
m2 <- piqr(y ~ X, formula.p = ~ qnorm(p) + p + I(log(p)) + I(log(1-p)), s=s)
best1 <- gof.piqr(m2, method="AIC", plot=FALSE)
best2 <- gof.piqr(m2, method="BIC", plot=FALSE)
summary(m2, best1$posMinLambda)
summary(m2, best2$posMinLambda)
par(mfrow = c(1,3)); plot(m2, xvar="lambda");
                     plot(m2, xvar="objective"); plot(m2, xvar="grad")

## End(Not run)
  # see the documentation for 'summary.piqr', and 'plot.piqr'
```

`plot.niqr` *Plot Nonlinear Quantile Regression Coefficients*

Description

Plots quantile regression coefficients $\beta(\theta, p)$ as a function of p, based on a fitted model of class "niqr".

Usage

```
## S3 method for class 'niqr'
plot(x, conf.int=TRUE, which=NULL, ask=TRUE, ...)
```

Arguments

`x`	an object of class "niqr", typically the result of a call to `niqr`.
`conf.int`	logical. If TRUE, asymptotic 95% confidence intervals are added to the plot.
`which`	an optional numerical vector indicating which coefficient(s) to plot. If which = NULL, all coefficients are plotted.
`ask`	logical. If which = NULL and ask = TRUE (the default), you will be asked interactively which coefficients to plot.
`...`	additional graphical parameters, that can include `xlim, ylim, xlab, ylab, col, lwd`. See `par`.

Author(s)

Gianluca Sottile <gianluca.sottile@unipa.ot>

See Also

`niqr` for model fitting; `testfit.niqr` for goodness of fit test; `summary.niqr` and `predict.niqr` for model summary and prediction.

Examples

```
  # using simulated data

n <- 300
x <- runif(n)
fun <- function(theta, p){
  beta0 <- theta[1] + exp(theta[2]*p)
  beta1 <- theta[3] + theta[4]*p
  cbind(beta0, beta1)}
beta <- fun(c(1,1,1,1), runif(n))
y <- beta[, 1] + beta[, 2]*x
model <- niqr(fun=fun, x0=rep(0, 4), X=cbind(1, x), y=y)

plot(model, ask=FALSE)
```

`plot.piqr` *Plot Penalized Quantile Regression Coefficients*

Description

Produces a coefficient profile plot of the quantile regression coefficient paths for a fitted model of class “`piqr`”.

Usage

```
## S3 method for class 'piqr'
plot(x, xvar=c("lambda", "objective", "grad", "beta"), pos.lambda,
                      label=FALSE, which=NULL, ask=TRUE, polygon=TRUE, ...)
```

Arguments

`x`	an object of class “`piqr`”, typically the result of a call to `piqr`.
`xvar`	What is on the X-axis. "lambda" against the log-lambda sequence, "objective" against the value of the minimized integrated loss function and "grad" the log-lambda sequence against the gradient. xvar = "beta" needs a lambda value to plot quantile regression coefficients $\beta(p\|\theta(\lambda))$ as a function of p, based on the fitted model of class “`piqr`”
`pos.lambda`	the position of a lambda in the sequence of the object of class “`piqr`”. Could be the best after selecting the result of a call to `gof.piqr`
`label`	If TRUE, label the curves with variable sequence numbers.
`which`	an optional numerical vector indicating which coefficient(s) to plot. If which = NULL, all coefficients are plotted.
`ask`	logical. If which = NULL and ask = TRUE (the default), you will be asked interactively which coefficients to plot.

polygon	ogical. If TRUE, confidence intervals are represented by shaded areas via polygon. Otherwise, dashed lines are used.
...	additional graphical parameters, that can include `xlim, ylim, xlab, ylab, col, lwd`. See `par`.

Details

A coefficient profile plot is produced.

Author(s)

Gianluca Sottile <gianluca.sottile@unipa.ot>

See Also

`piqr` for model fitting; `gof.piqr` for the model selection criteria; `summary.piqr` and `predict.piqr` for model summary and prediction.

Examples

```
# using simulated data

n <- 300
x <- runif(n)
qy <- function(p,x){p^2 + x*log(p)}
# true quantile function: Q(p | x) = beta0(p) + beta1(p)*x, with
   # beta0(p) = p^2
   # beta1(p) = log(p)
y <- qy(runif(n), x) # to generate y, plug uniform p in qy(p,x)

obj <- piqr(y ~ x, formula.p = ~ slp(p,3), nlambda=50)
best <- gof.piqr(obj, method="BIC", plot=FALSE)
par(mfrow = c(1,3))
plot(obj, xvar="lambda")
plot(obj, xvar="objective")
plot(obj, xvar="grad")
par(mfrow=c(1,2));plot(obj, xvar="beta", pos.lambda=best$posMinLambda, ask=FALSE)
# flexible fit with shifted Legendre polynomials
```

`predict.niqr` *Prediction After Nonlinear Quantile Regression Coefficients Modeling*

Description

Predictions from an object of class “`niqr`”.

Usage

```
## S3 method for class 'niqr'
predict(object, type=c("beta", "CDF", "QF", "sim"), newdata, p, ...)
```

Arguments

`object`	an object of class "`niqr`", the result of a call to `niqr`.
`type`	a character string specifying the type of prediction. See 'Details'.
`newdata`	an optional data frame in which to look for variables with which to predict. If omitted, the data are used. For `type = "CDF"`, it must include the response variable. Ignored if `type = "beta"`.
`p`	a numeric vector indicating the order(s) of the quantile to predict. Only used if `type = "beta"` or `type = "QF"`.
`...`	for future methods.

Details

Different type of prediction from the model.

Note

Prediction may generate quantile crossing if the support of the new covariates values supplied in `newdata` is different from that of the observed data.

Author(s)

Gianluca Sottile <gianluca.sottile@unipa.it>

See Also

`niqr`, for model fitting; `testfit.niqr`, to do goodness of fit test; `summary.niqr` and `plot.niqr`, for summarizing and plotting `niqr` objects.

Examples

```
# using simulated data

n <- 300
x <- runif(n)
fun <- function(theta, p){
  beta0 <- theta[1] + exp(theta[2]*p)
  beta1 <- theta[3] + theta[4]*p
  cbind(beta0, beta1)}
beta <- fun(c(1,1,1,1), runif(n))
y <- beta[, 1] + beta[, 2]*x
model <- niqr(fun=fun, x0=rep(0, 4), X=cbind(1,x), y=y)

# predict beta(0.25), beta(0.5), beta(0.75)
```

```
predict(model, type = "beta", p = c(0.25,0.5, 0.75))

# predict the CDF and the PDF at new values of x and y
predict(model, type = "CDF",
        newdata = data.frame(X1=runif(3), y = c(1,2,3)))

# computes the quantile function at new x, for p = (0.25,0.5,0.75)
predict(model, type = "QF", p = c(0.25,0.5,0.75),
        newdata = data.frame(X1=runif(3), y = c(1,2,3)))

# simulate data from the fitted model
ysim <- predict(model, type = "sim") # 'newdata' can be supplied

# if the model is correct, the distribution of y and that of ysim should be similar
qy <- quantile(y, prob = seq(.1,.9,.1))
qsim <- quantile(ysim, prob = seq(.1,.9,.1))
plot(qy, qsim); abline(0,1)
```

---

predict.piqr *Prediction After Penalized Quantile Regression Coefficients Modeling*

Description

Predictions from an object of class "piqr", after selecting the best tuning parameter.

Usage

```
## S3 method for class 'piqr'
predict(object, pos.lambda, type=c("beta", "CDF", "QF", "sim"), newdata,
                          p, se=TRUE, ...)
```

Arguments

object	an object of class "piqr", the result of a call to piqr.
pos.lambda	the positiion of a lambda in the sequence of the object of class "piqr". Could be the best after selecting the result of a call to gof.piqr
type	a character string specifying the type of prediction. See 'Details'.
newdata	an optional data frame in which to look for variables with which to predict. If omitted, the data are used. For type = "CDF", it must include the response variable. Ignored if type = "beta".
p	a numeric vector indicating the order(s) of the quantile to predict. Only used if type = "beta" or type = "QF".
se	logical. If TRUE (the default), standard errors of the prediction will be computed. Only used if type = "beta" or type = "QF".
...	for future methods.

Details

If the best lambda or one value of lambda is chosen, the function call `predict.iqr`.

Value

See details in `predict.iqr`

Note

Prediction may generate quantile crossing if the support of the new covariates values supplied in `newdata` is different from that of the observed data.

Author(s)

Gianluca Sottile <gianluca.sottile@unipa.it>

See Also

`piqr`, for model fitting; `gof.piqr`, to find the best lambda value; `summary.piqr` and `plot.piqr`, for summarizing and plotting `piqr` objects.

Examples

```
# using simulated data

set.seed(1234)
n <- 300
x1 <- rexp(n)
x2 <- runif(n, 0, 5)
x <- cbind(x1,x2)

b <- function(p){matrix(cbind(1, qnorm(p), slp(p, 2)), nrow=4, byrow=TRUE)}
theta <- matrix(0, nrow=3, ncol=4); theta[, 1] <- 1; theta[1,2] <- 1; theta[2:3,3] <- 2
qy <- function(p, theta, b, x){rowSums(x * t(theta %*% b(p)))}

y <- qy(runif(n), theta, b, cbind(1, x))

s <- matrix(1, nrow=3, ncol=4); s[1,3:4] <- 0
obj <- piqr(y ~ x1 + x2, formula.p = ~ I(qnorm(p)) + slp(p, 2), s=s, nlambda=50)

best <- gof.piqr(obj, method="AIC", plot=FALSE)

# predict beta(0.25), beta(0.5), beta(0.75)
predict(obj, best$posMinLambda, type = "beta", p = c(0.25,0.5, 0.75))

# predict the CDF and the PDF at new values of x and y
predict(obj, best$posMinLambda, type = "CDF",
        newdata = data.frame(x1=rexp(3), x2=runif(3), y = c(1,2,3)))

# computes the quantile function at new x, for p = (0.25,0.5,0.75)
```

```
predict(obj, best$posMinLambda, type = "QF", p = c(0.25,0.5,0.75),
        newdata = data.frame(x1=rexp(3), x2=runif(3), y = c(1,2,3)))

# simulate data from the fitted model
ysim <- predict(obj, best$posMinLambda, type = "sim") # 'newdata' can be supplied

# if the model is correct, the distribution of y and that of ysim should be similar
qy <- quantile(y, prob = seq(.1,.9,.1))
qsim <- quantile(ysim, prob = seq(.1,.9,.1))
plot(qy, qsim); abline(0,1)
```

`summary.niqr` *Summary After Nonlinear Quantile Regression Coefficients Modeling*

Description

Summary of an object of class “`niqr`”.

Usage

```
## S3 method for class 'niqr'
summary(object, p, ...)
```

Arguments

`object`	an object of class “`niqr`”, the result of a call to `niqr`.
`p`	an optional vector of quantiles.
`...`	for future methods.

Details

A summary of the model is printed.

Author(s)

Gianluca Sottile <`gianluca.sottile@unipa.it`>

See Also

`niqr`, for model fitting; `testfit.niqr`, for goodness of fit test; `predict.niqr` and `plot.niqr`, for predicting and plotting objects of class “`niqr`”.

Examples

```
n <- 300
x <- runif(n)
fun <- function(theta, p){
  beta0 <- theta[1] + exp(theta[2]*p)
  beta1 <- theta[3] + theta[4]*p
  cbind(beta0, beta1)}
beta <- fun(c(1,1,1,1), runif(n))
y <- beta[, 1] + beta[, 2]*x
model <- niqr(fun=fun, x0=rep(0, 4), X=cbind(1,x), y=y)

summary(model)
summary(model, p=c(.01,.05))
```

`summary.piqr` *Summary After Penalized Quantile Regression Coefficients Modeling*

Description

Summary of an object of class "`piqr`", after selecting the best tuning parameter.

Usage

```
## S3 method for class 'piqr'
summary(object, pos.lambda, SE=FALSE, p, cov=FALSE, ...)
```

Arguments

`object`	an object of class "`piqr`", the result of a call to `piqr`.
`pos.lambda`	the position of a lambda in the sequence of the object of class "`piqr`". Could be the best after selecting the result of a call to `gof.piqr`
`SE`	if TRUE standard errors are printed. Standard errors are computed through sandwich formula only for the regularized parameters.
`p`	an optional vector of quantiles.
`cov`	ff TRUE, the covariance matrix of $\beta(p)$ is reported. Ignored if p is missing.
`...`	for future methods.

Details

If the best lambda or one value of lambda is chosen a summary of the selected model is printed.

Value

See details in `summary.iqr`

Author(s)

Gianluca Sottile <gianluca.sottile@unipa.it>

See Also

piqr, for model fitting; gof.piqr, to find the best lambda value; predict.piqr and plot.piqr, for predicting and plotting objects of class "piqr".

Examples

```
# using simulated data

set.seed(1234)
n <- 300
x1 <- rexp(n)
x2 <- runif(n, 0, 5)
x <- cbind(x1,x2)

b <- function(p){matrix(cbind(1, qnorm(p), slp(p, 2)), nrow=4, byrow=TRUE)}
theta <- matrix(0, nrow=3, ncol=4); theta[, 1] <- 1; theta[1,2] <- 1; theta[2:3,3] <- 2
qy <- function(p, theta, b, x){rowSums(x * t(theta %*% b(p)))}

y <- qy(runif(n), theta, b, cbind(1, x))

s <- matrix(1, nrow=3, ncol=4); s[1,3:4] <- 0
obj <- piqr(y ~ x1 + x2, formula.p = ~ I(qnorm(p)) + slp(p, 2), s=s, nlambda=50)

best <- gof.piqr(obj, method="AIC", plot=FALSE)
best2 <- gof.piqr(obj, method="BIC", plot=FALSE)

summary(obj, best$posMinLambda)
summary(obj, best2$posMinLambda)
```

---

testfit.niqr *Goodness-of-Fit Test*

Description

Goodness-of-fit test for a model fitted with niqr. The Kolmogorov-Smirnov statistic and the Cramer-Von Mises statistic are computed. Their distribution under the null hypothesis is estimated with Monte Carlo (see 'Details').

Usage

```
testfit.niqr(obj, R = 100)
```

Arguments

`obj`	an object of class "`niqr`".
`R`	number of Monte Carlo replications.

Details

This function permits assessing goodness of fit by testing the null hypothesis that the CDF values follow a $U(0, 1)$ distribution, indicating that the model is correctly specified. Since the CDF values depend on estimated parameters, the distribution of the test statistic is not known. To evaluate it, the model is fitted on `R` simulated datasets generated under the null hypothesis.

Value

a matrix with columns `statistic` and `p.value`, reporting the Kolmogorov-Smirnov and Cramer-Von Mises statistic and the associated p-values evaluated with Monte Carlo.

Author(s)

Gianluca Sottile `<gianluca.sottile@unipa.it>`

References

Frumento, P., and Bottai, M. (2015). *Parametric modeling of quantile regression coefficient functions*. Biometrics, doi: 10.1111/biom.12410.

Examples

```
n <- 300
x <- runif(n)
fun <- function(theta, p){
  beta0 <- theta[1] + exp(theta[2]*p)
  beta1 <- theta[3] + theta[4]*p
  cbind(beta0, beta1)}
beta <- fun(c(1,1,1,1), runif(n))
y <- beta[, 1] + beta[, 2]*x
model <- niqr(fun=fun, x0=rep(0, 4), X=cbind(1,x), y=y)
## Not run: testfit.niqr(model, R=100)
```

Index