

# Package ‘quantreg’

November 5, 2009

**Title** Quantile Regression

**Description** Quantile regression and related methods.

**Version** 4.44

**Date** 2009-9-5

**Author** Roger Koenker <rkoenker@uiuc.edu>

**Maintainer** Roger Koenker <rkoenker@uiuc.edu>

**Depends** R (>= 2.6), stats, SparseM

**Suggests** tripack, akima, MASS, survival, rgl, logspline

**License** GPL (>= 2)

**URL** <http://www.r-project.org>

**Repository** CRAN

**Date/Publication** 2009-11-05 12:35:18

## R topics documented:

akj . . . . .	3
anova.rq . . . . .	4
bandwidth.rq . . . . .	7
barro . . . . .	8
boot.crq . . . . .	9
boot.rq . . . . .	10
Bosco . . . . .	11
CobarOre . . . . .	12
combos . . . . .	13
critval . . . . .	14
crq . . . . .	15
dither . . . . .	19
engel . . . . .	20

FAQ	21
KhmaladzeTest	22
kuantile	23
latex	24
latex.summary.rqs	25
latex.table	26
lm.fit.recursive	28
lprq	28
Mammals	30
nlrq	31
nlrq.control	33
Peirce	34
plot.rq	35
plot.rqs	36
plot.rqss	37
plot.summary.rqs	39
predict.rq	41
predict.rqss	43
print.rq	44
print.summary.rq	45
qss	45
QTECox	47
ranks	48
rearrange	49
residuals.nlrq	50
rq	51
rq.fit	54
rq.fit.br	55
rq.fit.fnb	56
rq.fit.fnc	57
rq.fit.lasso	58
rq.fit.pfn	60
rq.fit.scad	61
rq.fit.sfn	62
rq.fit.sfnc	64
rq.object	66
rq.process.object	67
rq.wfit	69
rqProcess	69
rqss	70
rqss.object	72
summary.crq	74
summary.rq	76
summary.rqss	78
table.rq	80
uis	80

**Description**

Univariate *adaptive* kernel density estimation a la Silverman. As used by Portnoy and Koenker (1989).

**Usage**

```
akj(x, z =, p =, h = -1, alpha = 0.5, kappa = 0.9, iker1 = 0)
```

**Arguments**

x	points used for centers of kernel assumed to be sorted.
z	points at which density is calculated; defaults to an equispaced sequence covering the range of x.
p	vector of probabilities associated with xs; defaults to 1/n for each x.
h	initial window size (overall); defaults to Silverman's normal reference.
alpha	a sensitivity parameter that determines the sensitivity of the local bandwidth to variations in the pilot density; defaults to .5.
kappa	constant multiplier for initial (default) window width
iker1	integer kernel indicator: 0 for normal kernel (default) while 1 for Cauchy kernel ( <a href="#">dcauchy</a> ).

**Value**

a [list](#) structure is with components

dens	the vector of estimated density values $f(z)$
psi	a vector of $\psi = -f'/f$ function values.
score	a vector of score $\psi' = (f'/f)^2 - f''/f$ function values.
h	same as the input argument h

**Note**

if the `score` function values are of interest, the Cauchy kernel may be preferable.

**References**

Portnoy, S and R Koenker, (1989) Adaptive L Estimation of Linear Models; *Annals of Statistics* **17**, 362–81.  
 Silverman, B. (1986) *Density Estimation*, pp 100–104.

**Examples**

```

set.seed(1)
x <- c(rnorm(600), 2 + 2*rnorm(400))
xx <- seq(-5, 8, length=200)
z <- akj(x, xx)
plot(xx, z$dens, ylim=range(0,z$dens), type="l", col=2)
abline(h=0, col="gray", lty=3)
plot(xx, z$psi, type="l", col=2, main = expression(hat(psi(x))))
plot(xx, z$score, type="l", col=2,
      main = expression("score " * hat(psi) * "" * (x)))

if(require("norlmix")) {
  m3 <- norMix(mu= c(-4, 0, 3), sig2 = c(1/3^2, 1, 2^2),
              w = c(.1, .5, .4))
  plot(m3, p.norm = FALSE)
  set.seed(11)
  x <- rnorMix(1000, m3)
  z2 <- akj(x, xx)
  lines(xx, z2$dens, col=2)
  z3 <- akj(x, xx, kappa = 0.5, alpha = 0.88)
  lines(xx, z3$dens, col=3)
}

```

anova.rq

*Anova function for quantile regression fits***Description**

Compute test statistics for two or more quantile regression fits.

**Usage**

```

## S3 method for class 'rq':
anova(object, ...)
## S3 method for class 'rqlist':
anova(object, ..., test = "Wald", joint = TRUE,
score = "wilcoxon", R = 200, trim = NULL)
rq.test.rank(x0, x1, y, v = NULL, score = "wilcoxon", weights = NULL, tau=.5,
             iid = TRUE, delta0 = rep(0, NCOL(x1)), omega = 1, trim = NULL, pvalue = "F")
rq.test.anovar(x0, x1, y, tau, R)
## S3 method for class 'anova.rq':
print(x, ...)

```

**Arguments**

`object, ...` objects of class 'rq', originating from a call to 'rq'.

`test` A character string specifying the test statistic to use. Can be either 'Wald' or 'rank'.

joint	A logical flag indicating whether tests of equality of slopes should be done as joint tests on all slope parameters, or whether (when joint = FALSE) separate tests on each of the slope parameters should be reported. This option applies only to the tests of equality of slopes in the case that estimated models correspond to distinct taus.
score	A character string specifying the score function to use, only needed or applicable for the 'rank' form of the test.
trim	optional trimming proportion parameter(s) – only applicable for the Wilcoxon score function – when one value is provided there is symmetric trimming of the score integral to the interval (trim, 1-trim), when there are two values provided, then the trimming restricts the integration to (trim[1], trim[2]).
x	objects of class 'summary.rq', originating from a call to 'summary'.
x0	design matrix for the null component of the rank and anowar tests.
x1	design matrix for the alternative component of the rank and anowar tests.
y	response vector for the alternative component of the rank and anowar tests.
v	optional rq process fit
tau	quantile of interest for quantile specific forms of testing.
iid	logical flag for quantile specific forms of testing, if TRUE the test presumes that the conditional densities take identical values, if it is FALSE then local densities are estimated and used, see Koenker(2005) p. 90.
delta0	vector of hypothetical parameter values under test, typically zeros but can be specified to be nonzero in cases where simulations are being used to evaluate the validity of the non-central chisquare theory of the test.
omega	value to be used for the score and F dependent constant appearing in the non-centrality parameter, this is only needed/useful when delta0 is specified to be non-zero. In the usual Wilcoxon (untrimmed) case this value is the integral the squared density.
pvalue	type of p-value to be used, by default a pseudo F-statistic is produced and the corresponding F p-value is computed, otherwise the more conventional chisquared p-values are reported.
weights	optional weight vector to be used for fitting.
R	The number of resampling replications for the anowar form of the test, used to estimate the reference distribution for the test statistic.

## Details

There are two (as yet) distinct forms of the test. In the first the fitted objects all have the same specified quantile (tau) and the intent is to test the hypothesis that smaller models are adequate relative to the largest specified model. In the second form of the test the linear predictor of the fits are all the same, but the specified quantiles (taus) are different.

In the former case there are three options for the argument 'test', by default a Wald test is computed as in Bassett and Koenker (1982). If `test = 'anowar'` is specified then the test is based on the procedure suggested in Chen, Ying, Zhang and Zhao (2008); the test is based on the difference in the QR objective functions at the restricted and unrestricted models with a reference distribution

computed by simulation. The p-value of this form of the test is produced by fitting a density to the simulation values forming the reference distribution using the `logspline` function from the **logspline** package. The acronym `anovar` stands for analysis of weighted absolute residuals. If `test='rank'` is specified, then a rank test statistic is computed as described in Gutenbrunner, Jureckova, Koenker and Portnoy (1993). In the latter case one can also specify a form for the score function of the rank test, by default the Wilcoxon score is used, the other options are `score='sign'` for median (sign) scores, or `score='normal'` for normal (van der Waerden) scores. A fourth option is `score='tau'` which is a generalization of median scores to an arbitrary quantile, in this case the quantile is assumed to be the one associated with the fitting of the specified objects. The computing of the rank form of the test is carried out in the `rq.test.rank` function, see `ranks` for further details on the score function options. The Wald form of the test is local in sense that the null hypothesis asserts only that a subset of the covariates are “insignificant” at the specified quantile of interest. The rank form of the test can also be used to test the global hypothesis that a subset is “insignificant” over an entire range of quantiles. The use of the score function `score = "tau"` restricts the rank test to the local hypothesis of the Wald test.

In the latter case the hypothesis of interest is that the slope coefficients of the models are identical. The test statistic is a variant of the Wald test described in Koenker and Bassett (1982).

By default, both forms of the tests return an F-like statistic in the sense that the an asymptotically Chi-squared statistic is divided by its degrees of freedom and the reported p-value is computed for an F statistic based on the numerator degrees of freedom equal to the rank of the null hypothesis and the denominator degrees of freedom is taken to be the sample size minus the number of parameters of the maintained model.

### Value

An object of class “anova” inheriting from class “data.frame”.

### WARNING

An attempt to verify that the models are nested in the first form of the test is made, but this relies on checking set inclusion of the list of variable names and is subject to obvious ambiguities when variable names are generic. The comparison between two or more models will only be valid if they are fitted to the same dataset. This may be a problem if there are missing values and R’s default of `'na.action = na.omit'` is used. The rank version of the nested model tests involves computing the entire regression quantile process using parametric linear programming and thus can be rather slow and memory intensive on problems with more than several thousand observations.

### Author(s)

Roger Koenker

### References

- [1] Bassett, G. and R. Koenker (1982). Tests of Linear Hypotheses and L1 Estimation, *Econometrica*, **50**, 1577–83.
- [2] Koenker, R. W. and Bassett, G. W. (1982). Robust Tests for Heteroscedasticity based on Regression Quantiles, *Econometrica*, **50**, 43–61.

- [3] Gutenbrunner, C., Jureckova, J., Koenker, R., and S. Portnoy (1993). Tests of Linear Hypotheses based on Regression Rank Scores, *J. of Nonparametric Statistics*, **2**, 307–331.
- [4] Chen, K. Z. Ying, H. Zhang, and L Zhao, (2008) Analysis of least absolute deviations, *Biometrika*, **95**, 107-122.
- [5] Koenker, R. W. (2005). *Quantile Regression*, Cambridge U. Press.

### See Also

The model fitting function `rq`, and the functions for testing hypothesis on the entire quantile regression process `KhmaladzeTest`. For further details on the rank tests see `ranks`.

### Examples

```
data(barro)
fit0 <- rq(y.net ~ lgdp2 + fse2 + gedy2 , data = barro)
fit1 <- rq(y.net ~ lgdp2 + fse2 + gedy2 + Iy2 + gcony2, data = barro)
fit2 <- rq(y.net ~ lgdp2 + fse2 + gedy2 + Iy2 + gcony2, data = barro, tau=.75)
fit3 <- rq(y.net ~ lgdp2 + fse2 + gedy2 + Iy2 + gcony2, data = barro, tau=.25)
anova(fit1, fit0)
anova(fit1, fit2, fit3)
anova(fit1, fit2, fit3, joint=FALSE)
```

---

bandwidth.rq

*bandwidth selection for rq functions*

---

### Description

function to compute bandwidth for sparsity estimation

### Usage

```
bandwidth.rq(p, n, hs=TRUE, alpha=0.05)
```

### Arguments

<code>p</code>	quantile(s) of interest
<code>n</code>	sample size
<code>hs</code>	flag for hall-sheather method
<code>alpha</code>	alpha level for intended confidence intervals

### Details

If `hs=TRUE` (default) then the Hall-Sheather(1988) rule  $O(n^{-1/3})$  is used, if `hs=FALSE` then the Bofinger  $O(n^{-1/5})$  is used.

### Value

returns a vector of bandwidths corresponding to the argument `p`.

**Author(s)**

Roger Koenker rkoenker@uiuc.edu

**References**

Hall and Sheather(1988, JRSS(B)),Bofinger (1975, Aus. J. Stat)

---

barro

*Barro Data*

---

**Description**

Version of the Barro Growth Data used in Koenker and Machado(1999). This is a regression data set consisting of 161 observations on determinants of cross country GDP growth rates. There are 13 covariates with dimnames corresponding to the original Barro and Lee source. See <http://www.nber.org/pub/barro.lee/>. The first 71 observations are on the period 1965-75, remainder on 1987-85.

**Usage**

data(barro)

**Format**

A data frame containing 161 observations on 14 variables:

[,1]	"Annual Change Per Capita GDP"
[,2]	"Initial Per Capita GDP"
[,3]	"Male Secondary Education"
[,4]	"Female Secondary Education"
[,5]	"Female Higher Education"
[,6]	"Male Higher Education"
[,7]	"Life Expectancy"
[,8]	"Human Capital"
[,9]	"Education/GDP"
[,10]	"Investment/GDP"
[,11]	"Public Consumption/GDP"
[,12]	"Black Market Premium"
[,13]	"Political Instability"
[,14]	"Growth Rate Terms Trade"

**References**

Koenker, R. and J.A.F. Machado (1999) Goodness of Fit and Related Inference Processes for Quantile Regression, JASA, 1296-1310.

boot.crq

*Bootstrapping Censored Quantile Regression***Description**

Functions used to estimated standard errors, confidence intervals and tests of hypotheses for censored quantile regression models using the Portnoy and Peng-Huang methods.

**Usage**

```
boot.crq(x, y, c, taus, method, R = 100, mboot, bmethod = "xy-pair", ...)
```

**Arguments**

x	The regression design matrix
y	The regression response vector
c	The censoring indicator
taus	The quantiles of interest
method	The fitting method: either "P" for Portnoy or "PH" for Peng and Huang.
R	The number of bootstrap replications
bmethod	The method to be employed. There are (as yet) two options: method = "xy-pair" uses the xy-pair method, that is the usual multinomial resampling of xy-pairs, while method = "Bose" uses the Bose and Chatterjee (2003) weighted resampling method with exponential weights.
mboot	optional argument for the bootstrap method "xy-pair" that permits subsampling (m out of n) bootstrap. Obviously mboot should be substantially larger than the column dimension of x, and should be less than the sample size.
...	Optional further arguments to control bootstrapping

**Details**

There are several refinements that are still unimplemented. Percentile methods should be incorporated, and extensions of the methods to be used in anova.rq should be made. Note that bootstrapping for the Powell method "FP" is done via [boot.rq](#).

**Value**

A matrix of dimension R by p is returned with the R resampled estimates of the vector of quantile regression parameters. When  $m < n$  for the "xy" method this matrix has been deflated by the factor  $\sqrt{m/n}$

**Author(s)**

Roger Koenker

## References

Bose, A. and S. Chatterjee, (2003) Generalized bootstrap for estimators of minimizers of convex functions, *J. Stat. Planning and Inf*, 117, 225-239.

## See Also

[summary.crq](#)

---

boot.rq

*Bootstrapping Quantile Regression*

---

## Description

These functions can be used to construct standard errors, confidence intervals and tests of hypotheses regarding quantile regression models.

## Usage

```
boot.rq(x, y, tau = 0.5, R = 200, bsmethod = "xy", mofn = length(y), ...)
```

## Arguments

x	The regression design matrix
y	The regression response vector
tau	The quantile of interest
R	The number of bootstrap replications
bsmethod	The method to be employed. There are (as yet) four options: method = "xy" uses the xy-pair method, and method = "pwy" uses the method of Parzen, Wei and Ying (1994) method = "mcmb" uses the Markov chain marginal bootstrap of He and Hu (2002) and Kocherginsky, He and Mu (2003). The fourth method = "wxy" uses the generalized bootstrap of Bose and Chatterjee (2003) with unit exponential weights, see also Chamberlain and Imbens (2003).
mofn	optional argument for the bootstrap method "xy" that permits subsampling (m out of n) bootstrap. Obviously mofn should be substantially larger than the column dimension of x, and should be less than the sample size.
...	Optional arguments to control bootstrapping

## Details

There are several refinements that are still unimplemented. Percentile methods should be incorporated, and extensions of the methods to be used in anova.rq should be made.

## Value

A matrix of dimension R by p is returned with the R resampled estimates of the vector of quantile regression parameters. When mofn < n for the "xy" method this matrix has been deflated by the factor  $\sqrt{m/n}$

**Author(s)**

Roger Koenker (and Xuming He and M. Kocherginsky for the mcmb code)

**References**

- [1] Koenker, R. W. (1994). Confidence Intervals for regression quantiles, in P. Mandl and M. Huskova (eds.), *Asymptotic Statistics*, 349–359, Springer-Verlag, New York.
- [2] Kocherginsky, M., He, X. and Mu, Y. (2005). Practical Confidence Intervals for Regression Quantiles, *Journal of Computational and Graphical Statistics*, 14, 41-55.
- [3] He, X. and Hu, F. (2002). Markov Chain Marginal Bootstrap. *Journal of the American Statistical Association*, Vol. 97, no. 459, 783-795.
- [4] Parzen, M. I., L. Wei, and Z. Ying (1994): A resampling method based on pivotal estimating functions,” *Biometrika*, 81, 341–350.
- [5] Bose, A. and S. Chatterjee, (2003) Generalized bootstrap for estimators of minimizers of convex functions, *J. Stat. Planning and Inf*, 117, 225-239.
- [6] Chamberlain G. and Imbens G.W. (2003) Nonparametric Applications of Bayesian Inference, *Journal of Business & Economic Statistics*, 21, pp. 12-18.

**See Also**

[summary.rq](#)

**Examples**

```
y <- rnorm(50)
x <- matrix(rnorm(100), 50)
fit <- rq(y~x, tau = .4)
summary(fit, se = "boot", bsmethod= "xy")
summary(fit, se = "boot", bsmethod= "pwy")
#summary(fit, se = "boot", bsmethod= "mcmb")
```

---

Bosco

*Boscovich Data*

---

**Description**

Boscovich data used to estimate the ellipticity of the earth. There are five measurements of the arc length of one degree of latitude taken at 5 different latitudes. See Koenker (2005) for further details and references.

**Usage**

```
data(Bosco)
```

**Format**

A data frame containing 5 observations on 2 variables

**x** sine squared of latitude measured in degrees

**y** arc length of one degree of latitude measured in toise - 56,700, one toise approximately equals 1.95 meters.

**References**

Koenker, R. (2005), "Quantile Regression", Cambridge.

**Examples**

```
data(Bosco)
plot(0:10/10,0:10*100,xlab="sin^2(latitude)",
     ylab="arc-length of 1 degree of latitude",type="n")
points(Bosco)
text(Bosco, pos = 3, rownames(Bosco))
z <- rq(y ~ x, tau = -1, data = Bosco)
title("Boscovitch Ellipticity of the Earth Example")
xb <- c(.85, .9, .6, .6)
yb <- c(400,600,450,600)
for(i in 1:4){
  abline(c(z$sol[4:5,i]))
  interval <- paste("t=", format(round(z$sol[1,i],2)), ", ",
                   format(round(z$sol[1,i+1],2)), ") ", delim="")
  text(xb[i], yb[i], interval)
}
```

---

CobarOre

*Cobar Ore data*

---

**Description**

Cobar Ore data from Green and Silverman (1994). The data consists of measurements on the "true width" of an ore-bearing rock layer from a mine in Cobar, Australia.

**Usage**

```
data(CobarOre)
```

**Format**

A data frame with 38 observations on the following 3 variables.

**x** x-coordinate of location of mine site

**y** y-coordinate of location of mine site

**z** ore thickness

**Source**

Green, P.J. and B.W. Silverman (1994) Nonparametric Regression Generalized Linear Models: A roughness penalty approach, Chapman Hall.

**Examples**

```
data (CobarOre)
plot (CobarOre)
```

---

`combos`*Ordered Combinations*

---

**Description**

All  $m$  combinations of the first  $n$  integers taken  $p$  at a time are computed and return as an  $p$  by  $m$  matrix. The columns of the matrix are ordered so that adjacent columns differ by only one element. This is just a reordered version of `combn` in base R, but the ordering is useful for some applications.

**Usage**

```
combos (n, p)
```

**Arguments**

<code>n</code>	The $n$ in $n$ choose $p$
<code>p</code>	The $p$ in $n$ choose $p$

**Value**

a matrix of dimension  $p$  by `choose (n, p)`

**Note**

Implementation based on a Pascal algorithm of Limin Xiang and Kazuo Ushijima (2001) translated to `ratfor` for R. If you have **rgl** installed you might try `demo ("combos")` for a visual impression of how this works.

**References**

Limin Xiang and Kazuo Ushijima (2001) "On  $O(1)$  Time Algorithms for Combinatorial Generation," *Computer Journal*, 44(4), 292-302.

**Examples**

```
H <- combos (20, 3)
```

---

`critval`*Hotelling Critical Values*

---

**Description**

Critical values for uniform confidence bands for rqss fitting

**Usage**

```
critval(kappa, alpha = 0.05, rdf = 0)
```

**Arguments**

<code>kappa</code>	length of the tube
<code>alpha</code>	desired non-coverage of the band, intended coverage is 1 - alpha
<code>rdf</code>	"residual" degrees of freedom of the fitted object. If <code>rdf=0</code> then the Gaussian version of the critical value is computed, otherwise the value is based on standard Student t theory.

**Details**

The Hotelling tube approach to inference has a long and illustrious history. See Johansen and Johnstone (1989) for an overview. The implementation here is based on Sun and Loader (1994) and Loader's **locfit** package, although a simpler root finding approach is substituted for the iterative method used there. At this stage, only univariate bands may be constructed.

**Value**

A scalar critical value that acts as a multiplier for the uniform confidence band construction.

**References**

- Hotelling, H. (1939): "Tubes and Spheres in  $n$ -spaces, and a class of statistical problems," *Am J. Math*, 61, 440–460.
- Johansen, S., I.M. Johnstone (1990): "Hotelling's Theorem on the Volume of Tubes: Some Illustrations in Simultaneous Inference and Data Analysis," *The Annals of Statistics*, 18, 652–684.
- Sun, J. and C.V. Loader: (1994) "Simultaneous Confidence Bands for Linear Regression and smoothing," *The Annals of Statistics*, 22, 1328–1345.

**See Also**

[plot.rqss](#)

**Description**

Fits a conditional quantile regression model for censored data. There are three distinct methods: the first is the fixed censoring method of Powell (1986) as implemented by Fitzenberger (1996), the second is the random censoring method of Portnoy (2003). The third method is based on Peng and Huang (2008).

**Usage**

```
crq(formula, taus, data, subset, weights, na.action, method = c("Powell", "Portnoy")
crq.fit.pow(x, y, yc, tau=0.5, weights=NULL, start, left=TRUE, maxit = 500)
crq.fit.pen(x, y, cen, weights=NULL, grid)
crq.fit.por(x, y, cen, weights=NULL, grid)
Curv(y, yc, type=c("left", "right"))
## S3 method for class 'crq':
print(x, ...)
## S3 method for class 'crq':
print(x, ...)
## S3 method for class 'crq':
predict(object, newdata, ...)
## S3 method for class 'crqs':
predict(object, newdata, type = NULL, ...)
## S3 method for class 'crq':
coef(object, taus = 1:4/5, ...)
```

**Arguments**

formula	A formula object, with the response on the left of the '~' operator, and the terms on the right. The response must be a <code>Surv</code> object as returned by either the <code>Curv</code> or <code>Surv</code> function. For the Powell method, the <code>Surv</code> object should be created by <code>Curv</code> and have arguments (event time, censoring time, type), where "type" can take values either "left" or "right". The default (for historical reasons) for type in this case is "left". For the Portnoy and Peng and Huang methods the <code>Surv</code> should be created with the usual <code>Surv</code> function and have (event time, censoring indicator).
y	The event time.
newdata	An optional data frame in which to look for variables with which to predict. If omitted, the fitted values are used.
grid	A vector of taus on which the quantile process should be evaluated. This should be monotonic, and take values in (0,1). For the "Portnoy" method, grid = "pivot" computes the full solution for all distinct taus.
x	An object of class <code>crq</code> or <code>crqs</code> .

<code>object</code>	An object of class <code>crq</code> or <code>crq</code> .
<code>yc</code>	The censoring times for the "Powell" method.
<code>type</code>	Censoring type for the "Powell" method, used in <code>Curv</code> , by default "left". If you don't like "left", maybe you will like "right". Note that for fixed censoring assumed in the "Powell" method, censoring times <code>yc</code> must be provided for all observations and the event times <code>y</code> must satisfy the (respective) inequality constraints.
<code>cen</code>	The censoring indicator for the "Portnoy" and "PengHuang" methods.
<code>maxit</code>	Maximum number of iterations allowed for the "Powell" methods.
<code>start</code>	The starting value for the coefs for the "Powell" method. Because the Fitzenberger algorithm stops when it achieves a local minimum of the Powell objective function, the starting value acts as an a priori "preferred point". This is advantageous in some instances since the global Powell solution can be quite extreme. By default the starting value is the "naive rq" solution that treats all the censored observations as uncensored. If <code>start</code> is equal to "global" then an attempt is made to compute to global optimum of the Powell objective. This entails an exhaustive evaluation of all $n$ choose $p$ distinct basic solution so is rather impractical for moderately large problems. Otherwise, the starting value can specify a set of $p$ indices from $1:n$ defining an initial basic solution, or it may specify a $p$ -vector of initial regression coefficients. In the latter case the initial basic solution is the one closest to the specified parameter vector.
<code>left</code>	A logical indicator for left censoring for the "Powell" method.
<code>taus</code>	The quantile(s) at which the model is to be estimated.
<code>tau</code>	The quantile at which the model is to be estimated.
<code>data</code>	A <code>data.frame</code> in which to interpret the variables named in the 'formula', in the 'subset', and the 'weights' argument.
<code>subset</code>	an optional vector specifying a subset of observations to be used in the fitting process.
<code>weights</code>	vector of observation weights; if supplied, the algorithm fits to minimize the sum of the weights multiplied into the absolute residuals. The length of weights vector must be the same as the number of observations. The weights must be nonnegative and it is strongly recommended that they be strictly positive, since zero weights are ambiguous.
<code>na.action</code>	a function to filter missing data. This is applied to the <code>model.frame</code> after any subset argument has been used. The default (with 'na.fail') is to create an error if any missing values are found. A possible alternative is 'na.omit', which deletes observations that contain one or more missing values.
<code>method</code>	The method used for fitting. There are currently two options: method "Powell" computes the Powell estimator using the algorithm of Fitzenberger (1996), method "Portnoy" computes the Portnoy (2003) estimator. The method is "PengHuang" uses the method of Peng and Huang (2007), in this case the variable "grid" can be passed to specify the vector of quantiles at which the solution is desired.
<code>contrasts</code>	a list giving contrasts for some or all of the factors default = 'NULL' appearing in the model formula. The elements of the list should have the same name as

the variable and should be either a contrast matrix (specifically, any full-rank matrix with as many rows as there are levels in the factor), or else a function to compute such a matrix given the number of levels.

... additional arguments for the fitting routine, for method "Powell" it may be useful to pass starting values of the regression parameter via the argument "start", while for methods "Portnoy" or "PengHuang" one may wish to specify an alternative to the default grid for evaluating the fit.

## Details

The Fitzenberger algorithm uses a variant of the Barrodale and Roberts simplex method. Exploiting the fact that the solution must be characterized by an exact fit to  $p$  points when there are  $p$  parameters to be estimated, at any trial basic solution it computes the directional derivatives in the  $2p$  distinct directions and chooses the direction that (locally) gives steepest descent. It then performs a one-dimensional line search to choose the new basic observation and continues until it reaches a local minimum. By default it starts at the naive `rq` solution ignoring the censoring; this has the (slight) advantage that the estimator is consequently equivariant to canonical transformations of the data. Since the objective function is no longer convex there can be no guarantee that this produces a global minimum estimate. In small problems exhaustive search over solutions defined by  $p$ -element subsets of the  $n$  observations can be used, but this quickly becomes impractical for large  $p$  and  $n$ . This global version of the Powell estimator can be invoked by specifying `start = "global"`. Users interested in this option would be well advised to compute `choose(n, p)` for their problems before trying it. The method operates by pivoting through this many distinct solutions and choosing the one that gives the minimal Powell objective. The algorithm used for the Portnoy method is described in considerable detail in Portnoy (2003). Both the Portnoy and Peng-Huang estimators may be unable to compute estimates of the conditional quantile parameters in the upper tail of distribution. Like the Kaplan-Meier estimator, when censoring is heavy in the upper tail the estimated distribution is defective and quantiles are only estimable on a sub-interval of  $(0,1)$ . The Peng and Huang estimator can be viewed as a generalization of the Nelson Aalen estimator of the cumulative hazard function, and can be formulated as a variant of the conventional quantile regression dual problem. See Koenker (2008) for further details. This paper is available from the package with `faux.vignette("crq")`.

## Value

An object of class `crq`.

## Author(s)

Steve Portnoy and Roger Koenker

## References

- Fitzenberger, B. (1996): "A Guide to Censored Quantile Regressions," in *Handbook of Statistics*, ed. by C.~Rao, and G.~Maddala. North-Holland: New York.
- Fitzenberger, B. and P. Winker (2007): "Improving the Computation of Censored Quantile Regression Estimators," *CSDA*, 52, 88-108.
- Koenker, R. (2008): "Censored Quantile Regression Redux," *J. Statistical Software*, 27, <http://www.jstatsoft.org/v27/i06>.

Peng, L and Y Huang, (2008) Survival Analysis with Quantile Regression Models, *J. Am. Stat. Assoc.*, 103, 637-649.

Portnoy, S. (2003) "Censored Quantile Regression," *JASA*, 98,1001-1012.

Powell, J. (1986) "Censored Regression Quantiles," *J. Econometrics*, 32, 143–155.

## See Also

[summary.crq](#)

## Examples

```
# An artificial Powell example
set.seed(2345)
x <- sqrt(rnorm(100)^2)
y <- -0.5 + x + (.25 + .25*x)*rnorm(100)
plot(x,y, type="n")
s <- (y > 0)
points(x[s],y[s],cex=.9,pch=16)
points(x[!s],y[!s],cex=.9,pch=1)
yLatent <- y
y <- pmax(0,y)
yc <- rep(0,100)
for(tau in (1:4)/5){
  f <- crq(Curv(y,yc) ~ x, tau = tau, method = "Pow")
  xs <- sort(x)
  lines(xs,pmax(0,cbind(1,xs)%*%f$coef),col="red")
  abline(rq(y ~ x, tau = tau), col="blue")
  abline(rq(yLatent ~ x, tau = tau), col="green")
}
legend(.15,2.5,c("Naive QR","Censored QR","Omniscient QR"),
      lty=rep(1,3),col=c("blue","red","green"))
data(uis)
#estimate the Peng and Huang model using log(TIME) AFT specification
fit <- crq(Surv(log(TIME), CENSOR) ~ ND1 + ND2 + IV3 +
          TREAT + FRAC + RACE + AGE * SITE, method = "Por", data = uis)
Sfit <- summary(fit,1:19/20)
PHit <- coxph(Surv(TIME, CENSOR) ~ ND1 + ND2 + IV3 +
            TREAT + FRAC + RACE + AGE * SITE, data = uis)
plot(Sfit, CoxPHit = PHit)
formula <- ~ ND1 + ND2 + IV3 + TREAT + FRAC + RACE + AGE * SITE -1
X <- data.frame(model.matrix(formula,data=uis))
newd <- as.list(apply(X,2,median))
pred <- predict(fit, newdata=newd, type = "stepfun")
plot(pred,do.points=FALSE,xlab = expression(tau), ylab = expression(Q(tau)),main= "Quantiles")
plot(rearrange(pred),add=TRUE,do.points=FALSE,col.vert = "red", col.hor="red")
legend(.15,7,c("Raw","Rearranged"),lty = rep(1,2),col=c("black","red"))
```

---

dither                      *Function to randomly perturb a vector*

---

### Description

With malice aforethought, `dither` adds a specified random perturbation to each element of the input vector, usually employed as a device to mitigate the effect of ties.

### Usage

```
dither(x, type = "symmetric", value = NULL)
```

### Arguments

<code>x</code>	<code>x</code> a numeric vector
<code>type</code>	<code>type</code> is either 'symmetric' or 'right'
<code>value</code>	<code>value</code> scale of dequantization

### Details

The function `dither` operates slightly differently than the function `jitter` in base R, permitting strictly positive perturbations with the option `type = "right"` and using somewhat different default schemes for the scale of the perturbation. Dithering the response variable is frequently a useful option in quantile regression fitting to avoid deleterious effects of degenerate solutions. See, e.g. Machado and Santos Silva (2005). For a general introduction and some etymology see the Wikipedia article on "dither". For integer data it is usually advisable to use `value = 1`. When 'x' is a matrix or array `dither` treats all elements as a vector but returns an object of the original class.

### Value

A dithered version of the input vector 'x'.

### Note

Some further generality might be nice, for example something other than uniform noise would be desirable in some circumstances. Note that when dithering you are entering into the "state of sin" that John von Neumann famously attributed to anyone considering "arithmetical methods of producing random digits." If you need to preserve reproducibility, then `set.seed` is your friend.

### Author(s)

R. Koenker

### References

Machado, J.A.F. and Santos Silva, J.M.C. (2005), Quantiles for Counts, *Journal of the American Statistical Association*, vol. 100, no. 472, pp. 1226-1237.

**See Also**[jitter](#)**Examples**

```
x <- rlnorm(40)
y <- rpois(40, exp(.5 + log(x)))
f <- rq(log(dither(y, type = "right", value = 1)) ~ x)
```

---

 engel

*Engel Data*


---

**Description**

Engel food expenditure data used in Koenker and Bassett(1982). This is a regression data set consisting of 235 observations on income and expenditure on food for Belgian working class households.

**Usage**

```
data(engel)
```

**Format**

A data frame containing 235 observations on 2 variables

**income** annual household income in Belgian francs

**foodexp** annual household food expenditure in Belgian francs

**References**

Koenker, R. and Bassett, G (1982) Robust Tests of Heteroscedasticity based on Regression Quantiles; *Econometrica* **50**, 43–61.

**Examples**

```
## See also      demo("engel1")
##      -----

data(engel)
plot(engel, log = "xy",
     main = "'engel' data (log - log scale)")
plot(log10(foodexp) ~ log10(income), data = engel,
     main = "'engel' data (log10 - tranformed)")
taus <- c(.15, .25, .50, .75, .95, .99)
rqs <- as.list(taus)
for(i in seq(along = taus)) {
  rqs[[i]] <- rq(log10(foodexp) ~ log10(income), tau = taus[i], data = engel)
  lines(log10(engel$income), fitted(rqs[[i]]), col = i+1)
```

```

}
legend("bottomright", paste("tau = ", taus), inset = .04,
      col = 2:(length(taus)+1), lty=1)

```

---

FAQ

*FAQ, ChangeLog, or faux.vignettes of a package*


---

## Description

Show the FAQ, ChangeLog or faux.vignettes of a specified package

## Usage

```

FAQ(pkg = "quantreg")
ChangeLog(pkg = "quantreg")
vignette(topic, package = NULL, lib.loc = NULL, all = TRUE)
faux.vignette(topic, pkg = "quantreg")

```

## Arguments

<code>pkg</code>	Package Name
<code>package</code>	Package Name
<code>topic</code>	topic of a vignette
<code>lib.loc</code>	a character vector of directory names of R libraries, or 'NULL'. The default value of 'NULL' corresponds to all libraries currently known.
<code>all</code>	logical; if 'TRUE' search all available packages in the library trees specified by 'lib.loc', and if 'FALSE', search only attached packages.

## Details

Assumes that the FAQ and/or ChangeLog files exist in the proper "inst" directory. The function vignette is a quantreg specific version of the usual utils function that facilitates access to faux vignettes for the quantreg package. Unless the package argument is specified to be "quantreg", the function reverts to the utils version. Faux vignettes are vignette-like pdf files that are not in official Sweave format. A list of faux vignettes for the package can be found by omitting the topic argument.

## Value

Has only the side effect of showing the files on the screen. At present opening vignettes happens on the remote machine if the user is connected via ssh – if someone knows how to make this happen locally, let me know.

## Note

Might be nice to have a more general system version of this but until then, this will have to suffice.

---

KhmaladzeTest	<i>Tests of Location and Location Scale Shift Hypotheses for Linear Models</i>
---------------	--

---

### Description

Tests of the hypothesis that a linear model specification is of the location shift or location-scale shift form. The tests are based on the Doob-Meyer Martingale transformation approach proposed by Khmaladze(1981) for general goodness of fit problems, and adapted to quantile regression by Koenker and Xiao (2002).

### Usage

```
KhmaladzeTest(formula, data = NULL, taus = -1, nullH = "location" , trim = c(0.05,
```

### Arguments

<code>formula</code>	a formula specifying the model to fit by <code>rqProcess</code>
<code>data</code>	a data frame within which to interpret the formula
<code>taus</code>	An equally spaced grid of points on which to evaluate the quantile regression process, if any taus fall outside (0,1) then the full process is computed.
<code>nullH</code>	a character vector indicating whether the "location" shift hypothesis (default) or the "location-scale" shift hypothesis should be tested.
<code>trim</code>	a vector indicating the lower and upper bound of the quantiles to included in the computation of the test statistics (only, not estimates).
<code>...</code>	other arguments to be passed to <code>summary.rq</code> .

### Value

an object of class `KhmaladzeTest` is returned containing:

<code>nullH</code>	The form of the null hypothesis.
<code>Tn</code>	Joint test statistic of the hypothesis that all the slope parameters of the model satisfy the hypothesis.
<code>THn</code>	Vector of test statistics testing whether individual slope parameters satisfy the null hypothesis.

### References

Khmaladze, E. (1981) "Martingale Approach in the Theory of Goodness-of-fit Tests," *Theory of Prob. and its Apps*, 26, 240–257.

Koenker, Roger and Zhijie Xiao (2002), "Inference on the Quantile Regression Process", *Econometrica*, 81, 1583–1612. <http://www.econ.uiuc.edu/~roger/research/inference/inference.html>

**Examples**

```
data(barro)
KhmaldzeTest( y.net ~ lgdp2 + fse2 + gedy2 + Iy2 + gcony2,
data = barro, taus = seq(.05,.95,by = .01))
```

kuantile

*Quicker Sample Quantiles***Description**

The function 'kuantile' computes sample quantiles corresponding to the specified probabilities. The intent is to mimic the generic (base) function 'quantile' but using a variant of the Floyd and Rivest (1975) algorithm which is somewhat quicker, especially for large sample sizes.

**Usage**

```
kuantile(x, probs = seq(0, 1, .25), na.rm = FALSE, names = TRUE, type = 7, ...)
```

**Arguments**

x	numeric vector whose sample quantiles are wanted.
probs	numeric vector of probabilities with values in [0,1].
type	an integer between 1 and 9 selecting one of the nine quantile algorithms detailed below to be used.
na.rm	logical: if true, any 'NA' and 'NaN's are removed from 'x' before the quantiles are computed.
names	logical: if true, the result has a 'names' attribute.
...	further arguments passed to or from other methods.

**Details**

A vector of length 'length(p)' is returned. See the documentation for 'quantile' for further details on the types. The algorithm was written by K.C. Kiwiel. It is a modified version of the (algol 68) SELECT procedure of Floyd and Rivest (1975), incorporating modifications of Brown(1976). The algorithm has linear growth in the number of comparisons required as sample size grows. For the median, average case behavior requires  $1.5n + O((n \log n)^{1/2})$  comparisons. See Kiwiel (2005) and Knuth (1998) for further details. When the number of required elements of p is large, it may be preferable to revert to a full sort.

**Value**

A vector of quantiles of the same length as the vector p.

**Author(s)**

K.C. Kiwiel, R interface: Roger Koenker

## References

R.W. Floyd and R.L. Rivest: "Algorithm 489: The Algorithm SELECT—for Finding the  $i$ th Smallest of  $n$  Elements", *Comm. ACM* 18, 3 (1975) 173,

T. Brown: "Remark on Algorithm 489", *ACM Trans. Math. Software* 3, 2 (1976), 301-304.

K.C. Kiwiel: On Floyd and Rivest's SELECT Algorithm, *Theoretical Computer Sci.* 347 (2005) 214-238.

D. Knuth, *The Art of Computer Programming, Volume 3, Sorting and Searching*, 2nd Ed., (1998), Addison-Wesley.

## See Also

[quantile](#)

## Examples

```
kuantile(x <- rnorm(1001))# Extremes & Quartiles by default

### Compare different types
p <- c(0.1,0.5,1,2,5,10,50)/100
res <- matrix(as.numeric(NA), 9, 7)
for(type in 1:9) res[type, ] <- y <- kuantile(x, p, type=type)
dimnames(res) <- list(1:9, names(y))
ktiles <- res

### Compare different types
p <- c(0.1,0.5,1,2,5,10,50)/100
res <- matrix(as.numeric(NA), 9, 7)
for(type in 1:9) res[type, ] <- y <- quantile(x, p, type=type)
dimnames(res) <- list(1:9, names(y))
qtiles <- res

max(abs(ktiles - qtiles))
```

---

latex

*Make a latex version of an R object*

---

## Description

Generic function for converting an R object into a latex file.

## Usage

```
latex(x, ...)
```

**Arguments**

x                    x is an R object  
 ...                 ... optional arguments

**See Also**

[latex.table](#), [latex.summary.rqs](#)

---

`latex.summary.rqs` *Make a latex table from a table of rq results*

---

**Description**

Produces a file with latex commands for a table of rq results.

**Usage**

```
## S3 method for class 'summary.rqs':
latex(x, transpose = FALSE, caption = "caption goes here.",
      digits = 3, file = as.character(substitute(x)), ...)
```

**Arguments**

x                    x is an object of class `summary.rqs`  
 transpose          if TRUE transpose table so that rows are quantiles and columns are covariates.  
 caption            caption for the table  
 digits             decimal precision of table entries.  
 file                name of file  
 ...                 optional arguments for `latex.table`

**Details**

Calls `latex.table`.

**Value**

Returns invisibly after writing the file.

**Author(s)**

R. Koenker

**See Also**

[summary.rqs](#), [latex.table](#)

---

latex.table	<i>Writes a latex formatted table to a file</i>
-------------	---

---

### Description

Automatically generates a latex formatted table from the matrix `x` Controls rounding, alignment, etc, etc

### Usage

```
## S3 method for class 'table':
latex(x, file=as.character(substitute(x)), rowlabel=file, rowlabel.just="l", cgroup
```

### Arguments

<code>x</code>	A matrix <code>x</code> with dimnames
<code>file</code>	Name of output file (.tex will be added)
<code>rowlabel</code>	If ' <code>x</code> ' has row dimnames, <code>rowlabel</code> is a character string containing the column heading for the row dimnames. The default is the name of the argument for <code>x</code> .
<code>rowlabel.just</code>	If ' <code>x</code> ' has row dimnames, specifies the justification for printing them. Possible values are ' <code>l</code> ', ' <code>r</code> ', ' <code>c</code> '. The heading ( <code>rowlabel</code> ) itself is left justified if ' <code>rowlabel.just="l</code> ', otherwise it is centered.
<code>cgroup</code>	a vector of character strings defining major column headings. The default is to have none.
<code>n.cgroup</code>	a vector containing the number of columns for which each element in <code>cgroup</code> is a heading. For example, specify ' <code>cgroup=c("Major 1","Major 2")</code> ', ' <code>n.cgroup=c(3,3)</code> ' if "Major 1" is to span columns 1-3 and "Major 2" is to span columns 4-6. ' <code>rowlabel</code> ' does not count in the column numbers. You can omit ' <code>n.cgroup</code> ' if all groups have the same number of columns.
<code>rgroup</code>	a vector of character strings containing headings for row groups. ' <code>n.rgroup</code> ' must be present when ' <code>rgroup</code> ' is given. The first ' <code>n.rgroup[1]</code> ' rows are sectioned off and ' <code>rgroup[1]</code> ' is used as a bold heading for them. The usual row dimnames (which must be present if ' <code>rgroup</code> ' is) are indented. The next ' <code>n.rgroup[2]</code> ' rows are treated likewise, etc.
<code>n.rgroup</code>	integer vector giving the number of rows in each grouping. If ' <code>rgroup</code> ' is not specified, ' <code>n.rgroup</code> ' is just used to divide off blocks of rows by horizontal lines. If ' <code>rgroup</code> ' is given but ' <code>n.rgroup</code> ' is omitted, ' <code>n.rgroup</code> ' will default so that each row group contains the same number of rows.
<code>digits</code>	causes all values in the table to be formatted to ' <code>digits</code> ' significant digits. ' <code>dec</code> ' is usually preferred.
<code>dec</code>	If ' <code>dec</code> ' is a scalar, all elements of the matrix will be rounded to ' <code>dec</code> ' decimal places to the right of the decimal. ' <code>dec</code> ' can also be a matrix whose elements correspond to ' <code>x</code> ', for customized rounding of each element.

rdec	a vector specifying the number of decimal places to the right for each row ('cdec' is more commonly used than 'rdec')
cdec	a vector specifying the number of decimal places for each column
append	defaults to 'F'. Set to 'T' to append output to an existing file.
dcolumn	Set to 'T' to use David Carlisles 'dcolumn' style for decimal alignment. Default is 'F', which aligns columns of numbers by changing leading blanks to "~", the LaTeX space-holder. You will probably want to use 'dcolumn' if you use 'rdec', as a column may then contain varying number of places to the right of the decimal. 'dcolumn' can line up all such numbers on the decimal point, with integer values right-justified at the decimal point location of numbers that actually contain decimal places.
cdot	Set to 'T' to use centered dots rather than ordinary periods in numbers.
longtable	Set to 'T' to use David Carlisles LaTeX 'longtable' style, allowing long tables to be split over multiple pages with headers repeated on each page.
table.env	Set 'table.env=FALSE' to suppress enclosing the table in a LaTeX 'table' environment. 'table.env' only applies when 'longtable=FALSE'. You may not specify a 'caption' if 'table.env=FALSE'.
lines.page	Applies if 'longtable=TRUE'. No more than 'lines.page' lines in the body of a table will be placed on a single page. Page breaks will only occur at 'rgroup' boundaries.
caption	a text string to use as a caption to print at the top of the first page of the table. Default is no caption.
caption.lot	a text string representing a short caption to be used in the "List of Tables". By default, LaTeX will use 'caption'.
label	a text string representing a symbolic label for the table for referencing with the LaTeX '\ref{label}' command. The default is 'file'. 'label' is only used if 'caption' is given.
double.slash	set to 'T' to output '\ as '\ in LaTeX commands. Useful when you are reading the output file back into an S vector for later output.
...	other optional arguments

**Value**

returns invisibly

**Author(s)**

Roger Koenker

**References**

Minor modification of Frank Harrell's Splus code

---

`lm.fit.recursive`     *Recursive Least Squares*

---

**Description**

This function fits a linear model by recursive least squares. It is a utility routine for the [khmaladzize](#) function of the quantile regression package.

**Usage**

```
lm.fit.recursive(X, y, int=TRUE)
```

**Arguments**

<code>X</code>	Design Matrix
<code>y</code>	Response Variable
<code>int</code>	if TRUE then append intercept to X

**Value**

return p by n matrix of fitted parameters, where p. The ith column gives the solution up to "time" i.

**Author(s)**

R. Koenker

**References**

A. Harvey, (1993) Time Series Models, MIT

**See Also**

[khmaladzize](#)

---

`lprq`     *locally polynomial quantile regression*

---

**Description**

This is a toy function to illustrate how to do locally polynomial quantile regression univariate smoothing.

**Usage**

```
lprq(x, y, h, tau = .5, m = 50)
```

**Arguments**

x	The conditioning covariate
y	The response variable
h	The bandwidth parameter
tau	The quantile to be estimated
m	The number of points at which the function is to be estimated

**Details**

The function obviously only does locally linear fitting but can be easily adapted to locally polynomial fitting of higher order. The author doesn't really approve of this sort of smoothing, being more of a spline person, so the code is left in its (almost) most trivial form.

**Value**

The function compute a locally weighted linear quantile regression fit at each of the m design points, and returns:

xx	The design points at which the evaluation occurs
fv	The estimated function values at these design points
dev	The estimated first derivative values at the design points

**Note**

One can also consider using B-spline expansions see `bs`.

**Author(s)**

R. Koenker

**References**

Koenker, R. (2004) Quantile Regression

**See Also**

`rqss` for a general approach to nonparametric QR fitting.

**Examples**

```
require(MASS)
data(mcycle)
attach(mcycle)
plot(times, accel, xlab = "milliseconds", ylab = "acceleration (in g)")
hs <- c(1, 2, 3, 4)
for(i in hs) {
  h = hs[i]
  fit <- lprq(times, accel, h=h, tau=.5)
  lines(fit$xx, fit$fv, lty=i)
```

```

    }
  legend(50, -70, c("h=1", "h=2", "h=3", "h=4"), lty=1:length(hs))

```

---

Mammals

*Garland(1983) Data on Running Speed of Mammals*


---

### Description

Observations on the maximal running speed of mammal species and their body mass.

### Usage

```
data(Mammals)
```

### Format

A data frame with 107 observations on the following 4 variables.

**weight** Body mass in Kg for "typical adult sizes"

**speed** Maximal running speed (fastest sprint velocity on record)

**hoppers** logical variable indicating animals that ambulate by hopping, e.g. kangaroos

**specials** logical variable indicating special animals with "lifestyles in which speed does not figure as an important factor": Hippopotamus, raccoon (*Procyon*), badger (*Meles*), coati (*Nasua*), skunk (*Mephitis*), man (*Homo*), porcupine (*Erithizon*), opossum (*didelphis*), and sloth (*Bradypus*)

### Details

Used by Chappell (1989) and Koenker, Ng and Portnoy (1994) to illustrate the fitting of piecewise linear curves.

### Source

Garland, T. (1983) The relation between maximal running speed and body mass in terrestrial mammals, *J. Zoology*, 199, 1557-1570.

### References

Koenker, R., P. Ng and S. Portnoy, (1994) Quantile Smoothing Splines" *Biometrika*, 81, 673-680.

Chappell, R. (1989) Fitting Bent Lines to Data, with Applications of Allometry, *J. Theo. Biology*, 138, 235-256.

### See Also

[rqss](#)

**Examples**

```

data(Mammals)
attach(Mammals)
x <- log(weight)
y <- log(speed)
plot(x,y, xlab="Weight in log(Kg)", ylab="Speed in log(Km/hour)",type="n")
points(x[hoppers],y[hoppers],pch = "h", col="red")
points(x[specials],y[specials],pch = "s", col="blue")
others <- (!hoppers & !specials)
points(x[others],y[others], col="black",cex = .75)
fit <- rqss(y ~ qss(x, lambda = 1),tau = .9)
plot(fit)

```

nlrq

*Function to compute nonlinear quantile regression estimates***Description**

This function implements an R version of an interior point method for computing the solution to quantile regression problems which are nonlinear in the parameters. The algorithm is based on interior point ideas described in Koenker and Park (1994).

**Usage**

```

nlrq(formula, data=parent.frame(), start, tau=0.5,
control, trace=FALSE,method="L-BFGS-B")
## S3 method for class 'nlrq':
summary(object, ...)
## S3 method for class 'summary.nlrq':
print(x, digits = max(5, .Options$digits - 2), ...)

```

**Arguments**

formula	formula for model in nls format; accept self-starting models
data	an optional data frame in which to evaluate the variables in ‘formula’
start	a named list or named numeric vector of starting estimates
tau	a vector of quantiles to be estimated
control	an optional list of control settings. See ‘nlrq.control’ for the names of the settable control values and their effect.
trace	logical value indicating if a trace of the iteration progress should be printed. Default is ‘FALSE’. If ‘TRUE’ intermediary results are printed at the end of each iteration.
method	method passed to optim for line search, default is "L-BFGS-B" but for some problems "BFGS" may be preferable. See <a href="#">optim</a> for further details. Note that the algorithm wants to pass upper and lower bounds for the line search to optim, which is fine for the L-BFGS-B method. Use of other methods will produce warnings about these arguments – so users should proceed at their own risk.

object	an object of class nlrq needing summary.
x	an object of class summary.nlrq needing printing.
digits	Significant digits reported in the printed table.
...	Optional arguments passed to printing function.

### Details

An 'nlrq' object is a type of fitted model object. It has methods for the generic functions 'coef' (parameters estimation at best solution), 'formula' (model used), 'deviance' (value of the objective function at best solution), 'print', 'summary', 'fitted' (vector of fitted variable according to the model), 'predict' (vector of data points predicted by the model, using a different matrix for the independent variables) and also for the function 'tau' (quantile used for fitting the model, as the tau argument of the function). Further help is also available for the method 'residuals'. The summary method for nlrq uses a bootstrap approach based on the final linearization of the model evaluated at the estimated parameters.

### Value

A list consisting of:

m	an 'nlrqModel' object similar to an 'nlsModel' in package nls
data	the expression that was passed to 'nlrq' as the data argument. The actual data values are present in the environment of the 'm' component.

### Author(s)

Based on S code by Roger Koenker modified for R and to accept models as specified by nls by Philippe Grosjean.

### References

Koenker, R. and Park, B.J. (1994). An Interior Point Algorithm for Nonlinear Quantile Regression, *Journal of Econometrics*, 71(1-2): 265-283.

### See Also

[nlrq.control](#), [residuals.nlrq](#)

### Examples

```
# build artificial data with multiplicative error
Dat <- NULL; Dat$x <- rep(1:25, 20)
set.seed(1)
Dat$y <- SSlogis(Dat$x, 10, 12, 2)*rnorm(500, 1, 0.1)
plot(Dat)
# fit first a nonlinear least-square regression
Dat.nls <- nls(y ~ SSlogis(x, Asym, mid, scal), data=Dat); Dat.nls
lines(1:25, predict(Dat.nls, newdata=list(x=1:25)), col=1)
# then fit the median using nlrq
Dat.nlrq <- nlrq(y ~ SSlogis(x, Asym, mid, scal), data=Dat, tau=0.5, trace=TRUE)
```

```

lines(1:25, predict(Dat.nlrq, newdata=list(x=1:25)), col=2)
# the 1st and 3rd quartiles regressions
Dat.nlrq <- nlrq(y ~ SSlogis(x, Asym, mid, scal), data=Dat, tau=0.25, trace=TRUE)
lines(1:25, predict(Dat.nlrq, newdata=list(x=1:25)), col=3)
Dat.nlrq <- nlrq(y ~ SSlogis(x, Asym, mid, scal), data=Dat, tau=0.75, trace=TRUE)
lines(1:25, predict(Dat.nlrq, newdata=list(x=1:25)), col=3)
# and finally "external envelopes" holding 95 percent of the data
Dat.nlrq <- nlrq(y ~ SSlogis(x, Asym, mid, scal), data=Dat, tau=0.025, trace=TRUE)
lines(1:25, predict(Dat.nlrq, newdata=list(x=1:25)), col=4)
Dat.nlrq <- nlrq(y ~ SSlogis(x, Asym, mid, scal), data=Dat, tau=0.975, trace=TRUE)
lines(1:25, predict(Dat.nlrq, newdata=list(x=1:25)), col=4)
leg <- c("least squares", "median (0.5)", "quartiles (0.25/0.75)", ".95 band (0.025/0.975)")
legend(1, 12.5, legend=leg, lty=1, col=1:4)

```

---

nlrq.control	<i>Set control parameters for nlrq</i>
--------------	--

---

## Description

Set algorithmic parameters for nlrq (nonlinear quantile regression function)

## Usage

```
nlrq.control(maxiter=100, k=2, InitialStepSize = 1, big=1e+20, eps=1e-07, beta=0.975)
```

## Arguments

<code>maxiter</code>	maximum number of allowed iterations
<code>k</code>	the number of iterations of the Meketon algorithm to be calculated in each step, usually 2 is reasonable, occasionally it may be helpful to set k=1
<code>InitialStepSize</code>	Starting value in <code>optim</code> to determine the step length of iterations. The default value of 1 is sometimes too optimistic. In such cases, the value 0 forces <code>optim</code> to just barely stick its toe in the water.
<code>big</code>	a large scalar
<code>eps</code>	tolerance for convergence of the algorithm
<code>beta</code>	a shrinkage parameter which controls the recentering process in the interior point algorithm.

## See Also

[nlrq](#)

---

Peirce

*C.S. Peirce's Auditory Response Data*

---

## Description

Data from sequence experiments conducted by C.S. Pierce in 1872 to determine the distribution of response times to an auditory stimulus.

## Usage

```
data(Peirce)
```

## Format

A `link{list}` of 24 objects each representing one day of the experiment. Each element of the list consists of three components: the date the measurements were made, an `x` component recording the response time in milliseconds, and an associated `y` component recording a count of the number of times that the response was recorded to be equal to the corresponding `x` entry. There are roughly 500 observations (counts) on each of the 24 days.

## Details

A detailed description of the experiment can be found in Peirce (1873). A young man of about 18 with no prior experience was employed to respond to a signal “consisting of a sharp sound like a rap, the answer being made upon a telegraph-operator’s key nicely adjusted.” The response times, made with the aid of a Hipp cronoscope were recorded to the nearest millisecond. The data was analyzed by Peirce who concluded that after the first day, when the the observer was entirely inexperienced, the curves representing the densities of the response times “differed very little from that derived from the theory of least squares,” i.e. from the Gaussian density.

The data was subsequently analysed by Samama, in a diploma thesis supervised by Maurice Frechet, who reported briefly the findings in Frechet (1924), and by Wilson and Hilferty (1929). In both instances the reanalysis showed that Laplace’s first law of error, the double exponential distribution, was a better representation for the data than was the Gaussian law. Koenker (2009) contains further discussion and an attempt to reproduce the Wilson and Hilferty analysis.

The data is available in two formats: The first in a "raw" form as 24 text files as scanned from the reprinted Peirce source, the second as an R dataset `Peirce.rda` containing the list. Only the latter is provided here, for the raw data and how to read see the more complete archive at: <http://www.econ.uiuc.edu/~roger/research/frechet/frechet.html> See the examples section below for some details on provisional attempt to reproduce part of the Wilson and Hilferty analysis. An open question regarding the dataset is: How did Wilson and Hilferty compute standard deviations for the median as they appear in their table? The standard textbook suggestion of Yule (1917) yields far too small a bandwidth. The methods employed in the example section below, which rely on relatively recent proposals, are somewhat closer, but still deviate somewhat from the results reported by Wilson and Hilferty.

## Source

Peirce, C.~S. (1873): "On the Theory of Errors of Observation," *Report of the Superintendent of the U.S. Coast Survey*, pp. 200–224, Reprinted in *The New Elements of Mathematics*, (1976) collected papers of C.S. Peirce, ed. by C. Eisele, Humanities Press: Atlantic Highlands, N.J., vol. 3, part 1, 639–676.

## References

Fréchet, M. (1924): "Sur la loi des erreurs d'observation," *Matematicheskii Sbornik*, 32, 5–8.  
 Koenker, R. (2009): "The Median is the Message: Wilson and Hilferty's Reanalysis of C.S. Peirce's Experiments on the Law of Errors," *American Statistician*, 63, 20–25. Wilson, E.~B., and M.~M. Hilferty (1929): "Note on C.S. Peirce's Experimental Discussion of the Law of Errors," *Proceedings of the National Academy of Sciences of the U.S.A.*, 15, 120–125. Yule, G.~U. (1917): *An Introduction to the Theory of Statistics*. Charles Griffen: London, 4 edn.

## Examples

```
# Make table like Wilson and Hilferty

data("Peirce")
set.seed(10) #Dither the counts
tab <- matrix(0,24,11)
for(i in 1:24){
  y <- rep(Peirce[[i]]$x, Peirce[[i]]$y) + runif(sum(Peirce[[i]]$y), -.5, .5)
  f1 <- summary(rq(y~1),se="iid")$coef[1:2]
  n <- length(y)
  f0 <- 1/(2 * sum(abs(y-f1[1])/n)) #Laplace proposal
  f0 <- (1/(2 * f0))/ sqrt(n)
  f2 <- summary(lm(y~1))$coef[1:2]
  outm <- sum(y < (f1[1] - 3.1 * sqrt(n) * f2[2]))
  outp <- sum(y > (f1[1] + 3.1 * sqrt(n) * f2[2]))
  outt <- outm + outp
  inm <- y > (f1[1] - 0.25 * sqrt(n) * f2[2])
  inp <- y < (f1[1] + 0.25 * sqrt(n) * f2[2])
  int <- sum(inm * inp)
  Eint <- round(n * (pnorm(.25) - pnorm(-.25)))
  excess <- round(100*(int - Eint)/Eint)
  tab[i,] <- c(f1, f0, f2, outm, outp, outt,int,Eint,excess)
  cnames <- c("med","sdmed1","sdmed0","mean","sdmean","below","above","outliers",
    "inliers","Einliers","ExcessIns")
  dimnames(tab) <- list(paste("Day",1:24),cnames)
}
```

**Description**

Function to plot quantile regression process.

**Usage**

```
## S3 method for class 'rq.process':
plot(x, nrow=3, ncol=2, ...)
```

**Arguments**

<code>x</code>	an object produced by <code>rq()</code> fitting
<code>nrow</code>	rows in mfrow
<code>ncol</code>	columns in mfrow
<code>...</code>	optional arguments to plot

**Author(s)**

Roger Koenker [rkoenker@uiuc.edu](mailto:rkoenker@uiuc.edu)

**See Also**

[rq](#)

---

plot.rqs

*Visualizing sequences of quantile regressions*

---

**Description**

A sequence of coefficient estimates for quantile regressions with varying `tau` parameters is visualized.

**Usage**

```
## S3 method for class 'rqs':
plot(x, parm = NULL, ols = TRUE,
     mfrow = NULL, mar = NULL, ylim = NULL, main = NULL, col = 1:2, lty = 1:2,
     cex = 0.5, pch = 20, type = "b", xlab = "", ylab = "", ...)
```

**Arguments**

<code>x</code>	an object of class "rqs" as produce by <a href="#">rq</a> (with a vector of <code>tau</code> values).
<code>parm</code>	a specification of which parameters are to be plotted, either a vector of numbers or a vector of names. By default, all parameters are considered.
<code>ols</code>	logical. Should a line for the OLS coefficient (as estimated by <a href="#">lm</a> ) be added?

`mfrow`, `mar`, `ylim`, `main`  
 graphical parameters. Suitable defaults are chosen based on the coefficients to be visualized.

`col`, `lty`  
 graphical parameters. For each parameter, the first element corresponds to the `rq` coefficients and the second to the `lm` coefficients.

`cex`, `pch`, `type`, `xlab`, `ylab`, ...  
 further graphical parameters passed to `plot`.

### Details

The `plot` method for "rqss" objects visualizes the coefficients only, confidence bands can be added by using the `plot` method for the associated "summary.rqss" object.

### Value

A matrix with all coefficients visualized is returned invisibly.

### See Also

[rq.plot.summary.rqss](#)

### Examples

```
## fit Engel models (in levels) for tau = 0.1, ..., 0.9
data("engel")
fm <- rq(foodexp ~ income, data = engel, tau = 1:9/10)

## visualizations
plot(fm)
plot(fm, parm = 2, mar = c(5.1, 4.1, 2.1, 2.1), main = "", xlab = "tau",
      ylab = "income coefficient", cex = 1, pch = 19)
```

---

plot.rqss

*Plot Method for rqss Objects*

---

### Description

Takes a fitted `rqss` object produced by `rqss()` and plots the component smooth functions that make up the ANOVA decomposition. Since the components "omit the intercept" the estimated intercept is added back in – this facilitates the comparison of quantile fits particularly. For models with a partial linear component or several `qss` components it may be preferable to plot the output of `predict.rqss`.

**Usage**

```
## S3 method for class 'rqss':
plot(x, rug = TRUE, jit = TRUE, bands = NULL, coverage = 0.95,
     add = FALSE, pages = 0, ...)
## S3 method for class 'qss1':
plot(x, rug = TRUE, jit = TRUE, add = FALSE, ...)
## S3 method for class 'qss2':
plot(x, render = "contour", ncol = 100, zcol = NULL, ...)
```

**Arguments**

x	a fitted <code>rqss</code> object produced by <code>rqss()</code> .
...	additional arguments for the plotting algorithm
rug	if TRUE, a rugplot for the x-coordinate is plotted
jit	if TRUE, the x-values of the rug plot are jittered
bands	if TRUE, confidence bands for the smoothed effects are plotted
coverage	desired coverage probability of confidence bands, if requested
pages	number of pages desired for the plots
render	a character specifying the rendering for bivariate fits; either "contour" (default) or "rgl". The latter requires package <code>rgl</code> .
add	if TRUE then add qss curve to existing (usually) scatterplot, otherwise initiate a new plot
ncol, zcol	Only for <code>render = "rgl"</code> : number of colors and z values for color construction.

**Value**

The function produces plots for the ANOVA components as a side effect. For "qss1" the "add = TRUE" can be used to overplot the fit on a scatterplot. When there are multiple pages required "par(ask = TRUE)" is turned on so that the plots may be examined sequentially. If `bands != NULL` then a list with three components for each qss component is returned:

x	The x coordinates of the confidence bands
blo	The y coordinates of the lower confidence curve
bhi	The y coordinates of the upper confidence curve

If `bands == "uniform"` then the bands are uniform bands based on the Hotelling (1939) tube approach. See also Naiman (1986), Johansen and Johnstone (1990), Sun and Loader (1994), and Krivobokova, Kneib, and Claeskens (2009), in particular the computation of the "tube length" is based on the last of these references. If `bands` is non null, and not "uniform" then pointwise bands are returned. Since bands for bivariate components are not (yet) supported, if requested such components will be returned as NULL.

**Author(s)**

Roger Koenker

## References

- [1] Hotelling, H. (1939): “Tubes and Spheres in  $n$ -spaces, and a class of statistical problems,” *Am J. Math*, 61, 440–460.
- [2] Johansen, S., and I.M. Johnstone (1990): “Hotelling’s Theorem on the Volume of Tubes: Some Illustrations in Simultaneous Inference and Data Analysis,” *The Annals of Statistics*, 18, 652–684.
- [3] Naiman, D. (1986) Conservative confidence bands in curvilinear regression, *The Annals of Statistics*, 14, 896–906.
- [4] Sun, J. and C.R. Loader, (1994) Simultaneous confidence bands for linear regression and smoothing, *The Annals of Statistics*, 22, 1328–1345.
- [5] Krivobokova, T., T. Kneib, and G. Claeskens (2009) Simultaneous Confidence Bands for Penalized Spline Estimators, preprint.

## See Also

[rqss](#)

## Examples

```
n <- 200
x <- sort(rchisq(n, 4))
z <- x + rnorm(n)
y <- log(x) + .1*(log(x))^2 + log(x)*rnorm(n)/4 + z
plot(x, y-z)
fN <- rqss(y~qss(x, constraint="N")+z)
plot(fN)
fI <- rqss(y~qss(x, constraint="I")+z)
plot(fI, col="blue")
fCI <- rqss(y~qss(x, constraint="CI")+z)
plot(fCI, col="red")

## A bivariate example
data(CobarOre)
fCO <- rqss(z~qss(cbind(x, y), lambda=.08), data = CobarOre)
plot(fCO)
```

---

plot.summary.rqs     *Visualizing sequences of quantile regression summaries*

---

## Description

A sequence of coefficient estimates for quantile regressions with varying  $\tau$  parameters is visualized along with associated confidence bands.

**Usage**

```
## S3 method for class 'summary.rqs':
plot(x, parm = NULL, level = 0.9, ols = TRUE,
     mfrow = NULL, mar = NULL, ylim = NULL, main = NULL,
     col = gray(c(0, 0.75)), border = NULL, lcol = 2, lty = 1:2,
     cex = 0.5, pch = 20, type = "b", xlab = "", ylab = "", ...)
```

**Arguments**

<code>x</code>	an object of class "summary.rqs" as produce by applying the <code>summary</code> method to a <code>rq</code> object (with a vector of <code>tau</code> values).
<code>parm</code>	a specification of which parameters are to be plotted, either a vector of numbers or a vector of names. By default, all parameters are considered.
<code>level</code>	numeric. Confidence level of bands.
<code>ols</code>	logical. Should a line for the OLS coefficient and their confidence bands (as estimated by <code>lm</code> ) be added?
<code>mfrow, mar, ylim, main</code>	graphical parameters. Suitable defaults are chosen based on the coefficients to be visualized.
<code>col</code>	vector of color specification for <code>rq</code> coefficients and the associated confidence polygon.
<code>border</code>	color specification for the confidence polygon. By default, the second element of <code>col</code> is used.
<code>lcol, lty</code>	color and line type specification for OLS coefficients and their confidence bounds.
<code>cex, pch, type, xlab, ylab, ...</code>	further graphical parameters passed to <code>points</code> .

**Details**

The `plot` method for "summary.rqs" objects visualizes the coefficients along with their confidence bands. The bands can be omitted by using the `plot` method for "rqs" objects directly.

**Value**

An array with all coefficients visualized (and associated confidence bands) is returned invisibly.

**See Also**

[rq.plot.rqs](#)

**Examples**

```
## fit Engel models (in levels) for tau = 0.1, ..., 0.9
data("engel")
fm <- rq(foodexp ~ income, data = engel, tau = 1:9/10)
sfm <- summary(fm)
```

```
## visualizations
plot(sfm)
plot(sfm, parm = 2, mar = c(5.1, 4.1, 2.1, 2.1), main = "", xlab = "tau",
      ylab = "income coefficient", cex = 1, pch = 19)
```

---

predict.rq

*Quantile Regression Prediction*


---

## Description

Prediction based on fitted quantile regression model

## Usage

```
## S3 method for class 'rq':
predict(object, newdata, type = "none", interval = c("none", "confidence"),
        level = .95, na.action = na.pass, ...)
## S3 method for class 'rqs':
predict(object, newdata, type = NULL, stepfun = FALSE, na.action = na.pass, ...)
## S3 method for class 'rq.process':
predict(object, newdata, type = NULL, stepfun = FALSE, na.action = na.pass, ...)
```

## Arguments

object	object of class rq or rqs or rq.process produced by rq
newdata	An optional data frame in which to look for variables with which to predict. If omitted, the fitted values are used.
interval	type of interval desired: default is 'none', when set to 'confidence' the function returns a matrix predictions with point predictions for each of the 'newdata' points as well as lower and upper confidence limits.
level	converage probability for the 'confidence' intervals.
type	method for 'confidence' intervals, if desired. If 'percentile' then one of the bootstrap methods is used to generate percentile intervals for each prediction, if 'direct' then a version of the Portnoy and Zhou (1998) method is used, and otherwise an estimated covariance matrix for the parameter estimates is used. Further arguments to determine the choice of bootstrap method or covariance matrix estimate can be passed via the ... argument.
stepfun	If 'TRUE' return stepfunctions otherwise return matrix of predictions.
na.action	function determining what should be done with missing values in 'newdata'. The default is to predict 'NA'.
...	Further arguments passed to or from other methods.

## Details

Produces predicted values, obtained by evaluating the quantile regression function in the frame 'newdata' (which defaults to 'model.frame(object)'). These predictions purport to estimate the conditional quantile function of the response variable of the fitted model evaluated at the covariate values specified in "newdata" and the quantile(s) specified by the "tau" argument. Several methods are provided to compute confidence intervals for these predictions.

## Value

A vector or matrix of predictions, depending upon the setting of 'interval'. In the case that there are multiple taus in `object` when `object` is of class 'rq' setting 'stepfun = TRUE' will produce a `stepfun` object or a list of `stepfun` objects. The function `rearrange` can be used to monotonize these step-functions, if desired.

## Author(s)

R. Koenker

## References

Zhou, Kenneth Q. and Portnoy, Stephen L. (1998) Statistical inference on heteroscedastic models based on regression quantiles *Journal of Nonparametric Statistics*, 9, 239-260

## See Also

[rq](#) [rearrange](#)

## Examples

```
data(airquality)
airq <- airquality[143:145,]
f <- rq(Ozone ~ ., data=airquality)
predict(f, newdata=airq)
f <- rq(Ozone ~ ., data=airquality, tau=1:19/20)
fp <- predict(f, newdata=airq, stepfun = TRUE)
fpr <- rearrange(fp)
plot(fp[[2]], main = "Ozone Prediction")
par(col="red")
lines(fpr[[2]])
legend(.2, 20, c("raw", "cooked"), lty = c(1, 1), col=c("black", "red"))
```

---

predict.rqss	<i>Predict from fitted nonparametric quantile regression smoothing spline models</i>
--------------	--

---

### Description

Additive models for nonparametric quantile regression using total variation penalty methods can be fit with the `rqss` function. Univariate and bivariate components can be predicted using these functions.

### Usage

```
## S3 method for class 'rqss':  
predict(object, newdata, ...)  
## S3 method for class 'qss1':  
predict(object, newdata, ...)  
## S3 method for class 'qss2':  
predict(object, newdata, ...)
```

### Arguments

<code>object</code>	is a fitted object produced by <code>rqss</code>
<code>newdata</code>	a data frame describing the observations at which prediction is to be made. For qss components, <code>newdata</code> should lie in strictly within the convex hull of the fitting data. Newdata corresponding to the partially linear component of the model may require caution concerning the treatment of factor levels, if any.
<code>...</code>	optional arguments

### Details

For both univariate and bivariate prediction linear interpolation is done. In the bivariate case, this involves computing barycentric coordinates of the new points relative to their enclosing triangles. It may be of interest to plot individual components of fitted `rqss` models: this is usually best done by fixing the values of other covariates at reference values typical of the sample data and predicting the response at varying values of one qss term at a time. Direct use of the `predict.qss1` and `predict.qss2` functions is discouraged since it usually corresponds to predicted values at absurd reference values of the other covariates, i.e. zero.

### Value

A vector of predictions.

### Author(s)

R. Koenker

**See Also**[rqss](#)**Examples**

```

n <- 200
x <- sort(rchisq(n,4))
z <- exp(rnorm(n)) + x
y <- log(x) + .1*(log(x))^2 + z/4 + log(x)*rnorm(n)/4
plot(x,y - z/4 + mean(z)/4)
fit <- rqss(y ~ qss(x,constraint="I") + z)
xz <- data.frame(z = mean(z),
                 x = seq(min(x)+.01,max(x)-.01,by=.25))
lines(xz[["x"]], predict(fit, xz), col=2)
title("Predicted Median Response at Mean Value of z")

## Bivariate example -- loads pkg "tripack"
data(CobarOre)
fit <- rqss(z ~ qss(cbind(x,y), lambda=.08),
            data= CobarOre)
plot(fit, col="grey",
     main = "CobarOre data -- rqss(z ~ qss( (x,y)))")
T <- with(CobarOre, tri.mesh(x, y))
set.seed(77)
ndum <- 100
xd <- with(CobarOre, runif(ndum, min(x), max(x)))
yd <- with(CobarOre, runif(ndum, min(y), max(y)))
table(s <- in.convex.hull(T, xd, yd))
pred <- predict(fit, data.frame(x = xd[s], y = yd[s]))
contour(interp(xd[s],yd[s], pred),
          col="red", add = TRUE)

```

---

`print.rq`*Print an rq object*

---

**Description**Print an object generated by `rq`**Usage**

```

## S3 method for class 'rq':
print(x, ...)
## S3 method for class 'rqs':
print(x, ...)

```

**Arguments**

x                    Object returned from rq representing the fit of the model.  
 ...                  Optional arguments.

**See Also**

[rq](#)

---

print.summary.rq     *Print Quantile Regression Summary Object*

---

**Description**

Print summary of quantile regression object

**Usage**

```
## S3 method for class 'summary.rq':
print(x, digits=max(5, .Options$digits - 2), ...)
## S3 method for class 'summary.rqs':
print(x, ...)
```

**Arguments**

x                    This is an object of class "summary.rq" produced by a call to summary.rq().  
 digits              Significant digits reported in the printed table.  
 ...                  Optional arguments passed to printing function

**See Also**

[summary.rq](#)

---

qss                    *Additive Nonparametric Terms for rqss Fitting*

---

**Description**

In the formula specification of `rqss` nonparametric terms are specified with `qss`. Both univariate and bivariate specifications are possible, and qualitative constraints may also be specified for the `qss` terms.

**Usage**

```
qss(x, constraint = "N", lambda = 1, ndum = 0, dummies = NULL, w = rep(1, length(x)))
```

## Arguments

<code>x</code>	The covariate determining the nonparametric component, if <code>x</code> is a matrix with two columns then the <code>qss</code> function will construct a penalized triogram term.
<code>lambda</code>	The smoothing parameter governing the tradeoff between fidelity and the penalty component for this term. In future versions there should be an automatic mechanism for default choice of the lambdas. For now, this is the responsibility of the user.
<code>constraint</code>	Optional specification of qualitative constraints on the fitted univariate <code>qss</code> functions, take the values: "N","I","D","U","C" "UI","UD","CI","CD" for none, increasing, decreasing, convex, concave, convex and increasing, etc. And for bivariate <code>qss</code> components can take the values "N","U","C" for none, convex, and concave.
<code>ndum</code>	number of dummy vertices: this is only relevant for <code>qss2</code> terms. In addition to vertices at the observed ( <code>x,y</code> ) points <code>ndum</code> dummy vertices are generated – distributed uniformly over the rectangle given by the Cartesian product of the ranges of <code>x</code> and <code>y</code> – observations that fall in the convex hull of the observations are retained. So the actual number of dummy vertices used is smaller than <code>ndum</code> . The values of these vertices are returned in the list <code>dummies</code> , so that they can be reused.
<code>dummies</code>	list of dummy vertices as generated, for example by <code>triogram.fidelity</code> when <code>ndum &gt; 0</code> . Should be a list with <code>x</code> and <code>y</code> components. These points should lie inside the convex hull of the real <code>xy</code> points, but no explicit checking of this assertion is currently done.
<code>w</code>	weights not yet unimplemented

## Details

The various pieces returned are stored in sparse matrix.csr form. See [rqss](#) for details on how they are assembled. To preserve the sparsity of the design matrix the first column of each `qss` term is dropped. This differs from the usual convention that would have forced `qss` terms to have mean zero. This convention has implications for prediction that need to be recognized. The penalty components for `qss` terms are based on total variation penalization of the first derivative (and gradient, for bivariate `x`) as described in the references appearing in the help for `rqss`.

For the bivariate case, package **tripack** (and for plotting also **akima**) are required (automatically, by the R code).

## Value

<code>F</code>	Fidelity component of the design matrix
<code>dummies</code>	List of dummy vertices
<code>A</code>	Penalty component of the design matrix
<code>R</code>	Constraint component of the design matrix
<code>r</code>	Constraint component of the rhs

**Author(s)**

Roger Koenker

**See Also**[rqss](#)

QTECox

*Function to obtain QTE from a Cox model***Description**

Computes quantile treatment effects comparable to those of crq model from a coxph object.

**Usage**

```
QTECox(x, smooth = TRUE)
```

**Arguments**

`x` An object of class `coxph` produced by `coxph`.  
`smooth` Logical indicator if TRUE (default) then Cox survival function is smoothed.

**Details**

Estimates of the Cox QTE,  $\frac{dQ(t|x)}{dx_j}$  at  $x = \bar{x}$ , can be expressed as a function of  $t$  as follows:

$$\frac{dQ(t|x)}{dx_j} = \frac{dt}{dx_j} \frac{dQ(t|x)}{dt}$$

The Cox survival function,  $S(y|x) = \exp\{-H_0(y) \exp(b'x)\}$

$$\frac{dS(y|x)}{dx_j} = S(y|x) \log\{S(y|x)\} b_j$$

where  $\frac{dQ(t|x)}{dx_j}$  can be estimated by  $\frac{\Delta(t)}{\Delta(S)}(1-t)$  where  $\Delta(t)$  and  $\Delta(S)$  denote the `surv` and `time` components of the `survfit` object. Note that since  $t = 1 - S(y|x)$ , the above is the value corresponding to the argument  $(1-t)$ ; and furthermore

$$\frac{dt}{dx_j} = -\frac{dS(y|x)}{dx_j} = -(1-t) \log(1-t) b_j$$

Thus the QTE at the mean of  $x$ 's is:

$$(1-S) = \frac{\Delta(t)}{\Delta(S)} S \log(S) b_j$$

Since  $\Delta S$  is negative and  $\log(S)$  is also negative this has the same sign as  $b_j$ . The crq model fits the usual AFT form  $\text{Surv}(\log(\text{Time}), \text{Status})$ , then

$$\frac{d \log(Q(t|x))}{dx_j} = \frac{dQ(t|x)}{dx_j} / Q(t|x)$$

This is the matrix form returned.

### Value

`taus` points of evaluation of the QTE.  
`QTE` matrix of QTEs, the *i*th column contains the QTE for the *i*th covariate effect. Note that there is no intercept effect. see `plot.summary.crqs` for usage.

### Author(s)

Roger Koenker Stephen Portnoy & Tereza Neocleous

### References

Koenker, R. and Geling, O. (2001). Reappraising Medfly longevity: a quantile regression survival analysis, *J. Amer. Statist. Assoc.*, 96, 458-468

### See Also

[crq](#)

---

ranks

*Quantile Regression Ranks*

---

### Description

Function to compute ranks from the dual (regression rankscore) process.

### Usage

```
ranks(v, score="wilcoxon", tau=0.5, trim = NULL)
```

### Arguments

`v` object of class "rq.process" generated by `rq()`  
`score` The score function desired. Currently implemented score functions are "wilcoxon", "normal", and "sign" which are asymptotically optimal for the logistic, Gaussian and Laplace location shift models respectively. The "normal" score function is also sometimes called van der Waerden scores. Also implemented are the "tau" which generalizes sign scores to an arbitrary quantile, and "interquartile" which is appropriate for tests of scale shift.

tau the optional value of tau if the "tau" score function is used.

trim optional trimming proportion parameter(s) – only applicable for the Wilcoxon score function – when one value is provided there is symmetric trimming of the score integral to the interval (trim, 1-trim), when there are two values provided, then the trimming restricts the integration to (trim[1], trim[2]).

### Details

See GJKP(1993) for further details.

### Value

The function returns two components. One is the ranks, the other is a scale factor which is the  $L_2$  norm of the score function. All score functions should be normalized to have mean zero.

### References

Gutenbrunner, C., J. Jureckova, Koenker, R. and Portnoy, S. (1993) Tests of linear hypotheses based on regression rank scores, *Journal of Nonparametric Statistics*, (2), 307–331.

### See Also

[rq](#), [rq.test](#), [rank anova](#)

### Examples

```
data(stackloss)
ranks(rq(stack.loss ~ stack.x, tau=-1))
```

---

rearrange	<i>Rearrangement</i>
-----------	----------------------

---

### Description

Monotonize a step function by rearrangement

### Usage

```
rearrange(f, xmin, xmax)
```

### Arguments

f object of class stepfun

xmin minimum of the support of the rearranged f

xmax maximum of the support of the rearranged f

**Details**

Given a stepfunction  $Q(u)$ , not necessarily monotone, let  $F(y) = \int \{Q(u) \leq y\} du$  denote the associated cdf obtained by randomly evaluating  $Q$  at  $U \sim U[0, 1]$ . The rearranged version of  $Q$  is  $\tilde{Q}(u) = \inf\{y : F(y) \geq u\}$ . The rearranged function inherits the right or left continuity of original stepfunction.

**Value**

Produces transformed stepfunction that is monotonic increasing.

**Author(s)**

R. Koenker

**References**

Chernozhukov, V., I. Fernandez-Val, and A. Galichon, (2006) Quantile and Probability Curves without Crossing, MIT Working Paper.

Chernozhukov, V., I. Fernandez-Val, and A. Galichon, (2006) Improving Estimates of Monotone Functions by Rearrangement, MIT Working Paper.

Hardy, G.H., J.E. Littlewood, and G. Polya (1934) Inequalities, Cambridge U. Press.

**See Also**

[rq rearrange](#)

**Examples**

```
data(engel)
z <- rq(foodexp ~ income, tau = -1, data = engel)
zp <- predict(z, newdata = list(income = quantile(engel$income, .03)), type = "stepfun")
plot(zp, do.points = FALSE, xlab = expression(tau),
      ylab = expression(Q ( tau )), main = "Engel Food Expenditure Quantiles")
plot(rearrange(zp), do.points = FALSE, add = TRUE, col.h = "red", col.v = "red")
legend(.6, 300, c("Before Rearrangement", "After Rearrangement"), lty = 1, col = c("black", "red"))
```

---

residuals.nlrq

*Return residuals of an nlrq object*

---

**Description**

Set algorithmic parameters for nlrq (nonlinear quantile regression function)

**Usage**

```
## S3 method for class 'nlrq':
residuals(object, type = c("response", "rho"), ...)
```

**Arguments**

object	an 'nlrq' object as returned by function 'nlrq'
type	the type of residuals to return: "response" is the distance between observed and predicted values; "rho" is the weighted distance used to calculate the objective function in the minimisation algorithm as $\tau * \text{pmax}(\text{resid}, 0) + (1 - \tau) * \text{pmin}(\text{resid}, 0)$ , where resid are the simple residuals as above (with type="response").
...	further arguments passed to or from other methods.

**See Also**

[nlrq](#)

---

rq	<i>Quantile Regression</i>
----	----------------------------

---

**Description**

Returns an object of class "rq" "rqs" or "rq.process" that represents a quantile regression fit.

**Usage**

```
rq(formula, tau=.5, data, subset, weights, na.action,
    method="br", model = TRUE, contrasts, ...)
```

**Arguments**

formula	a formula object, with the response on the left of a ~ operator, and the terms, separated by + operators, on the right.
tau	the quantile(s) to be estimated, this is generally a number strictly between 0 and 1, but if specified strictly outside this range, it is presumed that the solutions for all values of tau in (0,1) are desired. In the former case an object of class "rq" is returned, in the latter, an object of class "rq.process" is returned. As of version 3.50, tau can also be a vector of values between 0 and 1; in this case an object of class "rqs" is returned containing among other things a matrix of coefficient estimates at the specified quantiles.
data	a data.frame in which to interpret the variables named in the formula, or in the subset and the weights argument. If this is missing, then the variables in the formula should be on the search list. This may also be a single number to handle some special cases – see below for details.
subset	an optional vector specifying a subset of observations to be used in the fitting process.

<code>weights</code>	vector of observation weights; if supplied, the algorithm fits to minimize the sum of the weights multiplied into the absolute residuals. The length of weights must be the same as the number of observations. The weights must be nonnegative and it is strongly recommended that they be strictly positive, since zero weights are ambiguous.
<code>na.action</code>	a function to filter missing data. This is applied to the <code>model.frame</code> after any subset argument has been used. The default (with <code>na.fail</code> ) is to create an error if any missing values are found. A possible alternative is <code>na.omit</code> , which deletes observations that contain one or more missing values.
<code>model</code>	if TRUE then the model frame is returned. This is essential if one wants to call <code>summary</code> subsequently.
<code>method</code>	the algorithmic method used to compute the fit. There are several options: The default method is the modified version of the Barrodale and Roberts algorithm for $l_1$ -regression, used by <code>l1fit</code> in S, and is described in detail in Koenker and d'Orey(1987, 1994), default = "br". This is quite efficient for problems up to several thousand observations, and may be used to compute the full quantile regression process. It also implements a scheme for computing confidence intervals for the estimated parameters, based on inversion of a rank test described in Koenker(1994). For larger problems it is advantageous to use the Frisch–Newton interior point method "fn". And very large problems one can use the Frisch–Newton approach after preprocessing "pfn". Both of the latter methods are described in detail in Portnoy and Koenker(1997). There is a fifth option "fnc" that enables the user to specify linear inequality constraints on the fitted coefficients; in this case one needs to specify the matrix $R$ and the vector $r$ representing the constraints in the form $Rb \geq r$ . See the examples. Finally, there are two penalized methods: "lasso" and "scad" that implement the lasso penalty and Fan and Li's smoothly clipped absolute deviation penalty, respectively. These methods should probably be regarded as experimental.
<code>contrasts</code>	a list giving contrasts for some or all of the factors default = NULL appearing in the model formula. The elements of the list should have the same name as the variable and should be either a contrast matrix (specifically, any full-rank matrix with as many rows as there are levels in the factor), or else a function to compute such a matrix given the number of levels.
<code>...</code>	additional arguments for the fitting routines (see <code>rq.fit.br</code> and <code>rq.fit.fnb</code> and the functions they call).

## Details

For further details see the vignette available from R with `vignette("rq", package="quantreg")` and/or the Koenker (2005). For estimation of nonlinear (in parameters) quantile regression models there is the function `nllrq` and for nonparametric additive quantile regression there is the function `rqss`. Fitting of quantile regression models with censored data is handled by the `crq` function. Portnoy's (2003) method for estimating quantile regression models with random censoring is available in a separate package, `crq`, that will eventually be folded into `quantreg`.

**Value**

See `rq.object` and `rq.process.object` for details. Inferential matters are handled with `summary`. There are extractor methods `logLik` and `AIC` that are potentially relevant for model selection.

**Method**

The function computes an estimate on the tau-th conditional quantile function of the response, given the covariates, as specified by the formula argument. Like `lm()`, the function presumes a linear specification for the quantile regression model, i.e. that the formula defines a model that is linear in parameters. For non-linear quantile regression see the package `nlrq()`. The function minimizes a weighted sum of absolute residuals that can be formulated as a linear programming problem. As noted above, there are three different algorithms that can be chosen depending on problem size and other characteristics. For moderate sized problems ( $n \ll 5,000, p \ll 20$ ) it is recommended that the default "br" method be used. There are several choices of methods for computing confidence intervals and associated test statistics. See the documentation for `summary.rq` for further details and options.

**References**

- [1] Koenker, R. W. and Bassett, G. W. (1978). Regression quantiles, *Econometrica*, **46**, 33–50.
- [2] Koenker, R.W. and d’Orey (1987, 1994). Computing regression quantiles. *Applied Statistics*, **36**, 383–393, and **43**, 410–414.
- [3] Gutenbrunner, C. Jureckova, J. (1991). Regression quantile and regression rank score process in the linear model and derived statistics, *Annals of Statistics*, **20**, 305–330.
- [4] Koenker, R. W. (1994). Confidence Intervals for regression quantiles, in P. Mandl and M. Huskova (eds.), *Asymptotic Statistics*, 349–359, Springer-Verlag, New York.
- [5] Koenker, R. and S. Portnoy (1997) The Gaussian Hare and the Laplacean Tortoise: Computability of Squared-error vs Absolute Error Estimators, (with discussion). *Statistical Science*, **12**, 279–300.
- [6] Koenker, R. W. (2005). *Quantile Regression*, Cambridge U. Press.

There is also recent information available at the URL: <http://www.econ.uiuc.edu>.

**See Also**

`summary.rq`, `nlrq`, `rqss`, `rq.object`, `rq.process.object`

**Examples**

```
data(stackloss)
rq(stack.loss ~ stack.x, .5) #median (l1) regression fit for the stackloss data.
rq(stack.loss ~ stack.x, .25) #the 1st quartile,
#note that 8 of the 21 points lie exactly on this plane in 4-space!
rq(stack.loss ~ stack.x, tau=-1) #this returns the full rq process
rq(rnorm(50) ~ 1, ci=FALSE) #ordinary sample median --no rank inversion ci
rq(rnorm(50) ~ 1, weights=runif(50), ci=FALSE) #weighted sample median
#plot of engel data and some rq lines see KB(1982) for references to data
data(engel)
```

```

attach(engel)
plot(income, foodexp, xlab="Household Income", ylab="Food Expenditure", type = "n", cex=.5)
points(income, foodexp, cex=.5, col="blue")
taus <- c(.05, .1, .25, .75, .9, .95)
xx <- seq(min(income), max(income), 100)
f <- coef(rq((foodexp)~(income), tau=taus))
yy <- cbind(1, xx)%*%f
for(i in 1:length(taus)){
  lines(xx, yy[,i], col = "gray")
}
abline(lm(foodexp ~ income), col="red", lty = 2)
abline(rq(foodexp ~ income), col="blue")
legend(3000, 500, c("mean (LSE) fit", "median (LAE) fit"),
col = c("red", "blue"), lty = c(2, 1))
#Example of plotting of coefficients and their confidence bands
plot(summary(rq(foodexp~income, tau = 1:49/50, data=engel)))
#Example to illustrate inequality constrained fitting
n <- 100
p <- 5
X <- matrix(rnorm(n*p), n, p)
y <- .95*apply(X, 1, sum)+rnorm(n)
#constrain slope coefficients to lie between zero and one
R <- cbind(0, rbind(diag(p), -diag(p)))
r <- c(rep(0, p), -rep(1, p))
rq(y~X, R=R, r=r, method="fnc")

```

---

rq.fit

---

*Function to choose method for Quantile Regression*


---

## Description

Function to choose method for quantile regression

## Usage

```
rq.fit(x, y, tau=0.5, method="br", ...)
```

## Arguments

x	the design matrix
y	the response variable
tau	the quantile desired, if tau lies outside (0,1) the whole process is estimated.
method	method of computation: "br" is Barrodale and Roberts exterior point "fn" is the Frisch-Newton interior point method.
...	Optional arguments passed to fitting routine.

## See Also

[rq](#) [rq.fit.br](#) [rq.fit.fn](#)

**Description**

This function controls the details of QR fitting by the simplex approach embodied in the algorithm of Koenker and d'Orey based on the median regression algorithm of Barrodale and Roberts. Typically, options controlling the construction of the confidence intervals would be passed via the `...{}` argument of `rq()`.

**Usage**

```
rq.fit.br(x, y, tau=0.5, alpha=0.1, ci=FALSE, iid=TRUE, interp=TRUE, tcrit=TRUE)
```

**Arguments**

<code>x</code>	the design matrix
<code>y</code>	the response variable
<code>tau</code>	the quantile desired, if tau lies outside (0,1) the whole process is estimated.
<code>alpha</code>	the nominal noncoverage probability for the confidence intervals, i.e. 1-alpha is the nominal coverage probability of the intervals.
<code>ci</code>	logical flag if T then compute confidence intervals for the parameters using the rank inversion method of Koenker (1994). See <code>rq()</code> for more details. If F then return only the estimated coefficients. Note that for large problems the default option <code>ci = TRUE</code> can be rather slow. Note also that rank inversion only works for $p > 1$ , an error message is printed in the case that <code>ci=T</code> and $p=1$ .
<code>iid</code>	logical flag if T then the rank inversion is based on an assumption of iid error model, if F then it is based on an iid error assumption. See Koenker and Machado (1999) for further details on this distinction.
<code>interp</code>	As with typical order statistic type confidence intervals the test statistic is discrete, so it is reasonable to consider intervals that interpolate between values of the parameter just below the specified cutoff and values just above the specified cutoff. If <code>interp = F</code> then the 2 "exact" values above and below on which the interpolation would be based are returned.
<code>tcrit</code>	Logical flag if T - Student t critical values are used, if F then normal values are used.

**Details**

If tau lies in (0,1) then an object of class "rq" is returned with various related inference apparatus. If tau lies outside [0,1] then an object of class `rq.process` is returned. In this case parametric programming methods are used to find all of the solutions to the QR problem for tau in (0,1), the p-variate resulting process is then returned as the array `sol` containing the primal solution and `dsol` containing the dual solution. There are roughly  $O(n \log n)$  distinct solutions, so users should be aware that these arrays may be large and somewhat time consuming to compute for large problems.

**Value**

Returns an object of class "rq" for tau in (0,1), or else of class "rq.process". Note that `rq.fit.br` when called for a single tau value will return the vector of optimal dual variables. See [rq.object](#) and [rq.process.object](#) for further details.

**References**

Koenker, R. and J.A.F. Machado, (1999) Goodness of fit and related inference processes for quantile regression, *J. of Am Stat. Assoc.*, 94, 1296-1310.

**See Also**

[rq](#), [rq.fit.fnb](#)

**Examples**

```
data(stackloss)
rq.fit.br(stack.x, stack.loss, tau=.73 ,interp=FALSE)
```

---

rq.fit.fnb

*Quantile Regression Fitting via Interior Point Methods*

---

**Description**

This is a lower level routine called by `rq()` to compute quantile regression methods using the Frisch-Newton algorithm.

**Usage**

```
rq.fit.fnb(x, y, tau=0.5, beta=0.99995, eps=1e-06)
```

**Arguments**

x	The design matrix
y	The response vector
tau	The quantile of interest, must lie in (0,1)
beta	technical step length parameter – alter at your own risk!
eps	tolerance parameter for convergence. In cases of multiple optimal solutions there may be some discrepancy between solutions produced by method "fn" and method "br". This is due to the fact that "fn" tends to converge to a point near the centroid of the solution set, while "br" stops at a vertex of the set.

**Details**

The details of the algorithm are explained in Koenker and Portnoy (1997). The basic idea can be traced back to the log-barrier methods proposed by Frisch in the 1950's for constrained optimization. But the current implementation is based on proposals by Mehrotra and others in the recent (explosive) literature on interior point methods for solving linear programming problems. This function replaces an earlier one `rq.fit.fn`, which required the initial dual values to be feasible. This version allows the user to specify an infeasible starting point for the dual problem, that is one that may not satisfy the dual equality constraints. It still assumes that the starting value satisfies the upper and lower bounds.

**Value**

returns an object of class "rq", which can be passed to `summary.rq` to obtain standard errors, etc.

**References**

Koenker, R. and S. Portnoy (1997). The Gaussian Hare and the Laplacian Tortoise: Computability of squared-error vs. absolute-error estimators, with discussion, *Statistical Science*, **12**, 279-300.

**See Also**

[rq](#), [rq.fit.br](#), [rq.fit.pfn](#)

---

`rq.fit.fnc`

*Quantile Regression Fitting via Interior Point Methods*

---

**Description**

This is a lower level routine called by `rq()` to compute quantile regression methods using the Frisch-Newton algorithm. It allows the call to specify linear inequality constraints to which the fitted coefficients will be subjected. The constraints are assumed to be formulated as  $Rb \geq r$ .

**Usage**

```
rq.fit.fnc(x, y, R, r, tau=0.5, beta=0.9995, eps=1e-06)
```

**Arguments**

<code>x</code>	The design matrix
<code>y</code>	The response vector
<code>R</code>	The matrix describing the inequality constraints
<code>r</code>	The right hand side vector of inequality constraints
<code>tau</code>	The quantile of interest, must lie in (0,1)
<code>beta</code>	technical step length parameter – alter at your own risk!

`eps` tolerance parameter for convergence. In cases of multiple optimal solutions there may be some discrepancy between solutions produced by method "`fn`" and method "`br`". This is due to the fact that "`fn`" tends to converge to a point near the centroid of the solution set, while "`br`" stops at a vertex of the set.

### Details

The details of the algorithm are explained in Koenker and Ng (2002). The basic idea can be traced back to the log-barrier methods proposed by Frisch in the 1950's for constrained optimization. But the current implementation is based on proposals by Mehrotra and others in the recent (explosive) literature on interior point methods for solving linear programming problems. See "`rq`" helpfile for an example. It is an open research problem to provide an inference apparatus for inequality constrained quantile regression.

### Value

returns an object of class "`rq`", which can be passed to `summary.rq` to obtain standard errors, etc.

### References

Koenker, R. and S. Portnoy (1997). The Gaussian Hare and the Laplacian Tortoise: Computability of squared-error vs. absolute-error estimators, with discussion, *Statistical Science*, **12**, 279-300.

Koenker, R. and P. Ng(2005). Inequality Constrained Quantile Regression, *Sankhya*, 418-440.

### See Also

[rq](#), [rq.fit.br](#), [rq.fit.pfn](#)

---

`rq.fit.lasso`

*Lasso Penalized Quantile Regression*

---

### Description

The fitting method implements the lasso penalty of Tibshirani for fitting quantile regression models. When the argument `lambda` is a scalar the penalty function is the  $l_1$  norm of the last  $(p-1)$  coefficients, under the presumption that the first coefficient is an intercept parameter that should not be subject to the penalty. When `lambda` is a vector it should have length equal the column dimension of the matrix `x` and then defines a coordinatewise specific vector of lasso penalty parameters. In this case `lambda` entries of zero indicate covariates that are not penalized. There should be a sparse version of this, but isn't (yet).

### Usage

```
rq.fit.lasso(x, y, tau = 0.5, lambda = 1, beta = .9995, eps = 1e-06)
```

**Arguments**

x	the design matrix
y	the response variable
tau	the quantile desired, defaults to 0.5.
lambda	the value of the penalty parameter(s) that determine how much shrinkage is done. This should be either a scalar, or a vector of length equal to the column dimension of the x matrix.
beta	step length parameter for Frisch-Newton method.
eps	tolerance parameter for convergence.

**Value**

Returns a list with a coefficient, residual, tau and lambda components. When called from "rq" (as intended) the returned object has class "lassorqs".

**Author(s)**

R. Koenker

**References**

Koenker, R. (2005 *Quantile Regression*, CUP.

**See Also**

[rq](#)

**Examples**

```
n <- 60
p <- 7
rho <- .5
beta <- c(3,1.5,0,2,0,0,0)
R <- matrix(0,p,p)
for(i in 1:p){
  for(j in 1:p){
    R[i,j] <- rho^abs(i-j)
  }
}
set.seed(1234)
x <- matrix(rnorm(n*p),n,p) %*% t(chol(R))
y <- x %*% beta + rnorm(n)

f <- rq(y ~ x, method="lasso",lambda = 30)
g <- rq(y ~ x, method="lasso",lambda = c(rep(0,4),rep(30,4)))
```

---

`rq.fit.pfn`*Preprocessing Algorithm for Quantile Regression*

---

**Description**

A preprocessing algorithm for the Frisch Newton algorithm for quantile regression. This is one possible method for `rq()`.

**Usage**

```
rq.fit.pfn(x, y, tau=0.5, Mm.factor=0.8, max.bad.fixup=3, eps=1e-06)
```

**Arguments**

<code>x</code>	design matrix usually supplied via <code>rq()</code>
<code>y</code>	response vector usually supplied via <code>rq()</code>
<code>tau</code>	quantile of interest
<code>Mm.factor</code>	constant to determine sub sample size <code>m</code>
<code>max.bad.fixup</code>	number of allowed mispredicted signs of residuals
<code>eps</code>	convergence tolerance

**Details**

Preprocessing algorithm to reduce the effective sample size for QR problems with (plausibly) iid samples. The preprocessing relies on subsampling of the original data, so situations in which the observations are not plausibly iid, are likely to cause problems. The tolerance `eps` may be relaxed somewhat.

**Value**

Returns an object of type `rq`

**Author(s)**

Roger Koenker <rkoenker@uiuc.edu>

**References**

Portnoy and Koenker, *Statistical Science*, (1997) 279-300

**See Also**

[rq](#)

rq.fit.scad

*SCADPenalized Quantile Regression***Description**

The fitting method implements the smoothly clipped absolute deviation penalty of Fan and Li for fitting quantile regression models. When the argument `lambda` is a scalar the penalty function is the scad modified  $l_1$  norm of the last  $(p-1)$  coefficients, under the presumption that the first coefficient is an intercept parameter that should not be subject to the penalty. When `lambda` is a vector it should have length equal the column dimension of the matrix `x` and then defines a coordinatewise specific vector of scad penalty parameters. In this case `lambda` entries of zero indicate covariates that are not penalized. There should be a sparse version of this, but isn't (yet).

**Usage**

```
rq.fit.scad(x, y, tau = 0.5, alpha = 3.2, lambda = 1, start="rq",
beta = .9995, eps = 1e-06)
```

**Arguments**

<code>x</code>	the design matrix
<code>y</code>	the response variable
<code>tau</code>	the quantile desired, defaults to 0.5.
<code>alpha</code>	tuning parameter of the scad penalty.
<code>lambda</code>	the value of the penalty parameter that determines how much shrinkage is done. This should be either a scalar, or a vector of length equal to the column dimension of the <code>x</code> matrix.
<code>start</code>	starting method, default method 'rq' uses the unconstrained rq estimate, while method 'lasso' uses the corresponding lasso estimate with the specified lambda.
<code>beta</code>	step length parameter for Frisch-Newton method.
<code>eps</code>	tolerance parameter for convergence.

**Details**

The algorithm is an adaptation of the "difference convex algorithm" described in Wu and Liu (2008). It solves a sequence of (convex) QR problems to approximate solutions of the (non-convex) scad problem.

**Value**

Returns a list with a coefficient, residual, tau and lambda components. When called from "rq" as intended the returned object has class "scadrqs".

**Author(s)**

R. Koenker

## References

Wu, Y. and Y. Liu (2008) Variable Selection in Quantile Regression, *Statistica Sinica*, to appear.

## See Also

[rq](#)

## Examples

```
n <- 60
p <- 7
rho <- .5
beta <- c(3,1.5,0,2,0,0,0)
R <- matrix(0,p,p)
for(i in 1:p){
  for(j in 1:p){
    R[i,j] <- rho^abs(i-j)
  }
}
set.seed(1234)
x <- matrix(rnorm(n*p),n,p) %*% t(chol(R))
y <- x %*% beta + rnorm(n)

f <- rq(y ~ x, method="scad",lambda = 30)
g <- rq(y ~ x, method="scad", start = "lasso", lambda = 30)
```

---

rq.fit.sfn

*Sparse Regression Quantile Fitting*

---

## Description

Fit a quantile regression model using a sparse implementation of the Frisch-Newton interior-point algorithm.

## Usage

```
rq.fit.sfn(a, y, tau = 0.5, rhs = (1-tau)*c(t(a) %*% rep(1,length(y))),
  nsubmax, tmpmax, nnzlmax, cachsz = 64,
  small = 1e-06, maxiter = 100, warn.mesg = TRUE)
```

## Arguments

a	structure of the design matrix X stored in csr format
y	response vector
tau	desired quantile
rhs	the right-hand-side of the dual problem; regular users shouldn't need to specify this.

nsubmax	upper bound of the dimension of lindx in Cholesky factorization; computed automatically inside the routine if missing.
tmpmax	upper bound of the working array in Cholesky factorization; computed automatically inside the routine if missing.
nnzlmax	upper bound of the non-zero entries in the Cholesky factor $L$ ; computed automatically inside the routine if missing.
cachsz	size of the cache on the machine; default to 64.
small	convergence tolerance for the interior algorithm.
maxiter	upper limit for the number of iterations.
warn.mesg	flag for printing of warning message; default to TRUE

### Details

This is a sparse implementation of the Frisch-Newton algorithm for quantile regression described in Koenker and Portnoy (1996). The sparse matrix linear algebra is implemented through the functions available in the R package **SparseM**.

### Value

coef	Regression quantile coefficients
ierr	Error code for the internal Fortran routine <code>srqfnc</code> : <ol style="list-style-type: none"> <li><b>1:</b> insufficient work space in call to <code>extract</code></li> <li><b>2:</b> <code>nnzd &gt; nnzdmax</code></li> <li><b>3:</b> insufficient storage in <code>iwork</code> when calling <code>ordmmd</code></li> <li><b>4:</b> insufficient storage in <code>iwork</code> when calling <code>sfinit</code></li> <li><b>5:</b> <code>nnzl &gt; nnzlmax</code> when calling <code>sfinit</code></li> <li><b>6:</b> <code>nsub &gt; nsubmax</code> when calling <code>sfinit</code></li> <li><b>7:</b> insufficient work space in <code>iwork</code> when calling <code>symfct</code></li> <li><b>8:</b> inconsistency in input when calling <code>symfct</code></li> <li><b>9:</b> <code>tmpsiz &gt; tmpmax</code> when calling <code>bfinit</code>; increase <code>tmpmax</code></li> <li><b>10:</b> nonpositive diagonal encountered, not positive definite</li> <li><b>11:</b> insufficient work storage in <code>tmpvec</code> when calling <code>blkfct</code></li> <li><b>12:</b> insufficient work storage in <code>iwork</code> when calling <code>blkfct</code></li> <li><b>17:</b> tiny diagonals replaced with <code>Inf</code> when calling <code>blkfct</code></li> </ol>
it	Iteration count
time	Amount of time used in the computation

### Author(s)

Pin Ng

### References

Koenker, R and Ng, P. (2002). SparseM: A Sparse Matrix Package for R,  
<http://www.econ.uiuc.edu/~roger/research>

**See Also**

rq.fit.sfnc for the constrained version, SparseM for a sparse matrix package for R

**Examples**

```
## An artificial example :
n <- 200
p <- 50
set.seed(101)
X <- rnorm(n*p)
X[abs(X) < 2.0] <- 0
X <- cbind(1, matrix(X, n, p))
y <- 0.5 * apply(X,1,sum) + rnorm(n) ## true beta = (0.5, 0.5, ...)

sX <- as.matrix.csr(X)
try(rq.o <- rq.fit.sfn(sX, y)) #-> not enough tmp memory
(tmpmax <- floor(1e5 + exp(-12.1)*(sX@ia[p+1]-1)^2.35))
## now ok:
rq.o <- rq.fit.sfn(sX, y, tmpmax = tmpmax)
str(rq.o)
```

---

 rq.fit.sfnc

*Sparse Constrained Regression Quantile Fitting*


---

**Description**

Fit constrained regression quantiles using a sparse implementation of the Frisch-Newton Interior-point algorithm.

**Usage**

```
rq.fit.sfnc(x, y, R, r, tau = 0.5,
            rhs = (1-tau)*c(t(x) %**% rep(1,length(y))),
            nsubmax, tmpmax, nnzlmax, cachsz = 64,
            small = 1e-08, maxiter = 100, warn.mesg = TRUE)
```

**Arguments**

x	structure of the design matrix X stored in csr format
y	response vector
R	constraint matrix stored in csr format
r	right-hand-side of the constraint
tau	desired quantile
rhs	the right-hand-side of the dual problem; regular users shouldn't need to specify this.
nsubmax	upper bound of the dimension of lindx in Cholesky factorization; computed automatically inside the routine if missing.

tmpmax	upper bound of the working array in Cholesky factorization; computed automatically inside the routine if missing
nnzlmax	upper bound of the non-zero entries in the Cholesky factor $L$ ; computed automatically inside the routine if missing
cachsz	size of the cache on the machine; default to 64.
small	convergence tolerance for the interior algorithm.
maxiter	upper limit for the number of iterations.
warn.mesg	flag for printing of warning message; default to TRUE

### Details

This is a sparse implementation of the Frisch-Newton algorithm for constrained quantile regression described in Koenker and Portnoy (1996). The sparse matrix linear algebra is implemented through the functions available in the R package **SparseM**.

### Value

coef	Regression quantile coefficients
ierr	Error code for the internal Fortran routine <code>srqfn</code> : <b>1:</b> insufficient work space in call to <code>extract</code> <b>3:</b> insufficient storage in <code>iwork</code> when calling <code>ordmmd</code> <b>4:</b> insufficient storage in <code>iwork</code> when calling <code>sfinit</code> <b>5:</b> <code>nnzl &gt; nnzlmax</code> when calling <code>sfinit</code> <b>6:</b> <code>nsub &gt; nsubmax</code> when calling <code>sfinit</code> <b>7:</b> insufficient work space in <code>iwork</code> when calling <code>symfct</code> <b>8:</b> inconsistency in input when calling <code>symfct</code> <b>9:</b> <code>tmpsiz &gt; tmpmax</code> when calling <code>symfct</code> ; increase <code>tmpmax</code> <b>10:</b> nonpositive diagonal encountered when calling <code>blkfct</code> <b>11:</b> insufficient work storage in <code>tmpvec</code> when calling <code>blkfct</code> <b>12:</b> insufficient work storage in <code>iwork</code> when calling <code>blkfct</code> <b>13:</b> <code>nnzd &gt; nnzdmax</code> in <code>e,je</code> when calling <code>amub</code> <b>14:</b> <code>nnzd &gt; nnzdmax</code> in <code>g,jg</code> when calling <code>amub</code> <b>15:</b> <code>nnzd &gt; nnzdmax</code> in <code>h,jh</code> when calling <code>aplb</code> <b>15:</b> tiny diagonals replaced with <code>Inf</code> when calling <code>blkfct</code>
it	Iteration count
time	Amount of time used in the computation

### Author(s)

Pin Ng

### References

- Koenker, R and Ng, P. (2002). SparseM: A Sparse Matrix Package for R;  
<http://www.econ.uiuc.edu/~roger/research>  
 Koenker, R. and P. Ng(2005). Inequality Constrained Quantile Regression, *Sankhya*, 418-440.

**See Also**

[rq.fit.sfn](#) for the unconstrained version, **SparseM** for the underlying sparse matrix R package.

**Examples**

```
## An artificial example :
n <- 200
p <- 50
set.seed(17)
X <- rnorm(n*p)
X[abs(X) < 2.0] <- 0
X <- cbind(1, matrix(X, n, p))
y <- 0.5 * apply(X, 1, sum) + rnorm(n) ## true beta = (0.5, 0.5, ...)
R <- rbind(diag(p+1), -diag(p+1))
r <- c(rep( 0, p+1), rep(-1, p+1))

sX <- as.matrix.csr(X)
sR <- as.matrix.csr(R)
try(rq.o <- rq.fit.sfnc(sX, y, sR, r)) #-> not enough tmp memory

(tmpmax <- floor(1e5 + exp(-12.1)*(sX@ia[p+1]-1)^2.35))
## now ok:
rq.o <- rq.fit.sfnc(sX, y, sR, r, tmpmax = tmpmax)
str(rq.o)
```

---

 rq.object

---

*Linear Quantile Regression Object*


---

**Description**

These are objects of class "rq". They represent the fit of a linear conditional quantile function model.

**Details**

The coefficients, residuals, and effects may be extracted by the generic functions of the same name, rather than by the \$ operator. For pure rq objects this is less critical than for some of the inheritor classes. In particular, for fitted rq objects using "lasso" and "scad" penalties, logLik and AIC functions compute degrees of freedom of the fitted model as the number of estimated parameters whose absolute value exceeds a threshold `edfThresh`. By default this threshold is 0.0001, but this can be passed via the AIC function if this value is deemed unsatisfactory. The function AIC is a generic function in R, with parameter `k` that controls the form of the penalty: the default value of `k` is 2 which yields the classical Akaike form of the penalty, while `k <= 0` yields the Schwarz (BIC) form of the penalty. Note that the extractor function `coef` returns a vector with missing values omitted.

**Generation**

This class of objects is returned from the `rq` function to represent a fitted linear quantile regression model.

## Methods

The "rq" class of objects has methods for the following generic functions: `coef`, `effects`, `formula`, `labels`, `model.frame`, `model.matrix`, `plot`, `logLik`, `AIC`, `extractAIC`, `predict`, `print`, `print.summary`, `residuals`, `summary`

## Structure

The following components must be included in a legitimate `rq` object.

`coefficients` the coefficients of the quantile regression fit. The names of the coefficients are the names of the single-degree-of-freedom effects (the columns of the model matrix). If the model was fitted by method "br" with `ci=TRUE`, then the coefficient component consists of a matrix whose first column consists of the vector of estimated coefficients and the second and third columns are the lower and upper limits of a confidence interval for the respective coefficients.

`residuals` the residuals from the fit.

`dual` the vector dual variables from the fit.

`rho` The value(s) of objective function at the solution.

`contrasts` a list containing sufficient information to construct the contrasts used to fit any factors occurring in the model. The list contains entries that are either matrices or character vectors. When a factor is coded by contrasts, the corresponding contrast matrix is stored in this list. Factors that appear only as dummy variables and variables in the model that are matrices correspond to character vectors in the list. The character vector has the level names for a factor or the column labels for a matrix.

`model` optionally the model frame, if `model=TRUE`.

`x` optionally the model matrix, if `x=TRUE`.

`y` optionally the response, if `y=TRUE`.

## See Also

[rq.coefficients](#).

---

`rq.process.object` *Linear Quantile Regression Process Object*

---

## Description

These are objects of class `rq.process`. They represent the fit of a linear conditional quantile function model.

## Details

These arrays are computed by parametric linear programming methods using the exterior point (simplex-type) methods of the Koenker-d'Orey algorithm based on Barrodale and Roberts median regression algorithm.

## Generation

This class of objects is returned from the `rq` function to represent a fitted linear quantile regression model.

## Methods

The "`rq.process`" class of objects has methods for the following generic functions: `effects`, `formula`, `labels`, `model.frame`, `model.matrix`, `plot`, `predict`, `print`, `print.summary`, `summary`.

## Structure

The following components must be included in a legitimate `rq.process` object.

`sol` The primal solution array. This is a  $(p+3)$  by  $J$  matrix whose first row contains the 'break-points'  $\tau_{1}, \tau_{2}, \dots, \tau_{J}$ , of the quantile function, i.e. the values in  $[0,1]$  at which the solution changes, row two contains the corresponding quantiles evaluated at the mean design point, i.e. the inner product of  $\bar{x}$  and  $b(\tau_{i})$ , the third row contains the value of the objective function evaluated at the corresponding  $\tau_{j}$ , and the last  $p$  rows of the matrix give  $b(\tau_{i})$ . The solution  $b(\tau_{i})$  prevails from  $\tau_{i}$  to  $\tau_{i} + 1$ . Portnoy (1991) shows that  $J = O_p(n \log n)$ .

`dsol` The dual solution array. This is a  $n$  by  $J$  matrix containing the dual solution corresponding to `sol`, the  $ij$ -th entry is 1 if  $y_i > x_i b(\tau_j)$ , is 0 if  $y_i < x_i b(\tau_j)$ , and is between 0 and 1 otherwise, i.e. if the residual is zero. See Gutenbrunner and Jureckova(1991) for a detailed discussion of the statistical interpretation of `dsol`. The use of `dsol` in inference is described in Gutenbrunner, Jureckova, Koenker, and Portnoy (1994).

## References

- [1] Koenker, R. W. and Bassett, G. W. (1978). Regression quantiles, *Econometrica*, **46**, 33–50.
- [2] Koenker, R. W. and d'Orey (1987, 1994). Computing Regression Quantiles. *Applied Statistics*, **36**, 383–393, and **43**, 410–414.
- [3] Gutenbrunner, C. Jureckova, J. (1991). Regression quantile and regression rank score process in the linear model and derived statistics, *Annals of Statistics*, **20**, 305–330.
- [4] Gutenbrunner, C., Jureckova, J., Koenker, R. and Portnoy, S. (1994) Tests of linear hypotheses based on regression rank scores. *Journal of Nonparametric Statistics*, (2), 307–331.
- [5] Portnoy, S. (1991). Asymptotic behavior of the number of regression quantile breakpoints, *SIAM Journal of Scientific and Statistical Computing*, **12**, 867–883.

## See Also

[rq](#).

---

`rq.wfit`*Function to choose method for Weighted Quantile Regression*

---

**Description**

Weight the data and then call the chosen fitting algorithm.

**Usage**

```
rq.wfit(x, y, tau=0.5, weights, method="br", ...)
```

**Arguments**

<code>x</code>	the design matrix
<code>y</code>	the response variable
<code>tau</code>	the quantile desired, if tau lies outside (0,1) the whole process is estimated.
<code>weights</code>	weights used in the fitting
<code>method</code>	method of computation: "br" is Barrodale and Roberts exterior point "fn" is the Frisch-Newton interior point method.
<code>...</code>	Optional arguments passed to fitting routine.

**See Also**

[rq](#) [rq.fit.br](#) [rq.fit.fnb](#)

---

`rqProcess`*Compute Standardized Quantile Regression Process*

---

**Description**

Computes a standardize quantile regression process for the model specified by the formula, on the partition of [0,1] specified by the taus argument, and standardized according to the argument nullH. Intended for use in [KhmaldzeTest](#).

**Usage**

```
rqProcess(formula, data, taus, nullH = "location", ...)
```

**Arguments**

<code>formula</code>	model formula
<code>data</code>	data frame to be used to interpret formula
<code>taus</code>	quantiles at which the process is to be evaluated, if any of the taus lie outside (0,1) then the full process is computed for all distinct solutions.
<code>nullH</code>	Null hypothesis to be used for standardization
<code>...</code>	optional arguments passed to <a href="#">summary.rq</a>

**Details**

The process computes standardized estimates based on the hypothesis specified in the `nullH` argument. The `Vhat` component is rescaled by the Cholesky decomposition of the tau specific covariance matrix, the `vhat` component is rescaled by the marginal standard errors. The nature of the covariance matrix used for the standardization is controlled arguments passed via the `...` argument to [summary.rq](#). If the full process is estimated then these covariance options aren't available and only a simple iid-error form of the covariance matrix is used.

**Value**

<code>taus</code>	The points of evaluation of the process
<code>qtaus</code>	Values of <code>xbar'betahat(taus)</code>
<code>Vhat</code>	Joint parametric QR process
<code>vhat</code>	Marginal parametric QR processes

**Author(s)**

R. Koenker

**See Also**

[KhmaladzeTest](#)

---

`rqss`

*Additive Quantile Regression Smoothing*

---

**Description**

Fitting function for additive quantile regression models with possible univariate and/or bivariate nonparametric terms estimated by total variation regularization.

**Usage**

```
rqss(formula, tau = 0.5, data = parent.frame(), weights, na.action,
method = "sfn", lambda = NULL, contrasts = NULL, ztol = 1e-5, ...)
```

**Arguments**

<code>formula</code>	a formula object, with the response on the left of a '~' operator, and terms, separated by '+' operators, on the right. The terms may include <code>qss</code> terms that represent additive nonparametric components. These terms can be univariate or bivariate. See <a href="#">qss</a> for details on how to specify these terms.
<code>tau</code>	the quantile to be estimated, this must be a number between 0 and 1,
<code>data</code>	a <code>data.frame</code> in which to interpret the variables named in the formula, or in the subset and the weights argument.

<code>weights</code>	vector of observation weights; if supplied, the algorithm fits to minimize the sum of the weights multiplied into the absolute residuals. The length of weights must be the same as the number of observations. The weights must be nonnegative and it is strongly recommended that they be strictly positive, since zero weights are ambiguous.
<code>na.action</code>	a function to filter missing data. This is applied to the model.frame after any subset argument has been used. The default (with <code>na.fail</code> ) is to create an error if any missing values are found. A possible alternative is <code>na.omit</code> , which deletes observations that contain one or more missing values.
<code>method</code>	the algorithmic method used to compute the fit. There are currently two options. Both are implementations of the Frisch–Newton interior point method described in detail in Portnoy and Koenker(1997). Both are implemented using sparse Cholesky decomposition as described in Koenker and Ng (2003). Option <code>"sfnc"</code> is used if the user specifies inequality constraints. Option <code>"sfn"</code> is used if there are no inequality constraints. Linear inequality constraints on the fitted coefficients are specified by a matrix $R$ and a vector $r$ , specified inside the <code>qss</code> terms, representing the constraints in the form $Rb \geq r$ . The option <code>method = "lasso"</code> allows one to penalize the coefficients of the covariates that have been entered linearly as in <code>rq.fit.lasso</code> ; when this is specified then there should be an additional <code>lambda</code> argument specified that determines the amount of shrinkage.
<code>lambda</code>	can be either a scalar, in which case all the slope coefficients are assigned this value, or alternatively, the user can specify a vector of length equal to the number of linear covariates plus one (for the intercept) and these values will be used as coordinate dependent shrinkage factors.
<code>contrasts</code>	a list giving contrasts for some or all of the factors <code>default = NULL</code> appearing in the model formula. The elements of the list should have the same name as the variable and should be either a contrast matrix (specifically, any full-rank matrix with as many rows as there are levels in the factor), or else a function to compute such a matrix given the number of levels.
<code>ztol</code>	A zero tolerance parameter used to determine the number of zero residuals in the fitted object which in turn determines the effective dimensionality of the fit.
<code>...</code>	additional arguments for the fitting routines (see <code>rq.fit.sfn</code> and <code>rq.fit.sfnc</code> and the functions they call).

## Details

Total variation regularization for univariate and bivariate nonparametric quantile smoothing is described in Koenker, Ng and Portnoy (1994) and Koenker and Mizera(2003) respectively. The additive model extension of this approach depends crucially on the sparse linear algebra implementation for R described in Koenker and Ng (2003). There are extractor methods `logLik` and `AIC` that is relevant to lambda selection. A more detailed description of some recent developments of these methods is available from within the package with `faux.vignette("rqss")`.

## Value

The function returns a fitted object representing the estimated model specified in the formula. See `rqss.object` for further details on this object, and references to methods to look at it.

**Author(s)**

Roger Koenker

**References**

- [1] Koenker, R. and S. Portnoy (1997) The Gaussian Hare and the Laplacean Tortoise: Computability of Squared-error vs Absolute Error Estimators, (with discussion). *Statistical Science* **12**, 279–300.
- [2] Koenker, R., P. Ng and S. Portnoy, (1994) Quantile Smoothing Splines; *Biometrika* **81**, 673–680.
- [3] Koenker, R. and I. Mizera, (2003) Penalized Triograms: Total Variation Regularization for Bivariate Smoothing; *JRSS(B)* **66**, 145–163.
- [4] Koenker, R. and P. Ng (2003) SparseM: A Sparse Linear Algebra Package for R, *J. Stat. Software*.

**See Also**[qss](#)**Examples**

```
n <- 200
x <- sort(rchisq(n,4))
z <- x + rnorm(n)
y <- log(x) + .1*(log(x))^2 + log(x)*rnorm(n)/4 + z
plot(x, y-z)
f.N <- rqss(y ~ qss(x, constraint= "N") + z)
f.I <- rqss(y ~ qss(x, constraint= "I") + z)
f.CI <- rqss(y ~ qss(x, constraint= "CI") + z)

lines(x[-1], f.N $coef[1] + f.N $coef[-(1:2)])
lines(x[-1], f.I $coef[1] + f.I $coef[-(1:2)], col="blue")
lines(x[-1], f.CI$coef[1] + f.CI$coef[-(1:2)], col="red")

## A bivariate example
data(CobarOre)
fCO <- rqss(z ~ qss(cbind(x,y), lambda= .08), data=CobarOre)
plot(fCO)
```

---

rqss.object*RQSS Objects and Summarization Thereof*

---

**Description**Functions to reveal the inner meaning of objects created by `rqss` fitting.

**Usage**

```
## S3 method for class 'rqss':
logLik(object, ...)
## S3 method for class 'rqss':
AIC(object, ..., k=2)
## S3 method for class 'rqss':
print(x, ...)
```

**Arguments**

object	an object returned from <code>rqss</code> fitting, describing an additive model estimating a conditional quantile function. See <code>qss</code> for details on how to specify these terms.
x	an <code>rqss</code> object, as above.
k	a constant factor governing the weight attached to the penalty term on effective degrees of freedom of the fit. By default $k = 2$ corresponding to the Akaike version of the penalty, negative values indicate that the $k$ should be set to $\log(n)$ as proposed by Schwarz (1978).
...	additional arguments

**Details**

Total variation regularization for univariate and bivariate nonparametric quantile smoothing is described in Koenker, Ng and Portnoy (1994) and Koenker and Mizera(2003) respectively. The additive model extension of this approach depends crucially on the sparse linear algebra implementation for R described in Koenker and Ng (2003). Eventually, these functions should be expanded to provide an automated lambda selection procedure.

**Value**

The function `summary.rqss` returns a list consisting of the following components:

<code>fidelity</code>	Value of the quantile regression objective function.
<code>penalty</code>	A list consisting of the values of the total variation smoothing penalty for each of additive components.
<code>edf</code>	Effective degrees of freedom of the fitted model, defined as the number of zero residuals of the fitted model, Koenker Mizera (2003) for details.
<code>qssedfs</code>	A list of effective degrees of freedom for each of the additive components of the fitted model, defined as the number of non-zero elements of each penalty component of the residual vector.
<code>lamdas</code>	A list of the lambdas specified for each of the additive components of the model.

**Author(s)**

Roger Koenker

## References

- [1] Koenker, R. and S. Portnoy (1997) The Gaussian Hare and the Laplacean Tortoise: Computability of Squared-error vs Absolute Error Estimators, (with discussion). *Statistical Science* **12**, 279–300.
- [2] Koenker, R., P. Ng and S. Portnoy, (1994) Quantile Smoothing Splines; *Biometrika* **81**, 673–680.
- [3] Koenker, R. and I. Mizera, (2003) Penalized Triograms: Total Variation Regularization for Bivariate Smoothing; *JRSS(B)* **66**, 145–163.
- [4] Koenker, R. and P. Ng (2003) SparseM: A Sparse Linear Algebra Package for R, *J. Stat. Software*.

## See Also

[plot.rqss](#)

## Examples

```
n <- 200
x <- sort(rchisq(n, 4))
z <- x + rnorm(n)
y <- log(x) + .1*(log(x))^2 + log(x)*rnorm(n)/4 + z
plot(x, y-z)
f.N <- rqss(y ~ qss(x, constraint= "N") + z)
f.I <- rqss(y ~ qss(x, constraint= "I") + z)
f.CI <- rqss(y ~ qss(x, constraint= "CI") + z)

lines(x[-1], f.N $coef[1] + f.N $coef[-(1:2)])
lines(x[-1], f.I $coef[1] + f.I $coef[-(1:2)], col="blue")
lines(x[-1], f.CI$coef[1] + f.CI$coef[-(1:2)], col="red")

## A bivariate example
data(CobarOre)
fCO <- rqss(z ~ qss(cbind(x,y), lambda= .08), data=CobarOre)
plot(fCO)
```

## Description

Returns a summary object for a censored quantile regression fit. A null value will be returned if printing is invoked.

**Usage**

```
## S3 method for class 'crq':
summary(object, taus = 1:4/5, alpha = .05, se="boot", covariance=TRUE, ...)
## S3 method for class 'summary.crq':
print(x, digits = max(5, .Options$digits - 2), ...)
## S3 method for class 'summary.crq.s':
print(x, ...)
## S3 method for class 'summary.crq.s':
plot(x, nrow = 3, ncol = 3, CoxPHit = NULL, ...)
```

**Arguments**

object	An object of class "crq" produced by a call to <code>crq()</code> .
x	An object of class "crq" produced by a call to <code>crq()</code> .
se	specifies the method used to compute standard standard errors. but the only available method (so far) is "boot". Further arguments to <code>boot.crq</code> and <code>boot.rq</code> can be passed with the <code>...</code> argument.
covariance	logical flag to indicate whether the full covariance matrix of the estimated parameters should be returned.
nrow	Number of rows of the plot layout.
ncol	Number of columns of the plot layout.
taus	Quantiles to be summarized.
alpha	Confidence level for summary intervals.
digits	Number of digits to be printed in summary display.
CoxPHit	An object of class <code>coxph</code> produced by <code>coxph</code> .
...	Optional arguments to <code>summary</code> , e.g. to specify bootstrap methods sample sizes, etc. see <code>boot.rq</code> and <code>boot.crq</code>

**Details**

For the Fitzenberger-Powell method the resampling strategy used by the `se = "boot"` method is based on the Biliias, Chen and Ying (2000) proposal. For the Portnoy and Peng-Huang methods the bootstrapping is based on resampling  $xy$  pairs using either conventional multinomial resampling or using exponential weighting as in Bose and Chatterjee (2003). Note that the default number of replications is set at  $R = 100$  a value that is obviously too small for most applications. This is done merely to speed up the examples in the documentation and facilitate testing. Larger, more appropriate values of  $R$  can be passed to the bootstrapping functions via the `...` argument of the `summary` method.

**Value**

For method "FP" an object of class `summary.crqFP` is returned with the following components:

coefficients	a $p$ by 4 matrix consisting of the coefficients, their estimated standard errors, their t-statistics, and their associated p-values.
--------------	---

cov	the estimated covariance matrix for the coefficients in the model, provided that <code>COV=TRUE</code> in the called sequence.
rdf	the residual degrees of freedom
tau	the quantile estimated
cov	the estimated covariance matrix for the coefficients in the model, provided that <code>COV=TRUE</code> in the called sequence.

For method "PH" an object of class `summary.crq` is returned with the following components:

`coefficients` a list of `p` by 6 matrix consisting of the coefficients, upper and lower bounds for a (1-alpha) level confidence interval, their estimated standard errors, their t-statistics, and their associated p-values, one component for each element of the specified `taus` vector.

## References

Bose, A. and S. Chatterjee, (2003) Generalized bootstrap for estimators of minimizers of convex functions, *J. Stat. Planning and Inf*, 117, 225-239.

Biliias, Y. Chen, S. and Z. Ying, (2000) Simple resampling methods for censored quantile regression, *J. of Econometrics*, 99, 373-386.

## See Also

[crq](#), [QTECox](#)

---

summary.rq

*Summary methods for Quantile Regression*

---

## Description

Returns a summary list for a quantile regression fit. A null value will be returned if printing is invoked.

## Usage

```
## S3 method for class 'rq':
summary(object, se="nid", covariance=FALSE, hs = TRUE, ...)
## S3 method for class 'rqs':
summary(object, ...)
```

## Arguments

object	This is an object of class "rq" or "rqs" produced by a call to <code>rq()</code> , depending on whether one or more <code>taus</code> are specified.
se	specifies the method used to compute standard standard errors. There are currently five available methods:

1. "rank" which produces confidence intervals for the estimated parameters by inverting a rank test as described in Koenker (1994). The default option assumes that the errors are iid, while the option `iid = FALSE` implements the proposal of Koenker Machado (1999). This is the default method unless the sample size exceeds 1000, or `cov = FALSE` in which case `se = "nid"` is used.
  2. "iid" which presumes that the errors are iid and computes an estimate of the asymptotic covariance matrix as in KB(1978).
  3. "nid" which presumes local (in  $\tau$ ) linearity (in  $x$ ) of the the conditional quantile functions and computes a Huber sandwich estimate using a local estimate of the sparsity.
  4. "ker" which uses a kernel estimate of the sandwich as proposed by Powell(1990).
  5. "boot" which implements one of several possible bootstrapping alternatives for estimating standard errors.
- `covariance` logical flag to indicate whether the full covariance matrix of the estimated parameters should be returned.
- `hs` Use Hall Sheather bandwidth for sparsity estimation If false revert to Bofinger bandwidth.
- ... Optional arguments to summary, e.g. `bsmethod` to use bootstrapping. see [boot.rq](#)

## Details

When the default summary method is used, it tries to estimate a sandwich form of the asymptotic covariance matrix and this involves estimating the conditional density at each of the sample observations, negative estimates can occur if there is crossing of the neighboring quantile surfaces used to compute the difference quotient estimate. A warning message is issued when such negative estimates exist indicating the number of occurrences – if this number constitutes a large proportion of the sample size, then it would be prudent to consider an alternative inference method like the bootstrap. If the number of these is large relative to the sample size it is sometimes an indication that some additional nonlinearity in the covariates would be helpful, for instance quadratic effects. Note that the default `se` method is `rank`, unless the sample size exceeds 1001, in which case the `rank` method is used.

## Value

a list is returned with the following components, when `object` is of class "rq" then there is a list of such lists.

- `coefficients` a  $p$  by 4 matrix consisting of the coefficients, their estimated standard errors, their t-statistics, and their associated p-values.
- `cov` the estimated covariance matrix for the coefficients in the model, provided that `cov=TRUE` in the called sequence.
- `Hinv` inverse of the estimated Hessian matrix returned if `cov=TRUE` and `se != "iid"`.
- `J` Outer product of gradient matrix returned if `cov=TRUE` and `se != "iid"`. The Huber sandwich is `cov = Hinv %*% J %*% Hinv`.

## References

Koenker, R. (2004) *Quantile Regression*.

Biliyas, Y. Chen, S. and Z. Ying, Simple resampling methods for censored quantile regression, *J. of Econometrics*, 99, 373-386.

## See Also

[rq.bandwidth.rq](#)

## Examples

```
data(stackloss)
y <- stack.loss
x <- stack.x
summary(rq(y ~ x, method="fn")) # Compute se's for fit using "nid" method.
summary(rq(y ~ x, ci=FALSE), se="ker")
# default "br" alg, and compute kernel method se's
```

---

summary.rqss

*Summary of rqss fit*

---

## Description

Summary Method for a fitted rqss model.

## Usage

```
## S3 method for class 'rqss':
summary(object, cov = FALSE, ztol = 1e-5, ...)
```

## Arguments

object	an object returned from <code>rqss</code> fitting, describing an additive model estimating a conditional quantile function. See <code>rqss</code> for details on how to specify these terms.
cov	if TRUE return covariance matrix for the parametric components as $V_{cov}$ and a list of covariance matrices for the nonparametric components as $V_{qss}$
ztol	Zero tolerance parameter used to determine the number of zero residuals indicating the estimated parametric dimension of the model, the so-called effective degrees of freedom.
...	additional arguments

## Details

This is a highly experimental function intended to explore inferential methods for rqss fitting. The function is modeled after `summary.gam` in Simon Wood's (2006) **mgcv** package. (Of course, Simon should not be blamed for any deficiencies in the current implementation. The basic idea is to condition on the lambda selection and construct quasi-Bayesian credibility intervals based on normal approximation of the "posterior," as computed using the Powell kernel estimate of the usual quantile regression sandwich. See [summary.rq](#) for further details and references. The function produces a conventional coefficient table with standard errors t-statistics and p-values for the coefficients on the parametric part of the model, and another table for additive nonparametric effects. The latter reports F statistics intended to evaluate the significance of these components individually. In addition the fidelity (value of the QR objective function evaluated at the fitted model), the effective degrees of freedom, and the sample size are reported.

## Value

<code>coef</code>	Table of estimated coefficients and their standard errors, t-statistics, and p-values for the parametric components of the model
<code>qsstab</code>	Table of approximate F statistics, effective degrees of freedom and values of the penalty terms for each of the additive nonparametric components of the model, and the lambda values assigned to each.
<code>fidelity</code>	Value of the quantile regression objective function.
<code>tau</code>	Quantile of the estimated model
<code>formula</code>	formula of the estimated model
<code>edf</code>	Effective degrees of freedom of the fitted model, defined as the number of zero residuals of the fitted model, see Koenker Mizera (2003) for details.
<code>n</code>	The sample size used to fit the model.
<code>Vcov</code>	Estimated covariance matrix of the fitted parametric component
<code>Vqss</code>	List of estimated covariance matrices of the fitted nonparametric component

## Author(s)

Roger Koenker

## References

- [1] Koenker, R., P. Ng and S. Portnoy, (1994) Quantile Smoothing Splines; *Biometrika* **81**, 673–680.
- [2] Koenker, R. and I. Mizera, (2003) Penalized Triograms: Total Variation Regularization for Bivariate Smoothing; *JRSS(B)* **66**, 145–163.
- [3] Wood, S. (2006) *Generalized Additive Models*, Chapman-Hall.

## See Also

[plot.rqss](#)

**Examples**

```
n <- 200
x <- sort(rchisq(n,4))
z <- x + rnorm(n)
y <- log(x) + .1*(log(x))^2 + log(x)*rnorm(n)/4 + z
f <- rqss(y ~ qss(x) + z)
summary(f)
```

---

`table.rq`*Table of Quantile Regression Results*

---

**Description**

Defunct Function to produce a table of quantile regression results for a group of specified quantiles.  
See `rq` which now permits multiple taus.

**Usage**

```
table.rq(x, ...)
```

**Arguments**

<code>x</code>	input
<code>...</code>	other optional arguments

**Value**

None.

**See Also**

[rq](#),

---

`uis`*UIS Drug Treatment study data*

---

**Description**

There are 628 data points in the original data, 575 of which have no missing values.

Variable descriptions:

Variable	Description	Codes/Values
ID	Identification Code	1 - 628
AGE	Age at Enrollment	Years
BECK	Beck DepressionScore	0.000 - 54.000
HC	Heroin/Cocaine Use During 3 Months Prior to Admission	1 = Heroin & Cocaine 2 = Heroin Only 3 = Cocaine Only 4 = Neither Heroin nor Cocaine
IV	History of IV Drug Use	1 = Never 2 = Previous 3 = Recent
NDT	Number of Prior Drug Treatments	0 - 40
RACE	Subject's Race	0 = White 1 = Non-White
TREAT	Treatment Randomization Assignment	0 = Short 1 = Long
SITE	Treatment Site	0 = A 1 = B
LEN.T	Length of Stay in Treatment (Admission Date to Exit Date)	Days
TIME	Time to Drug Relapse (Measured from Admission Date)	Days
CENSOR	Event for Treating Lost to Follow-Up as Returned to Drugs	1 = Returned to Drugs or Lost to Follow-Up 0 = Otherwise
Y	log of TIME	
ND1	Component of NDT	
ND2	Component of NDT	
LNNT		
FRAC	Compliance fraction	LEN.T/90 for short trt LEN.T/180 for long trt
IV3	Recent IV use	1 = Yes 0 = No

**Usage**

```
data(uis)
```

**Format**

A data frame with dimension 575 by 18.

**Source**

Table 1.3 of Hosmer,D.W. and Lemeshow, S. (1998)

**References**

Hosmer, D.W. and Lemeshow, S. (1998) *Applied Survival Analysis: Regression Modeling of Time to Event Data*, John Wiley and Sons Inc., New York, NY

# Index

## \*Topic **IO**

latex, 23  
latex.summary.rqs, 24

## \*Topic **datasets**

barro, 7  
Bosco, 10  
CobarOre, 11  
engel, 19  
Mammals, 29  
Peirce, 33  
uis, 79

## \*Topic **documentation**

FAQ, 20

## \*Topic **environment**

nlrq.control, 32

## \*Topic **hplot**

plot.rq, 34  
plot.rqs, 35  
plot.summary.rqs, 38

## \*Topic **htest**

anova.rq, 3  
KhmaladzeTest, 21

## \*Topic **iplot**

plot.rqss, 36

## \*Topic **manip**

dither, 18

## \*Topic **methods**

lm.fit.recursive, 27

## \*Topic **models**

nlrq, 30  
residuals.nlrq, 49

## \*Topic **nonlinear**

nlrq, 30  
residuals.nlrq, 49

## \*Topic **regression**

anova.rq, 3  
bandwidth.rq, 6  
boot.crq, 7  
boot.rq, 9

critval, 13

crq, 14

nlrq, 30

plot.rqss, 36

predict.rq, 40

predict.rqss, 42

print.rq, 43

print.summary.rq, 44

ranks, 47

rearrange, 48

residuals.nlrq, 49

rq, 50

rq.fit, 53

rq.fit.br, 54

rq.fit.fnb, 55

rq.fit.fnc, 56

rq.fit.lasso, 57

rq.fit.pfn, 59

rq.fit.scad, 60

rq.fit.sfn, 61

rq.fit.sfnc, 63

rq.object, 65

rq.process.object, 66

rq.wfit, 68

rqProcess, 68

rqss, 69

rqss.object, 71

summary.crq, 73

summary.rq, 75

summary.rqss, 77

table.rq, 79

## \*Topic **robust**

anova.rq, 3

lprq, 27

nlrq, 30

predict.rqss, 42

qss, 44

rqss, 69

rqss.object, 71

- summary.rqss, 77
- \*Topic **smooth**
  - akj, 1
  - lprq, 27
  - plot.rqss, 36
  - predict.rqss, 42
  - qss, 44
  - rqss, 69
  - rqss.object, 71
  - summary.rqss, 77
- \*Topic **survival**
  - crq, 14
  - QTECox, 46
- \*Topic **univar**
  - kuantile, 22
- \*Topic **utilities**
  - combos, 12
  - latex.table, 25
- [.terms (rqss), 69
- AIC, 70
- AIC.rq (rq.object), 65
- AIC.rqs (rq.object), 65
- AIC.rqss (rqss.object), 71
- akj, 1
- anova, 48
- anova.rq, 3
- anova.rqlist (anova.rq), 3
- bandwidth.rq, 6, 77
- barro, 7
- boot.crq, 7, 74
- boot.rq, 8, 9, 74, 76
- Bosco, 10
- ChangeLog (FAQ), 20
- CobarOre, 11
- coef.crq (crq), 14
- coef.nlrq (nlrq), 30
- coefficients, 66
- combos, 12
- critval, 13
- crq, 14, 47, 75
- Curv (crq), 14
- dcauchy, 2
- deviance.nlrq (nlrq), 30
- dither, 18
- engel, 19
- extractAIC.rq (rq.object), 65
- FAQ, 20
- faux.vignette (FAQ), 20
- fitted.nlrq (nlrq), 30
- formula.nlrq (nlrq), 30
- formula.rq (rq.object), 65
- jitter, 19
- KhmaladzeTest, 5, 21, 68, 69
- khmaladzize, 27
- khmaladzize (KhmaladzeTest), 21
- kselect (kuantile), 22
- kuantile, 22
- kunique (kuantile), 22
- latex, 23
- latex.summary.rqs, 24, 24
- latex.table, 24, 25
- latex.table.rq (table.rq), 79
- list, 2
- lm, 35, 39
- lm.fit.recursive, 27
- logLik, 70
- logLik.rq (rq.object), 65
- logLik.rqs (rq.object), 65
- logLik.rqss (rqss.object), 71
- lprq, 27
- Mammals, 29
- nlrq, 30, 32, 50, 52
- nlrq.control, 31, 32
- nlrqModel (nlrq), 30
- optim, 30
- Peirce, 33
- plot, 36
- plot.qss1 (plot.rqss), 36
- plot.qss2 (plot.rqss), 36
- plot.rq, 34
- plot.rqs, 35, 39
- plot.rqss, 13, 36, 73, 78
- plot.summary.crqs (summary.crq), 73
- plot.summary.rqs, 36, 38
- plot.table.rq (table.rq), 79
- points, 39

- predict.crq (*crq*), 14
- predict.crqs (*crq*), 14
- predict.nlrq (*nlrq*), 30
- predict.qssl (*predict.rqss*), 42
- predict.qss2 (*predict.rqss*), 42
- predict.rq, 40
- predict.rqs (*predict.rq*), 40
- predict.rqss, 42
- print.anova.rq (*anova.rq*), 3
- print.crq (*crq*), 14
- print.nlrq (*nlrq*), 30
- print.rq, 43
- print.rqs (*print.rq*), 43
- print.rqss (*rqss.object*), 71
- print.summary.crq (*summary.crq*), 73
- print.summary.crqs (*summary.crq*), 73
- print.summary.nlrq (*nlrq*), 30
- print.summary.rq, 44
- print.summary.rqs (*print.summary.rq*), 44
- print.summary.rqss (*summary.rqss*), 77
- qdensity (*KhmaladzeTest*), 21
- qss, 44, 69, 71, 72, 77
- qss1 (*qss*), 44
- qss2 (*qss*), 44
- QTECox, 46, 75
- quantile, 23
- ranks, 4, 5, 47
- rearrange, 41, 48, 49
- residuals.nlrq, 31, 49
- rq, 5, 35, 36, 39, 41, 44, 48, 49, 50, 53, 55–59, 61, 66–68, 77, 79
- rq.fit, 53
- rq.fit.br, 51, 53, 54, 56, 57, 68
- rq.fit.fnb, 51, 53, 55, 55, 68
- rq.fit.fnc, 56
- rq.fit.lasso, 57, 70
- rq.fit.pfn, 56, 57, 59
- rq.fit.scad, 60
- rq.fit.sfn, 61, 65, 70
- rq.fit.sfnc, 63, 70
- rq.object, 52, 55, 65
- rq.process.object, 52, 55, 66
- rq.test.anovar (*anova.rq*), 3
- rq.test.rank, 4, 48
- rq.test.rank (*anova.rq*), 3
- rq.wfit, 68
- rqProcess, 21, 68
- rqss, 29, 37, 38, 42, 43, 45, 46, 52, 69
- rqss.object, 70, 71
- sfnMessage (*rq.fit.sfn*), 61
- summary, 52
- summary.crq, 8, 17, 73
- summary.crqs (*summary.crq*), 73
- summary.nlrq (*nlrq*), 30
- summary.rcrqs (*summary.rq*), 75
- summary.rq, 10, 21, 44, 52, 56, 57, 68, 69, 75, 78
- summary.rqs, 24
- summary.rqs (*summary.rq*), 75
- summary.rqss, 77
- table.rq, 79
- tau.nlrq (*nlrq*), 30
- triogram.fidelity (*qss*), 44
- triogram.penalty (*qss*), 44
- uis, 79
- untangle.specials (*rqss*), 69
- vignette (*FAQ*), 20