Package: quantreg (via r-universe)

October 8, 2026

Title Quantile Regression

Description Estimation and inference methods for models for conditional quantile functions: Linear and nonlinear parametric and non-parametric (total variation penalized) models for conditional quantiles of a univariate response and several methods for handling censored survival data. Portfolio selection methods based on expected shortfall risk are also now included. See Koenker, R. (2005) Quantile Regression, Cambridge U. Press, <doi:10.1017/CBO9780511754098> and Koenker, R. et al. (2017) Handbook of Quantile Regression, CRC Press, <doi:10.1201/9781315120256>.

Version 6.1

Maintainer Roger Koenker <rkoenker@illinois.edu>

Depends R (>= 3.5), stats, SparseM

Imports methods, graphics, Matrix, MatrixModels, survival, MASS

Suggests interp, rgl, logspline, nor1mix, Formula, zoo, R.rsp, conquer

License GPL (>= 2)

URL https://www.r-project.org

NeedsCompilation yes

VignetteBuilder R.rsp

Author Roger Koenker [cre, aut], Stephen Portnoy [ctb] (Contributions to Censored QR code), Pin Tian Ng [ctb] (Contributions to Sparse QR code), Blaise Melly [ctb] (Contributions to preprocessing code), Achim Zeileis [ctb] (Contributions to dynrq code essentially identical to his dynlm code), Philip Grosjean [ctb] (Contributions to nlrq code), Cleve Moler [ctb] (author of several linpack routines), Yousef Saad [ctb] (author of sparskit2), Victor Chernozhukov [ctb] (contributions to extreme value inference code), Ivan Fernandez-Val [ctb] (contributions to extreme value inference code), Martin Maechler [ctb] (tweaks (src/chlfct.f, 'tiny','Large'), <https://orcid.org/0000-0002-8685-9910>), Brian D Ripley [trl,

ctb] (Initial (2001) R port from S (to my everlasting shame -- how could I have been so slow to adopt R!) and for numerous other suggestions and useful advice)

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

Date/Publication 2025-03-10 09:29:02 UTC

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

RemoteRef HEAD

RemoteSha 43abb16645d5e78d1377896ebd19d760efd2ace9

Contents

akj . . . 4
anova.rq . . . 5
bandwidth.rq . . . 8
barro . . . 9
boot.crq . . . 10
boot.rq . . . 11
boot.rq.pwxy . . . 14
boot.rq.pxy . . . 15
Bosco . . . 16
CobarOre . . . 17
combos . . . 17
critval . . . 18
crq . . . 19
dither . . . 23
dynrq . . . 24
engel . . . 27
FAQ . . . 28
gasprice . . . 29
KhmaladzeTest . . . 29
kuantile . . . 30
LassoLambdaHat . . . 32
latex . . . 33
latex.summary.rqs . . . 33
latex.table . . . 34
lm.fit.recursive . . . 36
lprq . . . 37
Mammals . . . 38
MelTemp . . . 39
Munge . . . 40
nlrq . . . 41
nlrq.control . . . 43
ParetoTest . . . 44
Peirce . . . 45
plot.KhmaladzeTest . . . 47
plot.rq . . . 47

plot.rqs . . . 48
plot.rqss . . . 49
plot.summary.rqs . . . 51
predict.rq . . . 53
predict.rqss . . . 55
print.KhmaladzeTest . . . 57
print.rq . . . 57
print.summary.rq . . . 58
q489 . . . 58
qrisk . . . 59
qss . . . 61
QTECox . . . 63
ranks . . . 65
rearrange . . . 66
residuals.nlrq . . . 67
rq . . . 67
rq.fit . . . 71
rq.fit.br . . . 72
rq.fit.conquer . . . 73
rq.fit.fnb . . . 74
rq.fit.fnc . . . 75
rq.fit.hogg . . . 76
rq.fit.lasso . . . 78
rq.fit.pfn . . . 79
rq.fit.pfnb . . . 80
rq.fit.ppro . . . 81
rq.fit.qfnb . . . 83
rq.fit.scad . . . 84
rq.fit.sfn . . . 85
rq.fit.sfnc . . . 87
rq.object . . . 89
rq.process.object . . . 90
rq.wfit . . . 91
rqProcess . . . 92
rqs.fit . . . 93
rqss . . . 94
rqss.object . . . 96
sfn.control . . . 98
srisk . . . 100
summary.crq . . . 101
summary.rq . . . 103
summary.rqss . . . 106
table.rq . . . 107
uis . . . 108

Index **110**

akj *Density Estimation using Adaptive Kernel method*

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 (dcauchy).

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("nor1mix")) {
 m3 <- norMix(mu= c(-4, 0, 3), sigma = c(1/3, 1, 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, ..., test = "Wald", joint = TRUE, score =
                      "tau", se = "nid", iid = TRUE, R = 200, trim = NULL)
## S3 method for class 'rqs'
anova(object, ..., se = "nid", iid = TRUE, joint = TRUE)
## S3 method for class 'rqlist'
anova(object, ...,  test = "Wald", joint = TRUE,
 score = "tau", se = "nid", iid = TRUE, 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.anowar(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'. or a single object of class rqs, originating from a call to 'rq' with multiple taus specified.
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
se	method for computing standard errors, either "nid" or "ker", note that "boot" cannot be used for testing homogeneity of slopes.
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 anowar 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)
# Alternatively fitting can be done in one call:
fit <- rq(y.net ~  lgdp2 + fse2 + gedy2 + Iy2 + gcony2,
   method = "fn", tau = 1:4/5, data = barro)
```

`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

p	quantile(s) of interest
n	sample size
hs	flag for hall-sheather method
alpha	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 https://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, ctype = "right", R = 100, mboot, bmethod = "jack", ...)
```

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.
`ctype`	Either "right" or "left"
`R`	The number of bootstrap replications
`bmethod`	The bootstrap method to be employed. There are (as yet) three options: method = "jack" uses the delete-d jackknife method described by Portnoy (2013), 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. The "jack" method is now the default.

mboot	optional argument for the bootstrap method: for bmethod = "jack" it specifies the number, d, of the delete-d jackknife, for method = "xy-pair" it specifies the size of the bootstrap samples, that permits subsampling (m out of n) bootstrap. By default in the former case it is set to 2 [sqrt(n)], for the latter the default is n. Obviously mboot should be substantially larger than the column dimension of x, and should be less than the sample size in both cases.
...	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 "Powell" is done via `boot.rq`. For problems with `n > 3000` a message is printed indicated progress in the resampling.

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

References

Bose, A. and S. Chatterjee, (2003) Generalized bootstrap for estimators of minimizers of convex functions, *J. Stat. Planning and Inf*, 117, 225-239. Portnoy, S. (2013) The Jackknife's Edge: Inference for Censored Quantile Regression, *CSDA*, forthcoming.

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),
 coef = NULL, blbn = NULL, cluster = NULL, U = NULL,  ...)
```

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) five 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 "mcmb" method isn't compatible with sparse X matrices. The fourth method = "wxy" uses the generalized bootstrap of Bose and Chatterjee (2003) with unit exponential weights, see also Chamberlain and Imbens (2003). The fifth method "wild" uses the wild bootstrap method proposed by Feng, He and Hu (2011).
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.
coef	coefficients from initial fitted object
blbn	orginal sample size for the BLB model
cluster	If non-NULL this argument should specify cluster id numbers for each observation, in which case the clustered version of the bootstrap based on the proposal of Hagemann (2017). If present bsmethod is set to set to "cluster". If this option is used and the fitting method for the original call was "sfn" then the bootstrapping will be carried out with the "sfn" as well. This is usually substantially quicker than the older version which employed the "br" variant of the simplex method. Use of "sfn" also applies to the "pwy" method when the original fitting was done with "sfn". Finally, if na.action = "omit" and length(object$na.action) > 0 then these elements are also removed from the cluster variable. Consequently, the length of the cluster variable should always be the same as the length of the original response variable before any na.action takes place.
U	If non-NULL this argument should specify an array of indices or gradient evaluations to be used by the corresponding bootstrap method as specified by bsmethod. This is NOT intended as a user specified input, instead it is specified in summary.rqs to ensure that bootstrap samples for multiple taus use the same realizations of the random sampling.
...	Optional arguments to control bootstrapping

Details

Their 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. And more flexibility about what algorithm is used would also be good.

Value

A list consisting of two elements: A matrix B 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). A matrix `U` of sampled indices (for `bsmethod in c("xy", "wxy")`) or gradient evaluations (for `bsmethod in c("pwy", "cluster")`) used to generate the bootstrapped realization, and potentially reused for other `taus` when invoked from `summary.rqs`.

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] Hagemann, A. (2017) Cluster Robust Bootstrap inference in quantile regression models, Journal of the American Statistical Association , 112, 446–456.

[4] He, X. and Hu, F. (2002). Markov Chain Marginal Bootstrap. Journal of the American Statistical Association , Vol. 97, no. 459, 783-795.

[5] Parzen, M. I., L. Wei, and Z. Ying (1994): A resampling method based on pivotal estimating functions," Biometrika, 81, 341–350.

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

[7] Chamberlain G. and Imbens G.W. (2003) Nonparametric Applications of Bayesian Inference, Journal of Business & Economic Statistics, 21, pp. 12-18.

[8] Feng, Xingdong, Xuming He, and Jianhua Hu (2011) Wild Bootstrap for Quantile Regression, Biometrika, 98, 995–999.

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")
```

`boot.rq.pwxy` *Preprocessing weighted bootstrap method*

Description

Bootstrap method exploiting preprocessing strategy to reduce computation time for large problem. In contrast to `boot.rq.pxy` which uses the classical multinomial sampling scheme and is coded in R, this uses the exponentially weighted bootstrap scheme and is coded in fortran and consequently is considerably faster in larger problems.

Usage

```
boot.rq.pwxy(x, y, tau, coef, R = 200, m0 = NULL, eps = 1e-06, ...)
```

Arguments

`x`	Design matrix
`y`	response vector
`tau`	quantile of interest
`coef`	point estimate of fitted object
`R`	the number of bootstrap replications desired.
`m0`	constant to determine initial sample size, defaults to sqrt(n*p) but could use some further tuning...
`eps`	tolerance for convergence of fitting algorithm
`...`	other parameters not yet envisaged.

Details

The fortran implementation is quite similar to the R code for `boot.rq.pxy` except that there is no multinomial sampling. Instead `rexp(n)` weights are used.

Value

returns a list with elements:

`coefficients`	a matrix of dimension ncol(x) by R
`nit`	a 5 by m matrix of iteration counts
`info`	an m-vector of convergence flags

Author(s)

Blaise Melly and Roger Koenker

References

Chernozhukov, V. I. Fernandez-Val and B. Melly, Fast Algorithms for the Quantile Regression Process, 2019, arXiv, 1909.05782,

Portnoy, S. and R. Koenker, The Gaussian Hare and the Laplacian Tortoise, Statistical Science, (1997) 279-300

See Also

boot.rq.pxy

`boot.rq.pxy` *Preprocessing bootstrap method*

Description

Bootstrap method exploiting preprocessing strategy to reduce computation time for large problem.

Usage

```
boot.rq.pxy(x, y, s, tau = 0.5, coef, method = "fn", Mm.factor = 3)
```

Arguments

`x`	Design matrix
`y`	response vector
`s`	matrix of multinomial draws for xy bootstrap
`tau`	quantile of interest
`coef`	point estimate of fitted object
`method`	fitting method for bootstrap
`Mm.factor`	constant to determine initial sample size

Details

See references for further details.

Value

Returns matrix of bootstrap estimates.

Author(s)

Blaise Melly and Roger Koenker

References

Chernozhukov, V. I. Fernandez-Val and B. Melly, Fast Algorithms for the Quantile Regression Process, 2019, arXiv, 1909.05782,

Portnoy, S. and R. Koenker, The Gaussian Hare and the Laplacian Tortoise, Statistical Science, (1997) 279-300

See Also

rq.fit.ppro

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

n	The n in n choose p
p	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

kappa	length of the tube
alpha	desired non-coverage of the band, intended coverage is 1 - alpha
rdf	"residual" degrees of freedom of the fitted object. If rdf=0 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

---

crq *Functions to fit censored quantile regression models*

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", "Portnoy2", "PengHuang"), contrasts = NULL, ...)
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, ctype = "right")
crq.fit.por(x, y, cen, weights=NULL, grid, ctype = "right")
crq.fit.por2(x, y, cen, weights=NULL, grid, ctype = "right")
Curv(y, yc, ctype=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 Surv object as returned by either the Curv or Surv function. For the Powell method, the Surv object should be created by Curv 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 Surv should be created with the usual Surv 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. The "Portnoy" method also enforces an equally spaced grid, see the code for details.
x	An object of class crq or crq.
object	An object of class crq or crq.
yc	The censoring times for the "Powell" method.
ctype	Censoring type: for the "Powell" method, used in Curv, 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 yc must be provided for all observations and the event times y must satisfy the (respective) inequality constraints. For the Portnoy and Peng-Huang methods ctype is determined by the specification of the response as specified in Surv.
type	specifies either "left" or "right" as the form of censoring in the Surv function for the "Portnoy" and "PengHuang" methods.
cen	The censoring indicator for the "Portnoy" and "PengHuang" methods.
maxit	Maximum number of iterations allowed for the "Powell" methods.
start	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 start 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.
left	A logical indicator for left censoring for the "Powell" method.
taus	The quantile(s) at which the model is to be estimated.
tau	The quantile at which the model is to be estimated.

data	A data.frame in which to interpret the variables named in the 'formula', in the 'subset', and the 'weights' argument.
subset	an optional vector specifying a subset of observations to be used in the fitting process.
weights	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.
na.action	a function to filter missing data. This is applied to the model.frame 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.
method	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.
contrasts	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 choses 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 mimumum. 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). There is a somewhat simplified version of the Portnoy method that is written in R and iterates over a discrete grid. This version should be considered somewhat experimental at this stage, but it is known to avoid some difficulties with the more complicated

fortran version of the algorithm that can occur in degenerate problems. 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 `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, https://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"))

# crq example with left censoring
set.seed(1968)
n <- 200
x <-rnorm(n)
y <- 5 + x + rnorm(n)
plot(x,y,cex = .5)
c <- 4 + x + rnorm(n)
d <- (y > c)
points(x[!d],y[!d],cex = .5, col = 2)
f <- crq(survival::Surv(pmax(y,c), d, type = "left") ~ x, method = "Portnoy")
g <- summary(f)
for(i in 1:4) abline(coef(g[[i]])[,1])
```

`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

`x`	`x` a numeric vector
`type`	`type` is either 'symmetric' or 'right'
`value`	`value` 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(dither(y, type = "right", value = 1) ~ x)
```

`dynrq` *Dynamic Linear Quantile Regression*

Description

Interface to `rq.fit` and `rq.wfit` for fitting dynamic linear quantile regression models. The interface is based very closely on Achim Zeileis's dynlm package. In effect, this is mainly "syntactic sugar" for formula processing, but one should never underestimate the value of good, natural sweeteners.

Usage

```
dynrq(formula, tau = 0.5, data, subset, weights, na.action, method = "br",
  contrasts = NULL, start = NULL, end = NULL, ...)
```

Arguments

formula	a "formula" describing the linear model to be fit. For details see below and rq.
tau	the quantile(s) to be estimated, may be vector valued, but all all values must be in (0,1).
data	an optional "data.frame" or time series object (e.g., "ts" or "zoo"), containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which rq is called.
subset	an optional vector specifying a subset of observations to be used in the fitting process.
weights	an optional vector of weights to be used in the fitting process. If specified, weighted least squares is used with weights weights (that is, minimizing sum(w*e^2)); otherwise ordinary least squares is used.
na.action	a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The "factory-fresh" default is na.omit. Another possible value is NULL, no action. Note, that for time series regression special methods like na.contiguous, na.locf and na.approx are available.
method	the method to be used; for fitting, by default method = "br" is used; method = "fn" employs the interior point (Frisch-Newton) algorithm. The latter is advantageous for problems with sample sizes larger than about 5,000.
contrasts	an optional list. See the contrasts.arg of model.matrix.default.
start	start of the time period which should be used for fitting the model.
end	end of the time period which should be used for fitting the model.
...	additional arguments to be passed to the low level regression fitting functions.

Details

The interface and internals of dynrq are very similar to rq, but currently dynrq offers two advantages over the direct use of rq for time series applications of quantile regression: extended formula processing, and preservation of time series attributes. Both features have been shamelessly lifted from Achim Zeileis's package dynlm.

For specifying the formula of the model to be fitted, there are several functions available which allow for convenient specification of dynamics (via d() and L()) or linear/cyclical patterns (via trend(), season(), and harmon()). These new formula functions require that their arguments are time series objects (i.e., "ts" or "zoo").

Dynamic models: An example would be d(y) ~ L(y, 2), where d(x, k) is diff(x, lag = k) and L(x, k) is lag(x, lag = -k), note the difference in sign. The default for k is in both cases 1. For L(), it can also be vector-valued, e.g., y ~ L(y, 1:4).

Trends: y ~ trend(y) specifies a linear time trend where (1:n)/freq is used by default as the covariate, n is the number of observations and freq is the frequency of the series (if any, otherwise freq = 1). Alternatively, trend(y, scale = FALSE) would employ 1:n and time(y) would employ the original time index.

Seasonal/cyclical patterns: Seasonal patterns can be specified via season(x, ref = NULL) and harmonic patterns via harmon(x, order = 1). season(x, ref = NULL) creates a factor with levels for

each cycle of the season. Using the `ref` argument, the reference level can be changed from the default first level to any other. `harmon(x, order = 1)` creates a matrix of regressors corresponding to `cos(2 * o * pi * time(x))` and `sin(2 * o * pi * time(x))` where `o` is chosen from `1:order`.

See below for examples.

Another aim of `dynrq` is to preserve time series properties of the data. Explicit support is currently available for `"ts"` and `"zoo"` series. Internally, the data is kept as a `"zoo"` series and coerced back to `"ts"` if the original dependent variable was of that class (and no internal `NA`s were created by the `na.action`).

See Also

`zoo`, `merge.zoo`

Examples

```
############################
## Dynamic Linear Quantile Regression Models ##
############################

if(require(zoo)){
## multiplicative median SARIMA(1,0,0)(1,0,0)_12 model fitted to UK seatbelt data
     uk <- log10(UKDriverDeaths)
     dfm <- dynrq(uk ~ L(uk, 1) + L(uk, 12))
     dfm

     dfm3 <- dynrq(uk ~ L(uk, 1) + L(uk, 12),tau = 1:3/4)
     summary(dfm3)
 ## explicitly set start and end
     dfm1 <- dynrq(uk ~ L(uk, 1) + L(uk, 12), start = c(1975, 1), end = c(1982, 12))
 ## remove lag 12
     dfm0 <- update(dfm1, . ~ . - L(uk, 12))
     tuk1  <- anova(dfm0, dfm1)
 ## add seasonal term
     dfm1 <- dynrq(uk ~ 1, start = c(1975, 1), end = c(1982, 12))
     dfm2 <- dynrq(uk ~ season(uk), start = c(1975, 1), end = c(1982, 12))
     tuk2 <- anova(dfm1, dfm2)
 ## regression on multiple lags in a single L() call
     dfm3 <- dynrq(uk ~ L(uk, c(1, 11, 12)), start = c(1975, 1), end = c(1982, 12))
     anova(dfm1, dfm3)
}

##############################
## Time Series Decomposition ##
##############################

## airline data
## Not run:
ap <- log(AirPassengers)
fm <- dynrq(ap ~ trend(ap) + season(ap), tau = 1:4/5)
sfm <- summary(fm)
plot(sfm)
```

```
## End(Not run)

## Alternative time trend specifications:
##   time(ap)                 1949 + (0, 1, ..., 143)/12
##   trend(ap)                (1, 2, ..., 144)/12
##   trend(ap, scale = FALSE)  (1, 2, ..., 144)

##############################
## An Edgeworth (1886) Problem##
##############################
# DGP
## Not run:
fye <- function(n, m = 20){
    a <- rep(0,n)
    s <- sample(0:9, m, replace = TRUE)
    a[1] <- sum(s)
    for(i in 2:n){
       s[sample(1:20,1)] <- sample(0:9,1)
       a[i] <- sum(s)
    }
    zoo::zoo(a)
}
x <- fye(1000)
f <- dynrq(x ~ L(x,1))
plot(x,cex = .5, col = "red")
lines(fitted(f), col = "blue")

## End(Not run)
```

`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 - transformed)")
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 and ChangeLog of a package*

Description

Show the FAQ or ChangeLog of a specified package

Usage

```
FAQ(pkg = "quantreg")
ChangeLog(pkg = "quantreg")
```

Arguments

pkg	Package Name

Details

Assumes that the FAQ and/or ChangeLog files exist in the proper "inst" directory.

Value

Has only the side effect of showing the files on the screen.

gasprice *Time Series of US Gasoline Prices*

Description

Time Series of Weekly US Gasoline Prices: 1990:8 – 2003:26

Usage

```
data("gasprice")
```

Examples

```
data(gasprice)
```

KhmaladzeTest *Tests of Location and Location Scale Shift Hypotheses for Linear Models*

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 as adapted to quantile regression by Koenker and Xiao (2002).

Usage

```
KhmaladzeTest(formula, data = NULL, taus = 1:99/100, nullH = "location" ,
 trim = c(0.05, 0.95), h = 1, ...)
```

Arguments

formula	a formula specifying the model to fit by rqProcess
data	a data frame within which to interpret the formula
taus	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.
nullH	a character vector indicating whether the "location" shift hypothesis (default) or the "location-scale" shift hypothesis should be tested.
trim	a vector indicating the lower and upper bound of the quantiles to included in the computation of the test statistics (only, not estimates).
h	an initial bandwidth for the call to akj.
...	other arguments to be passed to summary.rq.

Value

an object of class KhmaladzeTest is returned containing:

nullH	The form of the null hypothesis.
Tn	Joint test statistic of the hypothesis that all the slope parameters of the model satisfy the hypothesis.
THn	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)
T = KhmaladzeTest( y.net ~ lgdp2 + fse2 + gedy2 + Iy2 + gcony2,
  data = barro, taus = seq(.05,.95,by = .01))
plot(T)
```

---

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 $1.5n + O((nlogn)^{1/2})$ 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 ith 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))
```

`LassoLambdaHat` *Lambda selection for QR lasso problems*

Description

Default procedure for selection of lambda in lasso constrained quantile regression as proposed by Belloni and Chernozhukov (2011)

Usage

```
LassoLambdaHat(X, R = 1000, tau = 0.5, C = 1, alpha = 0.95)
```

Arguments

`X`	Design matrix
`R`	Number of replications
`tau`	quantile of interest
`C`	Cosmological constant
`alpha`	Interval threshold

Details

As proposed by Belloni and Chernozhukov, a reasonable default lambda would be the upper quantile of the simulated values. The procedure is based on idea that a simulated gradient can be used as a pivotal statistic. Elements of the default vector are standardized by the respective standard deviations of the covariates. Note that the sqrt(tau(1-tau)) factor cancels in their (2.4) (2.6). In this formulation even the intercept is penalized. If the lower limit of the simulated interval is desired one can specify `alpha = 0.05`.

Value

vector of default lambda values of length p, the column dimension of X.

References

Belloni, A. and V. Chernozhukov. (2011) l1-penalized quantile regression in high-dimensional sparse models. *Annals of Statistics*, 39 82 - 130.

Examples

```
n <- 200
p <- 10
x <- matrix(rnorm(n*p), n, p)
b <- c(1,1, rep(0, p-2))
y <- x %*% b + rnorm(n)
f <- rq(y ~ x, tau = 0.8, method = "lasso")
# See f$lambda to see the default lambda selection
```

---

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 *Writes a latex formatted table to a file*

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, n.cgroup, rgroup, n.rgroup=NULL,
 digits, dec, rdec, cdec, append=FALSE, dcolumn=FALSE, cdot=FALSE,
 longtable=FALSE, table.env=TRUE, lines.page=40, caption, caption.lot,
 label=file, double.slash=FALSE,...)
```

Arguments

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

Usage

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

Arguments

X	Design Matrix
y	Response Variable
int	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

`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 is 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 oonparametric 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), oppossum (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 ot 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)
```

MelTemp *Daily maximum temperatures in Melbourne, Australia*

Description

Daily maximum temperatures in Melbourne, Australia, from 1981-1990. Leap days have been omitted.

Usage

```
data(MelTemp)
```

Format

Time series of frequency 365

Source

Hyndman, R.J., Bashtannyk, D.M. and Grunwald, G.K. (1996) "Estimating and visualizing conditional densities". _Journal of Computational and Graphical Statistics_, *5*, 315-336.

Examples

```
data(MelTemp)
demo(Mel)
```

Munge *Munge rqss formula*

Description

function to recursively substitute arguments into rqss formula

Usage

```
Munge(formula, ...)
```

Arguments

formula	A rqss formula
...	Arguments to be substituted into formula

Details

Intended (originally) for use with demo(MCV). Based on an R-help suggestion of Gabor Grothendieck.

Value

A new formula after substitution

See Also

demo(MCV)

Examples

```
lams <- c(1.3, 3.3)
f <- y ~ qss(x, lambda = lams[1]) + qss(z, lambda = lams[2]) + s
ff <- Munge(f, lams = lams)
```

`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 `optim` 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` *Set control parameters for nlrq*

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=0.97)
```

Arguments

`maxiter`	maximum number of allowed iterations
`k`	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
`InitialStepSize`	Starting value in `optim` to determine the step length of iterations. The default value of 1 is sometimes too optimistic. In such cases, the value 0 forces optim to just barely stick its toe in the water.
`big`	a large scalar
`eps`	tolerance for convergence of the algorithm
`beta`	a shrinkage parameter which controls the recentering process in the interior point algorithm.

See Also

`nlrq`

`ParetoTest` *Estimation and Inference on the Pareto Tail Exponent for Linear Models*

Description

Estimation and inference about the tail behavior of the response in linear models are based on the adaptation of the univariate Hill (1975) and Pickands (1975) estimators for quantile regression by Chernozhukov, Fernandez-Val and Kaji (2018).

Usage

```
ParetoTest(formula, tau = 0.1, data = NULL, flavor = "Hill", m = 2, cicov = .9, ...)
```

Arguments

`formula`	a formula specifying the model to fit by `rq`
`tau`	A threshold on which to base the estimation
`data`	a data frame within which to interpret the formula
`flavor`	Currently limited to either "Hill" or "Pickands"
`m`	a tuning parameter for the Pickands method .
`cicov`	Desired coverage probability of confidence interval.
`...`	other arguments to be passed to `summary.rq`. by default the summary method is the usual xy bootstrap, with `B = 200` replications.

Value

an object of class ParetoTest is returned containing:

`z`	A named vector with components: the estimate, a bias corrected estimate, a lower bound of the confidence interval, an upper bound of the confidence interval, and a Bootstrap Standard Error estimate.
`tau`	The tau threshold used to compute the estimate

References

Chernozhukov, Victor, Ivan Fernandez-Val, and Tetsuya Kaji, (2018) Extremal Quantile Regression, in Handbook of Quantile Regression, Eds. Roger Koenker, Victor Chernozhukov, Xuming He, Limin Peng, CRC Press.

Hill, B. M. (1975). A simple general approach to inference about the tail of a distribution. The Annals of Statistics 3(5), 1163-1174.

Pickands, J. (1975). Statistical inference using extreme order statistics. The Annals of Statistics 3(1), 119-131.

Examples

```
n = 500
x = rnorm(n)
y = x + rt(n,2)
Z = ParetoTest(y ~ x, .9, flavor = "Pickands")
```

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 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) constains 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

Fr\'echet, M. (1924): "Sur la loi des erreurs d'observation," *Matematichiskii 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. Peirces 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)
 }
```

plot.KhmaladzeTest *Plot a KhmaladzeTest object*

Description

Plot an object generated by KhmaladzeTest

Usage

```
## S3 method for class 'KhmaladzeTest'
plot(x, ...)
```

Arguments

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

See Also

KhmaladzeTest

plot.rq *plot the coordinates of the quantile regression process*

Description

Function to plot quantile regression process.

Usage

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

Arguments

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

Author(s)

Roger Koenker 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

x	an object of class "rqs" as produce by rq (with a vector of tau values).
parm	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.
ols	logical. Should a line for the OLS coefficient (as estimated by lm) 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.

Details

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

Value

A matrix with all coefficients visualized is returned invisibly.

See Also

rq, plot.summary.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)

## 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`. Note that these functions are intended to plot `rqss` objects only, attempting to plot `summary.rqss` objects just generates a warning message.

Usage

```
## S3 method for class 'rqss'
plot(x, rug = TRUE, jit = TRUE, bands = NULL, coverage = 0.95,
 add = FALSE, shade = TRUE, select = NULL, pages = 0, titles = NULL,
 bcol = NULL, ...)
## S3 method for class 'qss1'
plot(x, rug = TRUE, jit = TRUE, add = FALSE, ...)
## S3 method for class 'qts1'
plot(x, rug = TRUE, jit = TRUE, add = FALSE, ...)
## S3 method for class 'qss2'
plot(x, render = "contour", ncol = 100, zcol = NULL, ...)
## S3 method for class 'summary.rqss'
plot(x, ...)
```

Arguments

`x`	a fitted `rqss` object produced by `rqss()`.
`...`	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, if "uniform" then uniform bands are plotted, if "both" then both the uniform and the pointwise bands are plotted.

coverage	desired coverage probability of confidence bands, if requested
select	vector of indices of qss objects to be plotted, by default all
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 **rgl**.
add	if TRUE then add qss curve to existing (usually) scatterplot, otherwise initiate a new plot
shade	if TRUE then shade the confidence band
titles	title(s) as vector of character strings, by default titles are chosen for each plot as "Effect of CovariateName"
bcol	vector of two colors for confidence bands
ncol, zcol	Only for render = "rgl": number of colors and z values for color construction.

Details

For univariate qss components with Dorder = 0 the fitted function is piecewise constant, not piecewise linear. In this case the constraints are limited to increasing, decreasing or none. 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.

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 (invisibly):

x	The x coordinates of the confidence bands
blo	The y coordinates of the lower confidence curve, if bands = "both" then this is a matrix with two columns
bhi	The y coordinates of the upper confidence curve, if bands = "both" then this is a matrix with two columns

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.

[6] Koenker, R. (2010) Additive Models for Quantile Regression: Model Selection and Confidence Bandaids, 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
if(requireNamespace("interp")){
if(requireNamespace("interp")){
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

x	an object of class "summary.rqs" as produce by applying the summary method to a rq object (with a vector of tau values).
parm	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.
level	Confidence level of bands. When using the rank based confidence intervals in summary, which is the default method for sample sizes under 1000, you will need to control the level of the intervals by passing the parameter alpha to summary.rq, prior to calling plot.summary.rqs. Note also that alpha = 1 - level.
ols	logical. Should a line for the OLS coefficient and their confidence bands (as estimated by lm) be added?
mfrow, mar, ylim, main	graphical parameters. Suitable defaults are chosen based on the coefficients to be visualized. It can be useful to use a common vertical scale when plotting as a way of comparing confidence bands constructed by different methods. For this purpose one can specify a ylim as a 2 by length(parm) matrix.
col	vector of color specification for rq coefficients and the associated confidence polygon.
border	color specification for the confidence polygon. By default, the second element of col is used.
lcol, lty	color and line type specification for OLS coefficients and their confidence bounds.
cex, pch, type, xlab, ylab, ...	further graphical parameters passed to points.

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

A list with components z, an array with all coefficients visualized (and associated confidence bands), and Ylim, a 2 by p matrix containing the y plotting limits. The latter component may be useful for establishing a common scale for two or more similar plots. The list 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 = "Qhat", stepfun = FALSE, na.action = na.pass, ...)
## S3 method for class 'rq.process'
predict(object, newdata, type = "Qhat", 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	For predict.rq, the 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. For predict.rqs and predict.rq.process when stepfun = TRUE, type is "Qhat", "Fhat" or "fhat" depending on whether the user would like to have estimates of the conditional quantile, distribution or density functions respectively. As noted below the two former estimates can be monotonized with the function rearrange. When the "fhat" option is invoked, a list of conditional density functions is returned based on Silverman's adaptive kernel method as implemented in akj and approxfun.

stepfun	If 'TRUE' return stepfunctions otherwise return matrix of predictions. these functions can be estimates of either the conditional quantile or distribution functions depending upon the type argument. When stepfun = FALSE a matrix of point estimates of the conditional quantile function at the points specified by the newdata argument.
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 'rqs' 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 = "Conditional Ozone Quantile Prediction")
lines(fpr[[2]], col="red")
legend(.2,20,c("raw","cooked"),lty = c(1,1),col=c("black","red"))
fp <- predict(f, newdata=airq, type = "Fhat", stepfun = TRUE)
```

```
fpr <- rearrange(fp)
plot(fp[[2]],main = "Conditional Ozone Distribution Prediction")
lines(fpr[[2]], col="red")
legend(20,.4,c("raw","cooked"),lty = c(1,1),col=c("black","red"))
```

`predict.rqss` *Predict from fitted nonparametric quantile regression smoothing spline models*

Description

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

Usage

```
## S3 method for class 'rqss'
predict(object, newdata, interval = "none", level = 0.95, ...)
## S3 method for class 'qss1'
predict(object, newdata, ...)
## S3 method for class 'qss2'
predict(object, newdata, ...)
```

Arguments

`object`	is a fitted object produced by `rqss`
`newdata`	a data frame describing the observations at which prediction is to be made. For qss components, newdata 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.
`interval`	If set to `confidence` then a `level` confidence interval for the predictions is returned.
`level`	intended coverage probability for the confidence intervals
`...`	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, or in the case that `interval = "confidence"`) a matrix whose first column is the vector of predictions and whose second and third columns are the lower and upper confidence limits for each prediction.

Author(s)

R. Koenker

See Also

rqss

Examples

```
n <- 200
lam <- 2
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)
Ifit <- rqss(y ~ qss(x,constraint="I") + z)
sfit <- rqss(y ~ qss(x,lambda = lam) + z)
xz <- data.frame(z = mean(z),
                 x = seq(min(x)+.01,max(x)-.01,by=.25))
lines(xz[["x"]], predict(Ifit, xz), col=2)
lines(xz[["x"]], predict(sfit, xz), col=3)
legend(10,2,c("Increasing","Smooth"),lty = 1, col = c(2,3))
title("Predicted Median Response at Mean Value of z")

## Bivariate example -- loads pkg "interp"
if(requireNamespace("interp")){
if(requireNamespace("interp")){
data(CobarOre)
fit <- rqss(z ~ qss(cbind(x,y), lambda=.08),
            data= CobarOre)
plot(fit, col="grey",
     main = "CobarOre data -- rqss(z ~ qss(cbind(x,y)))")
T <- with(CobarOre, interp::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 <- interp::in.convex.hull(T, xd, yd))
pred <- predict(fit, data.frame(x = xd[s], y = yd[s]))
contour(interp::interp(xd[s],yd[s], pred),
        col="red", add = TRUE)
}}
```

print.KhmaladzeTest *Print a KhmaladzeTest object*

Description

Print an object generated by KhmaladzeTest

Usage

```
## S3 method for class 'KhmaladzeTest'
print(x, ...)
```

Arguments

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

See Also

KhmaladzeTest

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

q489 *Even Quicker Sample Quantiles*

Description

The function q489 computes a single sample quantile using a fortran implementation of the Floyd and Rivest (1975) algorithm. In contrast to the more elaborate function kuantile that uses the Kiweil (2005) implementation it does not attempt to replicate the nine varieties of quantiles as documented in the base function. quantile

Usage

```
q489(x, tau = .5)
```

Arguments

x	numeric vector
tau	the quantile of intereste.

Details

This is a direct translation of the Algol 68 implementation of Floyd and Rivest (1975), implemented in Ratfor. For the median, average case behavior requires $1.5n + O((nlogn)^{1/2})$ comparisons. In preliminary experiments it seems to be somewhat faster in large samples than the implementation `kuantile` of Kiwiel (2005). See Knuth (1998) for further details. No provision is made for non-uniqueness of the quantile. so, when τn is an integer there may be some discrepancy.

Value

A scalar quantile of the same length as the vector p.

Author(s)

R.W.Floyd and R.L.Rivest, R implementation: Roger Koenker

References

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

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

```
medx <- q489(rnorm(1001))
```

---

qrisk *Function to compute Choquet portfolio weights*

Description

This function solves a weighted quantile regression problem to find the optimal portfolio weights minimizing a Choquet risk criterion described in Bassett, Koenker, and Kordas (2002).

Usage

```
qrisk(x, alpha = c(0.1, 0.3), w = c(0.7, 0.3), mu = 0.07,
      R = NULL, r = NULL, lambda = 10000)
```

Arguments

x	n by q matrix of historical or simulated asset returns
alpha	vector of alphas receiving positive weights in the Choquet criterion
w	weights associated with alpha in the Choquet criterion
mu	targeted rate of return for the portfolio
R	matrix of constraints on the parameters of the quantile regression, see below
r	rhs vector of the constraints described by R
lambda	Lagrange multiplier associated with the constraints

Details

The function calls `rq.fit.hogg` which in turn calls the constrained Frisch Newton algorithm. The constraints Rb=r are intended to apply only to the slope parameters, not the intercept parameters. The user is completely responsible to specify constraints that are consistent, ie that have at least one feasible point. See examples for imposing non-negative portfolio weights.

Value

pihat	the optimal portfolio weights
muhat	the in-sample mean return of the optimal portfolio
qrisk	the in-sample Choquet risk of the optimal portfolio

Author(s)

R. Koenker

References

http://www.econ.uiuc.edu/~roger/research/risk/risk.html

Bassett, G., R. Koenker, G Kordas, (2004) Pessimistic Portfolio Allocation and Choquet Expected Utility, J. of Financial Econometrics, 2, 477-492.

See Also

rq.fit.hogg, srisk

Examples

```
#Fig 1:  ... of Choquet paper
        mu1 <- .05; sig1 <- .02; mu2 <- .09; sig2 <- .07
        x <- -10:40/100
        u <- seq(min(c(x)),max(c(x)),length=100)
        f1 <- dnorm(u,mu1,sig1)
        F1 <- pnorm(u,mu1,sig1)
        f2 <- dchisq(3-sqrt(6)*(u-mu1)/sig1,3)*(sqrt(6)/sig1)
        F2 <- pchisq(3-sqrt(6)*(u-mu1)/sig1,3)
        f3 <- dnorm(u,mu2,sig2)
```

```
        F3 <- pnorm(u,mu2,sig2)
        f4 <- dchisq(3+sqrt(6)*(u-mu2)/sig2,3)*(sqrt(6)/sig2)
        F4 <- pchisq(3+sqrt(6)*(u-mu2)/sig2,3)
        plot(rep(u,4),c(f1,f2,f3,f4),type="n",xlab="return",ylab="density")
        lines(u,f1,lty=1,col="blue")
        lines(u,f2,lty=2,col="red")
        lines(u,f3,lty=3,col="green")
        lines(u,f4,lty=4,col="brown")
        legend(.25,25,paste("Asset ",1:4),lty=1:4,col=c("blue","red","green","brown"))
#Now generate random sample of returns from these four densities.
n <- 1000
if(TRUE){ #generate a new returns sample if TRUE
 x1 <- rnorm(n)
 x1 <- (x1-mean(x1))/sqrt(var(x1))
 x1 <- x1*sig1 + mu1
 x2 <- -rchisq(n,3)
 x2 <- (x2-mean(x2))/sqrt(var(x2))
 x2 <- x2*sig1 +mu1
 x3 <- rnorm(n)
 x3 <- (x3-mean(x3))/sqrt(var(x3))
 x3 <- x3*sig2 +mu2
 x4 <- rchisq(n,3)
 x4 <- (x4-mean(x4))/sqrt(var(x4))
 x4 <- x4*sig2 +mu2
 }
library(quantreg)
x <- cbind(x1,x2,x3,x4)
qfit <- qrisk(x)
sfit <- srisk(x)
# Try new distortion function
qfit1 <- qrisk(x,alpha = c(.05,.1), w = c(.9,.1),mu = 0.09)
# Constrain portfolio weights to be non-negative
qfit2 <- qrisk(x,alpha = c(.05,.1), w = c(.9,.1),mu = 0.09,
        R = rbind(rep(-1,3), diag(3)), r = c(-1, rep(0,3)))
```

---

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,
    Dorder = 1, w = rep(1, length(x)))
```

Arguments

x	The covariate determining the nonparametric component, if x is a matrix with two columns then the qss function will construct a penalized triogram term.
lambda	The smoothing parameter governing the tradeoff between fidelity and the penalty component for this term. Larger lambdas produce smoother fits. In future versions there should be an automatic mechanism for default choice of the lambdas. For now, this is the responsibility of the user.
constraint	Optional specification of qualitative constraints on the fitted univariate qss functions, take the values: "N","I","D","V","C" "VI","VD","CI","CD" for none, increasing, decreasing, convex, concave, convex and increasing, etc. And for bivariate qss components can take the values "N","V","C" for none, convex, and concave. Note that confidence bands for constrained fits of this sort, while available from `plot.rqss` as of yet lack a formal justification.
ndum	number of dummy vertices: this is only relevant for qss2 terms. In addition to vertices at the observed (x,y) points ndum dummy vertices are generated – distributed uniformly over the rectangle given by the Cartesian product of the ranges of x and y – observations that fall in the convex hull of the observations are retained. So the actual number of dummy vertices used is smaller than ndum. The values of these vertices are returned in the list dummies, so that they can be reused.
Dorder	Order of the total variation penalty, the default of 1 implies a penalty on the first derivative of the fitted function, a value of 0 implies total variation of the fitted function itself will be penalized. Note that only monotonicity constraints, "I" and "D" are allowed when `Dorder = 0`, and result in estimates that are equivalent to a form of isotonic regression when lambda is sufficiently near zero. Results in this case from the package **isotone** may differ slightly when plotted due to multiple solutions so it is prudent to evaluate the objective function for both solutions.
dummies	list of dummy vertices as generated, for example by triogram.fidelity when ndum > 0. Should be a list with x and y components. These points should lie inside the convex hull of the real xy points, but no explicit checking of this assertion is currently done.
w	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`. When Dorder = 0, fitting is like the taut string methods of Davies (2014), except for the fact that fidelity is quantilesque rather than quadratic, and that no provision is made for automatic selection of the smoothing parameter.

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

Value

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

Author(s)

Roger Koenker

References

Davies, Laurie (2014) *Data Analysis and Approximate Models*, CRC Press.

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 S and t 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 ΔS is negative and $log (S)$ is also negative this has the same sign as b_j The crq model fits the usual AFT form Surv(log(Time),Status), then

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

This is the matrix form returned.

Value

`taus`	points of evaluation of the QTE.
`QTE`	matrix of QTEs, the ith column contains the QTE for the ith 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, "interquartile" which is appropriate for tests of scale shift, normalscale for Gaussian scale shift, halfnormalscale for Gaussian scale shift only to the right of the median, and lehmann for Lehmann local alternatives. See Koenker (2010) for further details on the last three of these scores.
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.

Koenker, R. Rank Tests for Heterogeneous Treatment Effects with Covariates, preprint.

See Also

rq, rq.test.rank anova

Examples

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

`rearrange` *Rearrangement*

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 $\tilde{Q}(u) = \inf\{u : F(y) \geq u\}.The rearranged function inherits the right or left continuity of original step function.$

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, Econometrica, forthcoming.

Chernozhukov, V., I. Fernandez-Val, and A. Galichon, (2009) Improving Estimates of Monotone Functions by Rearrangement, Biometrika, 96, 559–575.

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)),stepfun = TRUE)
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 * pmax(resid, 0) + (1 - tau) * pmin(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 *Quantile Regression*

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.
weights	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.
na.action	a function to filter missing data. This is applied to the model.frame 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.
model	if TRUE then the model frame is returned. This is essential if one wants to call summary subsequently.
method	the algorithmic method used to compute the fit. There are several options:

1. "br" The default method is the modified version of the Barrodale and Roberts algorithm for l_1-regression, used by l1fit 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).
2. "fn" For larger problems it is advantageous to use the Frisch–Newton interior point method "fn". This is described in detail in Portnoy and Koenker(1997).
3. "pfn" For even larger problems one can use the Frisch–Newton approach after preprocessing "pfn". Also described in detail in Portnoy and Koenker(1997), this method is primarily well-suited for large n, small p problems, that is when the parametric dimension of the model is modest.
4. "sfn" For large problems with large parametric dimension it is often advantageous to use method "sfn" which also uses the Frisch-Newton algorithm, but exploits sparse algebra to compute iterates. This is especially helpful when the model includes factor variables that, when expanded, generate

design matrices that are very sparse. At present options for inference, i.e. summary methods are somewhat limited when using the "sfn" method. Only the option se = "nid" is currently available, but I hope to implement some bootstrap options in the near future.

5. "fnc" Another option 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 below.
6. "conquer" For very large problems especially those with large parametric dimension, this option provides a link to the **conquer** of He, Pan, Tan, and Zhou (2020). Calls to summary when the fitted object is computed with this option invoke the multiplier bootstrap percentile method of the **conquer** package and can be considerably quicker than other options when the problem size is large. Further options for this fitting method are described in the **conquer** package. Note that this option employs a smoothing form of the usual QR objective function so solutions may be expected to differ somewhat from those produced with the other options.
7. "pfnb" This option is intended for applications with large sample sizes and/or moderately fine tau grids. It uses a form of preprocessing to accelerate the solution process. The loop over taus occurs inside the Fortran call and there should be more efficient than other methods in large problems.
8. "qfnb" This option is like the preceeding one except that it doesn't use the preprocessing option.
9. "ppro" This option is an R prototype of the pfnb and is offered for historical/interpretative purposes, but probably should be considered deprecated.
10. "lasso" There are two penalized methods: "lasso" and "scad" that implement the lasso penalty and Fan and Li smoothly clipped absolute deviation penalty, respectively. These methods should probably be regarded as experimental. Note: weights are ignored when the method is penalized.

contrasts — 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 routines (see rq.fit.br and rq.fit.fnb, etc. 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 nlrq 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.

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 (in parameters) 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 several 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] Xuming He and Xiaoou Pan and Kean Ming Tan and Wen-Xin Zhou, (2020) conquer: Convolution-Type Smoothed Quantile Regression, https://CRAN.R-project.org/package=conquer

[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/~roger/.

See Also

FAQ, summary.rq, nlrq, rq.fit, rq.wfit, 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.fnb`

`rq.fit.br` *Quantile Regression Fitting by Exterior Point Methods*

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

`x`	the design matrix
`y`	the response variable
`tau`	the quantile desired, if tau lies outside (0,1) the whole process is estimated.
`alpha`	the nominal noncoverage probability for the confidence intervals, i.e. 1-alpha is the nominal coverage probability of the intervals.
`ci`	logical flag if T then compute confidence intervals for the parameters using the rank inversion method of Koenker (1994). See `rq()` for more details. If F then return only the estimated coefficients. Note that for large problems the default option ci = TRUE can be rather slow. Note also that rank inversion only works for p>1, an error message is printed in the case that ci=T and p=1.
`iid`	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 nid error assumption. See Koenker and Machado (1999) for further details on this distinction.
`interp`	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 `interp = F` then the 2 "exact" values above and below on which the interpolation would be based are returned.
`tcrit`	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.conquer *Optional Fitting Method for Quantile Regression*

Description

This fitting method provides a link to the gradient descent for convolution smoothed quantile regression problem implemented in the **conquer** package of He et al (2020).

Usage

```
rq.fit.conquer (x, y, tau=0.5, kernel = c("Gaussian", "uniform",
    "parabolic", "triangular"), h = 0, tol = 1e-04,
    iteMax = 5000, ci = FALSE, alpha = 0.05, B = 200)
```

Arguments

x	design matrix usually supplied via rq(), expected to have a intercept as the first column
y	response vector usually supplied via rq()
tau	quantile of interest
kernel	A character string specifying the choice of kernel function. Default is "Gaussian". Other choices are "uniform", "parabolic" or "triangular".
h	The bandwidth parameter for kernel smoothing of the QR objective function. Default is $max((log(n)+p)/n)^0.4, 0.05$. The default is used if the input value is less than 0.05.

tol	Tolerance level of the gradient descent algorithm. The gradient descent algorithm terminates when the maximal entry of the gradient is less than "tol". Default is 1e-05.
iteMax	Maximum number of iterations. Default is 5000.
ci	A logical flag. Default is FALSE. If "ci = TRUE", then three types of confidence intervals (percentile, pivotal and normal) will be constructed via multiplier bootstrap. This option is subsumed in normal use by the summary.rq functionality.
alpha	Nominal level for confidence intervals, may be passed via the call to summary
B	Number of bootstrap replications. May be passed via summary.

Details

See documentation in the **conquer** package.

Value

Returns an object of class "rq".

References

Xuming He and Xiaoou Pan and Kean Ming Tan and Wen-Xin Zhou, (2020) conquer: Convolution-Type Smoothed Quantile Regression, https://CRAN.R-project.org/package=conquer

See Also

rq

---

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, rhs = (1-tau)*apply(x,2,sum), 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)
rhs	The right hand size of the dual equality constraint, modify at your own risk.
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 >= r.

Usage

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

Arguments

x	The design matrix
y	The response vector
R	The matrix describing the inequality constraints
r	The right hand side vector of inequality constraints
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 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, *Sankya*, 418-440.

See Also

rq, rq.fit.br, rq.fit.pfn

---

rq.fit.hogg *weighted quantile regression fitting*

Description

Function to estimate a regression mmodel by minimizing the weighted sum of several quantile regression functions. See Koenker(1984) for an asymptotic look at these estimators. This is a slightly generalized version of what Zou and Yuan (2008) call composite quantile regression in that it permits weighting of the components of the objective function and also allows further linear inequality constraints on the coefficients.

Usage

```
rq.fit.hogg(x, y, taus = c(0.1, 0.3, 0.5), weights = c(0.7, 0.2, 0.1),
     R = NULL, r = NULL, beta = 0.99995, eps = 1e-06)
```

Arguments

x	design matrix
y	response vector
taus	quantiles getting positive weight
weights	weights assigned to the quantiles
R	optional matrix describing linear inequality constraints
r	optional vector describing linear inequality constraints
beta	step length parameter of the Frisch Newton Algorithm
eps	tolerance parameter for the Frisch Newton Algorithm

Details

Mimimizes a weighted sum of quantile regression objective functions using the specified taus. The model permits distinct intercept parameters at each of the specified taus, but the slope parameters are constrained to be the same for all taus. This estimator was originally suggested to the author by Bob Hogg in one of his famous blue notes of 1979. The algorithm used to solve the resulting linear programming problems is either the Frisch Newton algorithm described in Portnoy and Koenker (1997), or the closely related algorithm described in Koenker and Ng(2002) that handles linear inequality constraints. See qrisk for illustration of its use in portfolio allocation.

Linear inequality constraints of the form $Rb \geq r$ can be imposed with the convention that b is a $m + p$ where m is the length(taus) and p is the column dimension of x without the intercept.

Value

coefficients	estimated coefficients of the model

Author(s)

Roger Koenker

References

Zou, Hui and and Ming Yuan (2008) Composite quantile regression and the Oracle model selection theory, Annals of Statistics, 36, 1108–11120.

Koenker, R. (1984) A note on L-estimates for linear models, Stat. and Prob Letters, 2, 323-5.

Portnoy, S. and Koenker, R. (1997) The Gaussian Hare and the Laplacean Tortoise: Computability of Squared-error vs Absolute Error Estimators, (with discussion). Statistical Science, (1997) 12, 279-300.

Koenker, R. and Ng, P (2003) Inequality Constrained Quantile Regression, preprint.

See Also

qrisk

---

rq.fit.lasso *Lasso Penalized Quantile Regression*

Description

The fitting method implements the lasso penalty for fitting quantile regression models. When the argument lambda is a scalar the penalty function is the l1 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. If lambda is not specified, a default value is selected according to the proposal of Belloni and Chernozhukov (2011). See LassoLambdaHat for further details. There should be a sparse version of this, but isn't (yet). There should also be a preprocessing version, but isn't (yet).

Usage

```
rq.fit.lasso(x, y, tau = 0.5, lambda = NULL, beta = .99995, 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. If unspecified, a default value is chosen according to the proposal of Belloni and Chernozhukov (2011).
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.

Belloni, A. and V. Chernozhukov. (2011) l1-penalized quantile regression in high-dimensional sparse models. *Annals of Statistics*, 39 82 - 130.

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.fixups=3, eps=1e-06)
```

Arguments

x	design matrix usually supplied via rq()
y	response vector usually supplied via rq()
tau	quantile of interest
Mm.factor	constant to determine sub sample size m
max.bad.fixups	number of allowed mispredicted signs of residuals
eps	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.pfnb` *Quantile Regression Fitting via Interior Point Methods*

Description

This is a lower level routine called by `rq()` to compute quantile regression parameters using the Frisch-Newton algorithm. It uses a form of preprocessing to accelerate the computations for situations in which several taus are required for the same model specification.

Usage

```
rq.fit.pfnb(x, y, tau, m0 = NULL, eps = 1e-06)
```

Arguments

`x`	The design matrix
`y`	The response vector
`tau`	The quantiles of interest, must lie in (0,1), be sorted and preferably equally spaced.
`m0`	An initial reduced sample size by default is set to be `round((n * (log(p) + 1) )^(2/3)` this could be explored further to aid performance in extreme cases.
`eps`	A tolerance parameter intended to bound the confidence band entries away from zero.

Details

The details of the Frisch-Newton algorithm are explained in Koenker and Portnoy (1997), as is the preprocessing idea which is related to partial sorting and the algorithms such as `kuantile` for univariate quantiles that operate in time O(n). The preprocessing idea of exploiting nearby quantile solutions to accelerate estimation of adjacent quantiles is proposed in Chernozhukov et al (2020). This version calls a fortran version of the preprocessing algorithm that accepts multiple taus. The preprocessing approach is also implemented for a single tau in `rq.fit.pfn` which may be regarded as a prototype for this function since it is written entirely in R and therefore is easier to experiment with.

Value

returns a list with elements consisting of

`coefficients`	a matrix of dimension ncol(x) by length(taus)
`nit`	a 5 by m matrix of iteration counts: first two coordinates of each column are the number of interior point iterations, the third is the number of observations in the final globbed sample size, and the last two are the number of fixups and bad-fixups respectively. This is intended to aid fine tuning of the initial sample size, m0.
`info`	an m-vector of convergence flags

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.

Chernozhukov, V., I., Fernandez-Val, and Melly, B. (2020), 'Fast algorithms for the quantile regression process', Empirical Economics, forthcoming.

See Also

`rq`, `rq.fit.br`, `rq.fit.pfn`

---

rq.fit.ppro *Preprocessing fitting method for QR*

Description

Preprocessing method for fitting quantile regression models that exploits the fact that adjacent tau's should have nearly the same sign vectors for residuals.

Usage

```
rq.fit.ppro(x, y, tau, weights = NULL, Mm.factor = 0.8, eps = 1e-06, ...)
```

Arguments

x	Design matrix
y	Response vector
tau	quantile vector of interest
weights	case weights
Mm.factor	constant determining initial sample size
eps	Convergence tolerance
...	Other arguments

Details

See references for further details.

Value

Returns a list with components:

coefficients	Matrix of coefficient estimates
residuals	Matrix of residual estimates
rho	vector of objective function values
weights	vector of case weights

Author(s)

Blaise Melly and Roger Koenker

References

Chernozhukov, V. I. Fernandez-Val and B. Melly, Fast Algorithms for the Quantile Regression Process, 2020, Empirical Economics.,

Portnoy, S. and R. Koenker, The Gaussian Hare and the Laplacian Tortoise, Statistical Science, (1997) 279-300

See Also

rq.fit.pfn, boot.rq.pxy

`rq.fit.qfnb` *Quantile Regression Fitting via Interior Point Methods*

Description

This is a lower level routine called by `rq()` to compute quantile regression parameters using the Frisch-Newton algorithm. In contrast to method "fn" it computes solutions for all the specified taus inside a fortran loop. See `rq.fit.pfnb` for further details on a more efficient preprocessing method.

Usage

```
rq.fit.qfnb(x, y, tau)
```

Arguments

`x`	The design matrix
`y`	The response vector
`tau`	The quantiles of interest, must lie in (0,1), be sorted and preferably equally spaced.

Details

The details of the Frisch-Newton 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 linear programming. 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. The current 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 a list with elements consisting of

`coefficients`	a matrix of dimension ncol(x) by length(taus)
`nit`	a 3-vector of iteration counts
`info`	a convergence flag

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.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 l1 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

`x`	the design matrix
`y`	the response variable
`tau`	the quantile desired, defaults to 0.5.
`alpha`	tuning parameter of the scad penalty.
`lambda`	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 `x` matrix.
`start`	starting method, default method 'rq' uses the unconstrained rq estimate, while method 'lasso' uses the corresponding lasso estimate with the specified lambda.
`beta`	step length parameter for Frisch-Newton method.
`eps`	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))), control)
```

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, but in special cases can be quite usefully altered to meet special needs. See e.g. Section 6.8 of Koenker (2005).
control	control parameters for fitting routines: see sfn.control

Details

This is a sparse implementation of the Frisch-Newton algorithm for quantile regression described in Portnoy and Koenker (1997). 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 `srqfnc`:

- **1:** insufficient work space in call to `extract`
- **2:** nnzd > nnzdmax
- **3:** insufficient storage in iwork when calling ordmmd
- **4:** insufficient storage in iwork when calling sfinit
- **5:** nnzl > nnzlmax when calling sfinit
- **6:** nsub > nsubmax when calling sfinit
- **7:** insufficient work space in iwork when calling symfct
- **8:** inconsistancy in input when calling symfct
- **9:** tmpsiz > tmpmax when calling bfinit; increase tmpmax
- **10:** nonpositive diagonal encountered, not positive definite
- **11:** insufficient work storage in tmpvec when calling blkfct
- **12:** insufficient work storage in iwork when calling blkfct
- **17:** tiny diagonals replaced with Inf when calling blkfct

`it` Iteration count

`time` Amount of time used in the computation

Author(s)

Pin Ng

References

Portnoy, S. and R. Koenker (1997) The Gaussian Hare and the Laplacean Tortoise: Computability of Squared-error vs Absolute Error Estimators, (with discussion). *Statistical Science*, 12, 279-300.

Koenker, R and Ng, P. (2003). SparseM: A Sparse Matrix Package for R, *J. of Stat. Software*, 8, 1–9.

Koenker, R. (2005) *Quantile Regression*, Cambridge U. Press.

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, control = list(tmpmax= tmpmax))
```

`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))),control)
```

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.
`control`	control paramters for fitting see `sfn.control`

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 `srqfn`:

- **1:** insufficient work space in call to `extract`
- **3:** insufficient storage in iwork when calling ordmmd
- **4:** insufficient storage in iwork when calling sfinit
- **5:** nnzl > nnzlmax when calling sfinit
- **6:** nsub > nsubmax when calling sfinit
- **7:** insufficient work space in iwork when calling symfct
- **8:** inconsistancy in input when calling symfct
- **9:** tmpsiz > tmpmax when calling symfct; increase tmpmax
- **10:** nonpositive diagonal encountered when calling blkfct
- **11:** insufficient work storage in tmpvec when calling blkfct
- **12:** insufficient work storage in iwork when calling blkfct
- **13:** nnzd > nnzdmax in e,je when calling amub
- **14:** nnzd > nnzdmax in g,jg when calling amub
- **15:** nnzd > nnzdmax in h,jh when calling aplb
- **15:** tiny diagonals replaced with Inf when calling blkfct

`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; https://CRAN.R-project.org/package=SparseM

Koenker, R. and P. Ng(2005). Inequality Constrained Quantile Regression, *Sankya*, 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, control = list(tmpmax = tmpmax))
```

`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 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 'breakpoints' $tau_1, tau_2, \ldots, tau_J$, 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 xbar 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

x	the design matrix
y	the response variable
tau	the quantile desired, if tau lies outside (0,1) the whole process is estimated.
weights	weights used in the fitting
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.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 KhmaladzeTest.

Usage

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

Arguments

formula	model formula
data	data frame to be used to interpret formula
taus	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.
nullH	Null hypothesis to be used for standardization
...	optional arguments passed to summary.rq

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

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

Author(s)

R. Koenker

See Also

KhmaladzeTest

---

rqs.fit *Function to fit multiple response quantile regression models*

Description

Function intended for multiple response quantile regression called from boot.rq for wild bootstrap option.

Usage

```
rqs.fit(x, y, tau=0.5, tol = 0.0001)
```

Arguments

x	the design matrix an n by p matrix.
y	the response variable as a n by m matrix
tau	the quantile desired, if tau lies outside (0,1)
tol	tolerance parameter for Barrodale and Roberts exterior point method.

See Also

boot.rq

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. See `summary.rqss` and `plot.rqss` for further details on inference and confidence bands.

Usage

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

Arguments

`formula`	a formula object, with the response on the left of a '~' operator, and terms, separated by '+' operators, on the right. The terms may include `qss` terms that represent additive nonparametric components. These terms can be univariate or bivariate. See `qss` for details on how to specify these terms.
`tau`	the quantile to be estimated, this must be a number between 0 and 1,
`data`	a data.frame in which to interpret the variables named in the formula, or in the subset and the weights argument.
`weights`	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.
`subset`	an optional vector specifying a subset of observations to be used in the fitting. This can be a vector of indices of observations to be included, or a logical vector.
`na.action`	a function to filter missing data. This is applied to the model.frame 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.
`method`	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 `"sfnc"` is used if the user specifies inequality constraints. Option `"sfn"` 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 `qss` terms, representing the constraints in the form $Rb \geq r$. The option `method = "lasso"` allows one to penalize the coefficients of the covariates that have been entered linearly as in `rq.fit.lasso`; when this is specified then there should be an additional `lambda` argument specified that determines the amount of shrinkage.

lambda	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.
contrasts	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.
ztol	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.
control	control argument for the fitting routines (see sfn.control
...	Other arguments passed to fitting routines

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 vignette("rq"). Since this function uses sparse versions of the interior point algorithm it may also prove to be useful for fitting linear models without qss terms when the design has a sparse structure, as for example when there is a complicated factor structure.

If the **MatrixModels** and **Matrix** packages are both loadable then the linear-in-parameters portion of the design matrix is made in sparse matrix form; this is helpful in large applications with many factor variables for which dense formation of the design matrix would take too much space.

Although modeling with rqss typically imposes smoothing penalties on the total variation of the first derivative, or gradient, of the fitted functions, for univariate smoothing, it is also possible to penalize total variation of the function itself using the option Dorder = 0 inside qss terms. In such cases, estimated functions are piecewise constant rather than piecewise linear. See the documentation for qss for further details.

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.

Note

If you intend to embed calls to rqss inside another function, then it is advisable to pass a data frame explicitly as the data argument of the rqss call, rather than relying on the magic of R scoping rules.

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
if(requireNamespace("interp")){
if(requireNamespace("interp")){
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, ...)
## S3 method for class 'rqss'
resid(object, ...)
## S3 method for class 'rqss'
fitted(object, ...)
```

Arguments

object	an object returned from rqss fitting, describing an additive model estimating a conditional quantile function. See qss for details on how to specify these terms.
x	an rqss 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:

fidelity	Value of the quantile regression objective function.
penalty	A list consisting of the values of the total variation smoothing penalty for each of additive components.
edf	Effective degrees of freedom of the fitted model, defined as the number of zero residuals of the fitted model, Koenker Mizera (2003) for details.
qssedfs	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.
lamdas	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

```
require(MatrixModels)
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
if(requireNamespace("interp")){
if(requireNamespace("interp")){
data(CobarOre)
fCO <- rqss(z ~ qss(cbind(x,y), lambda= .08), data=CobarOre)
plot(fCO)
}}
```

`sfn.control` *Set Control Parameters for Sparse Fitting*

Description

Auxiliary function for setting storage dimensions and other parameters for `rq.fit.sfn()`, `rq.fit.sfnc()` and `rqss()`.

Usage

```
sfn.control(nsubmax = NULL, tmpmax = NULL, nnzlmax = NULL, cachsz = 64,
            small = 1e-06, maxiter = 100,
            tiny = 1e-30, Large = 1e128,
            warn.mesg = TRUE)
```

Arguments

nsubmax	upper bound for dimension of lindx
tmpmax	upper bound for dimension of tmpvec
nnzlmax	upper bound for non-zero entries of L stored in lnz, including diagonal
cachsz	size of cache in kbytes on target machine
small	convergence tolerance for interior point algorithm
maxiter	maximal number of interior point iterations
tiny	a tiny positive number; values below `tiny * max(diag)` are replaced by `Large`; originally was 10^{-30} hardcoded in Fortran code.
Large	a large number, practically "Infinite" to replace `tiny` diagonal entries in Cholesky; was 10^{128}, hardcoded in compiled code.
warn.mesg	logical flag controlling printing of warnings.

Details

Sparse fitting requires a number of temporary storage arrays whose size depends on problem specific features in somewhat mysterious ways, parameters controlling these sizes and some other fitting aspects can be controlled by specifying elements of this control object.

Value

A `list` with components named as the arguments given above.

Author(s)

Roger Koenker

See Also

`rq.fit.sfn`, `rq.fit.sfnc`, and `rqss` from which `sfn.control()` is called.

srisk *Markowitz (Mean-Variance) Portfolio Optimization*

Description

This function estimates optimal mean-variance portfolio weights from a matrix of historical or simulated asset returns.

Usage

```
srisk(x, mu = 0.07, lambda = 1e+08, alpha = 0.1, eps = 1e-04)
```

Arguments

x	Matrix of asset returns
mu	Required mean rate of return for the portfolio
lambda	Lagrange multiplier associated with mean return constraint
alpha	Choquet risk parameter, unimplemented
eps	tolerance parameter for mean return constraint

Details

The portfolio weights are estimated by solving a constrained least squares problem.

Value

pihat	Optimal portfolio weights
muhat	Mean return in sample
sighat	Standard deviation of returns in sample

Author(s)

R. Koenker

See Also

qrisk

summary.crq *Summary methods for Censored Quantile Regression*

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.crqs'
print(x,  ...)
## S3 method for class 'summary.crqs'
plot(x, nrow = 3, ncol = 3, CoxPHit = NULL,  ...)
```

Arguments

object	An object of class "crq" produced by a call to crq().
taus	Quantiles to be summarized. This should be a vector of length greater than one.
x	An object of class "crq" produced by a call to crq().
se	specifies the method used to compute standard standard errors. but the only available method (so far) is "boot". Further arguments to boot.crq and boot.rq can be passed with the ... 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.
alpha	Confidence level for summary intervals.
digits	Number of digits to be printed in summary display.
CoxPHit	An object of class coxph produced by coxph.
...	Optional arguments to summary, e.g. to specify bootstrap methods sample sizes, etc. see boot.rq and boot.crq

Details

For the Powell method the resampling strategy used by the se = "boot" method is based on the Bilias, Chen and Ying (2000) proposal. For the Portnoy and Peng-Huang methods the bootstrapping is by default actually based on a delete-d jackknife, as described in Portnoy (2013), but resampling xy pairs using either conventional multinomial resampling or using exponential weighting as in Bose and Chatterjee (2003) can be used by specifying the bmethod argument. 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. It is important to recognize that when some of the bootstrap replications are NA they are simply ignored in the computation of the confidence bands and standard errors as currently reported. The number of these NAs is returned as part of the summary.crq object, and when printed is also reported.

Value

For method "Powell" an object of class summary.crq 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 covariance = TRUE appears in the calling sequence.
rdf	the residual degrees of freedom
tau	the quantile estimated

For the other methods 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.
cov	the estimated covariance matrix for the coefficients in the model, provided that covariance = TRUE in the called sequence.

References

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

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

Portnoy, S. (2013) The Jackknife's Edge: Inference for Censored Quantile Regression, *CSDA*, forthcoming.

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 = NULL, covariance=FALSE, hs = TRUE, U = NULL, gamma = 0.7, ...)
## S3 method for class 'rqs'
summary(object, ...)
```

Arguments

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

1. "rank" which produces confidence intervals for the estimated parameters by inverting a rank test as described in Koenker (1994). This method involves solving a parametric linear programming problem, and for large sample sizes can be extremely slow, so by default it is only used when the sample size is less than 1000, see below. The default option assumes that the errors are iid, while the option iid = FALSE implements a proposal of Koenker Machado (1999). See the documentation for rq.fit.br for additional arguments.
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. If the initial fitting was done with method "sfn" then use of se = "nid" is recommended. However, if the cluster option is also desired then se = "boot" can be used and bootstrapping will also employ the "sfn" method.
4. "ker" which uses a kernel estimate of the sandwich as proposed by Powell(1991).
5. "boot" which implements one of several possible bootstrapping alternatives for estimating standard errors including a variate of the wild bootstrap for clustered response. See boot.rq for further details.
6. "BLB" which implements the bag of little bootstraps method proposed in Kleiner, et al (2014). The sample size of the little bootstraps is controlled by the parameter gamma, see below. At present only bsmethod = "xy" is

sanction, and even that is experimental. This option is intended for applications with very large n where other flavors of the bootstrap can be slow.

7. `"conquer"` which is invoked automatically if the fitted object was created with `method = "conquer"`, and returns the multiplier bootstrap percentile confidence intervals described in He et al (2020).
8. `"extreme"` which uses the subsampling method of Chernozhukov Fernandez-Val, and Kaji (2018) designed for inference on extreme quantiles.

If `se = NULL` (the default) and `covariance = FALSE`, and the sample size is less than 1001, then the "rank" method is used, otherwise the "nid" method is used.

`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.

`U` Resampling indices or gradient evaluations used for bootstrap, see `boot.rq`.

`gamma` parameter controlling the effective sample size of the'bag of little bootstrap samples that will be `b = n^gamma` where `n` is the sample size of the original model.

`...` Optional arguments to summary, e.g. bsmethod to use bootstrapping. see `boot.rq`. When using the "rank" method for confidence intervals, which is the default method for sample sizes less than 1000, the type I error probability of the intervals can be controlled with the alpha parameter passed via "...", thereby controlling the width of the intervals plotted by `plot.summary.rqs`. Similarly, the arguments alpha, mofn and kex can be passed when invoking the `"extreme"` option for "se" to control the percentile interval reported, given by estimated quantiles [alpha/2, 1 - alpha/2]; `kex` is a tuning parameter for the extreme value confidence interval construction. The size of the bootstrap subsamples for the "extreme" option can also be controlled by passing the argument `mofm` via "...". Default values for kex, mofn and alpha are 20, `floor(n/5)` and 0.1, respectively.

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. There are several options for alternative resampling methods. When `summary.rqs` is invoked, that is when `summary` is called for a `rqs` object consisting of several `taus`, the `B` components of the returned object can be used to construct a joint covariance matrix for the full object.

Value

a list is returned with the following components, when `object` is of class `"rqs"` 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, in the case of most "se" methods. For methods "rank" and "extreme" potentially asymetric confidence intervals are return in lieu of standard errors and p-values.
`cov`	the estimated covariance matrix for the coefficients in the model, provided that `cov=TRUE` in the called sequence. This option is not available when se = "rank".
`Hinv`	inverse of the estimated Hessian matrix returned if `cov=TRUE` and `se %in% c("nid","ker")` , note that for `se = "boot"` there is no way to split the estimated covariance matrix into its sandwich constituent parts.
`J`	Unscaled Outer product of gradient matrix returned if `cov=TRUE` and `se != "iid"`. The Huber sandwich is `cov = tau (1-tau) Hinv %*% J %*% Hinv`. as for the `Hinv` component, there is no `J` component when `se == "boot"`. (Note that to make the Huber sandwich you need to add the tau (1-tau) mayonnaise yourself.)
`B`	Matrix of bootstrap realizations.
`U`	Matrix of bootstrap randomization draws.

References

Chernozhukov, Victor, Ivan Fernandez-Val, and Tetsuya Kaji, (2018) Extremal Quantile Regression, in Handbook of Quantile Regression, Eds. Roger Koenker, Victor Chernozhukov, Xuming He, Limin Peng, CRC Press.

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

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

Kleiner, A., Talwalkar, A., Sarkar, P. and Jordan, M.I. (2014) A Scalable bootstrap for massive data, *JRSS(B)*, 76, 795-816.

Powell, J. (1991) Estimation of Monotonic Regression Models under Quantile Restrictions, in Nonparametric and Semiparametric Methods in Econometrics, W. Barnett, J. Powell, and G Tauchen (eds.), Cambridge U. Press.

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 rqss fitting, describing an additive model estimating a conditional quantile function. See qss for details on how to specify these terms.
cov	if TRUE return covariance matrix for the parametric components as Vcov and a list of covariance matrices for the nonparametric components as Vqss
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 function is 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

coef	Table of estimated coefficients and their standard errors, t-statistics, and p-values for the parametric components of the model
qsstab	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.
fidelity	Value of the quantile regression objective function.
tau	Quantile of the estimated model

formula	formula of the estimated model
edf	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.
n	The sample size used to fit the model.
Vcov	Estimated covariance matrix of the fitted parametric component
Vqss	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

x	input
...	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	
LNDT		
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

∗ **IO**
latex, 33
latex.summary.rqs, 33
∗ **~manip**
Munge, 40
∗ **bootstrap**
boot.rq.pwxy, 14
boot.rq.pxy, 15
∗ **datasets**
barro, 9
Bosco, 16
CobarOre, 17
engel, 27
gasprice, 29
Mammals, 38
MelTemp, 39
Peirce, 45
uis, 108
∗ **documentation**
FAQ, 28
∗ **environment**
nlrq.control, 43
∗ **hplot**
plot.rq, 47
plot.rqs, 48
plot.summary.rqs, 51
∗ **htest**
anova.rq, 5
KhmaladzeTest, 29
ParetoTest, 44
∗ **iplot**
plot.rqss, 49
∗ **manip**
dither, 23
∗ **methods**
lm.fit.recursive, 36
∗ **models**
nlrq, 41
residuals.nlrq, 67
∗ **nonlinear**
nlrq, 41
residuals.nlrq, 67
∗ **regression**
anova.rq, 5
bandwidth.rq, 8
boot.crq, 10
boot.rq, 11
critval, 18
crq, 19
dynrq, 24
nlrq, 41
plot.KhmaladzeTest, 47
plot.rqss, 49
predict.rq, 53
predict.rqss, 55
print.KhmaladzeTest, 57
print.rq, 57
print.summary.rq, 58
qrisk, 59
ranks, 65
rearrange, 66
residuals.nlrq, 67
rq, 67
rq.fit, 71
rq.fit.br, 72
rq.fit.conquer, 73
rq.fit.fnb, 74
rq.fit.fnc, 75
rq.fit.hogg, 76
rq.fit.lasso, 78
rq.fit.pfn, 79
rq.fit.pfnb, 80
rq.fit.ppro, 81
rq.fit.qfnb, 83
rq.fit.scad, 84
rq.fit.sfn, 85
rq.fit.sfnc, 87
rq.object, 89

rq.process.object, 90
rq.wfit, 91
rqProcess, 92
rqs.fit, 93
rqss, 94
rqss.object, 96
srisk, 100
summary.crq, 101
summary.rq, 103
summary.rqss, 106
table.rq, 107
∗ **robust**
anova.rq, 5
lprq, 37
nlrq, 41
predict.rqss, 55
qrisk, 59
qss, 61
rq.fit.hogg, 76
rqss, 94
rqss.object, 96
summary.rqss, 106
∗ **smooth**
akj, 4
lprq, 37
plot.rqss, 49
predict.rqss, 55
qss, 61
rqss, 94
rqss.object, 96
summary.rqss, 106
∗ **survival**
crq, 19
QTECox, 63
∗ **univar**
kuantile, 30
q489, 58
∗ **utilities**
combos, 17
latex.table, 34
sfn.control, 98
[.terms *(rqss)*, 94

AIC, *95*
AIC.nlrq *(nlrq)*, 41
AIC.rq *(rq.object)*, 89
AIC.rqs *(rq.object)*, 89
AIC.rqss *(rqss.object)*, 96
akj, 4, *29*

anova, *65*
anova.rq, 5
anova.rqlist *(anova.rq)*, 5
anova.rqs *(anova.rq)*, 5

bandwidth.rq, 8, *105*
barro, 9
boot.crq, 10, *101*
boot.rq, 11, *11*, *93*, *101*, *103*, *104*
boot.rq.pwxy, 14
boot.rq.pxy, *14*, 15, *15*, *82*
Bosco, 16

ChangeLog *(FAQ)*, 28
CobarOre, 17
coef.crq *(crq)*, 19
coef.nlrq *(nlrq)*, 41
coefficients, *90*
combos, 17
critval, 18
crq, 19, *64*, *102*
Curv *(crq)*, 19

dcauchy, *4*
deviance.nlrq *(nlrq)*, 41
dither, 23
dynrq, 24

end.dynrq *(dynrq)*, 24
engel, 27
extractAIC.nlrq *(nlrq)*, 41
extractAIC.rq *(rq.object)*, 89

FAQ, 28, *70*
fitted.nlrq *(nlrq)*, 41
fitted.rqss *(rqss.object)*, 96
formula.nlrq *(nlrq)*, 41
formula.rq *(rq.object)*, 89

gasprice, 29

Hill *(ParetoTest)*, 44

index.dynrq *(dynrq)*, 24

jitter, *24*

KhmaladzeTest, *8*, 29, *36*, *47*, *57*, *92*, *93*
kselect *(kuantile)*, 30
kuantile, 30

kunique (kuantile), 30

LassoLambdaHat, 32
latex, 33
latex.summary.rqs, 33, *33*
latex.table, *33*, 34, *34*
latex.table.rq (table.rq), 107
list, *4*, *99*
lm, *48*, *52*
lm.fit.recursive, 36
logLik, *95*
logLik.nlrq (nlrq), 41
logLik.rq (rq.object), 89
logLik.rqs (rq.object), 89
logLik.rqss (rqss.object), 96
lprq, 37

Mammals, 38
MelTemp, 39
merge.zoo, *26*
Munge, 40

na.approx, *25*
na.contiguous, *25*
na.fail, *25*
na.locf, *25*
na.omit, *25*
nlrq, 41, *43*, *67*, *70*
nlrq.control, *42*, 43
nlrqModel (nlrq), 41

optim, *41*
options, *25*

ParetoTest, 44
Peirce, 45
Pickands (ParetoTest), 44
plot.KhmaladzeTest, 47
plot.qss1 (plot.rqss), 49
plot.qss2 (plot.rqss), 49
plot.qts1 (plot.rqss), 49
plot.rq, 47
plot.rqs, 48, *52*
plot.rqss, *19*, 49, *98*, *107*
plot.summary.crqs (summary.crq), 101
plot.summary.rq (plot.summary.rqs), 51
plot.summary.rqs, *48*, 51, *52*
plot.summary.rqss (plot.rqss), 49
plot.table.rq (table.rq), 107
points, *52*
predict.crq (crq), 19
predict.crqs (crq), 19
predict.nlrq (nlrq), 41
predict.qss1 (predict.rqss), 55
predict.qss2 (predict.rqss), 55
predict.rq, 53
predict.rqs (predict.rq), 53
predict.rqss, 55
print.anova.rq (anova.rq), 5
print.crq (crq), 19
print.dynrq (dynrq), 24
print.dynrqs (dynrq), 24
print.Hill (ParetoTest), 44
print.KhmaladzeTest, 57
print.nlrq (nlrq), 41
print.Pickands (ParetoTest), 44
print.rq, 57
print.rqs (print.rq), 57
print.rqss (rqss.object), 96
print.summary.crq (summary.crq), 101
print.summary.crqs (summary.crq), 101
print.summary.dynrq (dynrq), 24
print.summary.dynrqs (dynrq), 24
print.summary.Hill (ParetoTest), 44
print.summary.nlrq (nlrq), 41
print.summary.Pickands (ParetoTest), 44
print.summary.rq, 58
print.summary.rqs (print.summary.rq), 58
print.summary.rqss (summary.rqss), 106

q489, 58
qrisk, 59, *77*, *78*, *100*
qss, 61, *94–97*, *106*
qss1 (qss), 61
qss2 (qss), 61
QTECox, 63, *102*
qts1 (qss), 61
quantile, *31*, *59*

ranks, *7*, *8*, 65
rearrange, *54*, 66, *66*
resid.rqss (rqss.object), 96
residuals.nlrq, *42*, 67
rq, *8*, *25*, *44*, *47*, *48*, *52*, *54*, *57*, *65*, *66*, 67, *71*, *73–76*, *79–81*, *83*, *85*, *90–92*, *105*, *108*
rq.fit, *24*, *70*, 71
rq.fit.br, *69*, *71*, 72, *75*, *76*, *81*, *83*, *92*

rq.fit.conquer, 73
rq.fit.fnb, *69*, *71*, *73*, 74, *92*
rq.fit.fnc, 75
rq.fit.hogg, *60*, 76
rq.fit.lasso, 78, *94*
rq.fit.pfn, *75, 76*, 79, *81–83*
rq.fit.pfnb, 80, *83*
rq.fit.ppro, *16*, 81
rq.fit.qfnb, 83
rq.fit.scad, 84
rq.fit.sfn, 85, *88*, *98, 99*
rq.fit.sfnc, 87, *98, 99*
rq.object, *69, 70*, *73*, 89
rq.process.object, *69, 70*, *73*, 90
rq.test.anowar *(*anova.rq*)*, 5
rq.test.rank, *7*, *65*
rq.test.rank *(*anova.rq*)*, 5
rq.wfit, *24*, *70*, 91
rqProcess, *29*, 92
rqs.fit, 93
rqss, *39*, *49*, *51*, *55, 56*, *62, 63*, *70*, 94, *98, 99*
rqss.object, *95*, 96

sfn.control, *95*, 98
sfnMessage *(*rq.fit.sfn*)*, 85
srisk, *60*, 100
start.dynrq *(*dynrq*)*, 24
summary, *69*
summary.crq, *11*, *22*, 101
summary.crqs *(*summary.crq*)*, 101
summary.dynrq *(*dynrq*)*, 24
summary.dynrqs *(*dynrq*)*, 24
summary.Hill *(*ParetoTest*)*, 44
summary.nlrq *(*nlrq*)*, 41
summary.Pickands *(*ParetoTest*)*, 44
summary.rcrqs *(*summary.rq*)*, 103
summary.rq, *13*, *29*, *44*, *52*, *58*, *70*, *75, 76*, *92*, 103, *106*
summary.rqs, *34*
summary.rqs *(*summary.rq*)*, 103
summary.rqss, 106

table.rq, 107
tau.nlrq *(*nlrq*)*, 41
time.dynrq *(*dynrq*)*, 24
triogram.fidelity *(*qss*)*, 61
triogram.penalty *(*qss*)*, 61

uis, 108
untangle.specials *(*rqss*)*, 94

zoo, *26*