Package 'rqPen'

June 4, 2024

Type Package

Title Penalized Quantile Regression

Version 4.1.1

Date 2024-05-31

Author Ben Sherwood [aut, cre], Adam Maidman [aut], Shaobo Li [aut]

Depends R (>= 3.0.0)

Imports methods, quantreg, hqreg, hrqglas, data.table, Rdpack, lifecycle, plyr, Matrix, Rcpp

LinkingTo Rcpp, RcppArmadillo

RdMacros Rdpack

Suggests splines, knitr

Maintainer Ben Sherwood <ben.sherwood@ku.edu>

Description

Performs penalized quantile regression with LASSO, elastic net, SCAD and MCP penalty functions including group penalties. In addition, offers a group penalty that provides consistent variable selection across quantiles. Provides a function that automatically generates lambdas and evaluates different models with cross validation or BIC, including a large p version of BIC. Below URL provides a link to a work in progress vignette.

ByteCompile TRUE

Encoding UTF-8

License MIT + file LICENSE

URL https://github.com/bssherwood/rqpen/blob/master/ignore/rqPenArticle.pdf

RoxygenNote 7.2.3

NeedsCompilation yes

Repository CRAN

Date/Publication 2024-06-04 13:30:01 UTC

Contents

`bytau.plot` *Plot of how coefficients change with tau*

Description

Plot of how coefficients change with tau

Usage

```
bytau.plot(x, ...)
```

Arguments

`x`	A rq.pen.seq or rq.pen.seq.cv object.
`...`	Additional arguments see bytau.plot.rq.pen.seq() or bytau.plot.rq.pen.seq.cv() for more information.

Value

Returns the plot of how coefficients change with tau.

Author(s)

Ben Sherwood, <ben.sherwood@ku.edu>

`bytau.plot.rq.pen.seq` *Plot of how coefficients change with tau.*

Description

Plot of how coefficients change with tau.

Usage

```
## S3 method for class 'rq.pen.seq'
bytau.plot(x, a = NULL, lambda = NULL, lambdaIndex = NULL, vars = NULL, ...)
```

Arguments

`x`	An rq.pen.seq object
`a`	The tuning parameter a of interest
`lambda`	The lambda value of interest.
`lambdaIndex`	The lambda index of interest. Only specify lambdaIndex or lambda, not both.
`vars`	Index of the variables to plot with 1 being the intercept, 2 being the first predictor, etc. Default is to include all variables.
`...`	Additional parameters sent to coef()

Value

A plot of coefficient values by tau.

Author(s)

Ben Sherwood, <ben.sherwood@ku.edu>

Examples

```
set.seed(1)
x <- matrix(rnorm(800),nrow=100)
y <- 1 + x[,1] - 3*x[,5] + rnorm(100)
lassoModels <- rq.pen(x,y,tau=seq(.1,.9,.1))
bytau.plot(lassoModels,lambda=lassoModels$lambda[5])
```

bytau.plot.rq.pen.seq.cv *Plot of coefficients varying by quantiles for rq.pen.seq.cv object*

Description

Produces plots of how coefficient estimates vary by quantile for models selected by using cross validation.

Usage

```
## S3 method for class 'rq.pen.seq.cv'
bytau.plot(
  x,
  septau = ifelse(x$fit$penalty != "gq", TRUE, FALSE),
  cvmin = TRUE,
  useDefaults = TRUE,
  vars = NULL,
  ...
)
```

Arguments

x	An rq.pen.seq.cv object
septau	Whether optimal tuning parameters are estimated separately for each quantile.
cvmin	Whether the minimum cv error should be used or the one standard error rule.
useDefaults	Set to FALSE if you want to use something besides minimum cv or 1se.
vars	Index of the variables to plot with 1 being the intercept, 2 being the first predictor, etc. Default is to include all variables.
...	Additional parameters sent to coef()

Value

Returns plots of coefficient estimates varying by quantile.

Author(s)

Ben Sherwood, <ben.sherwood@ku.edu>

Examples

```
set.seed(1)
x <- matrix(runif(800),nrow=100)
y <- 1 + x[,1] - 3*x[,5] + (1+x[,4])*rnorm(100)
lmcv <- rq.pen.cv(x,y,tau=seq(.1,.9,.1))
bytau.plot(lmcv)
```

`coef.rq.pen.seq` *Returns coefficients of a rq.pen.seq object*

Description

Returns coefficients of a rq.pen.seq object

Usage

```
## S3 method for class 'rq.pen.seq'
coef(
  object,
  tau = NULL,
  a = NULL,
  lambda = NULL,
  modelsIndex = NULL,
  lambdaIndex = NULL,
  ...
)
```

Arguments

`object`	rq.pen.seq object
`tau`	Quantile of interest. Default is NULL, which will return all quantiles. Should not be specified if modelsIndex is used.
`a`	Tuning parameter of a. Default is NULL, which returns coefficients for all values of a. Should not be specified if modelsIndex is used.
`lambda`	Tuning parameter of λ. Default is NULL, which returns coefficients for all values of λ.
`modelsIndex`	Index of the models for which coefficients should be returned. Does not need to be specified if tau or a are specified.
`lambdaIndex`	Index of the lambda values for which coefficients should be returned. Does not need to be specified if lambda is specified.
`...`	Additional parameters.

Value

A list of a matrix of coefficients for each tau and a combination

Author(s)

Ben Sherwood, <ben.sherwood@ku.edu>

Examples

```
x <- matrix(runif(800),ncol=8)
y <- 1 + x[,1] + x[,8] + (1+.5*x[,3])*rnorm(100)
m1 <- rq.pen(x,y,penalty="ENet",a=c(0,.5,1),tau=c(.25,.75),lambda=c(.1,.05,.01))
allCoefs <- coef(m1)
targetCoefs <- coef(m1,tau=.25,a=.5,lambda=.1)
idxApproach <- coef(m1,modelsIndex=2)
bothIdxApproach <- coef(m1,modelsIndex=2,lambdaIndex=1)
```

`coef.rq.pen.seq.cv` *Returns coefficients from a rq.pen.seq.cv object.*

Description

Returns coefficients from a rq.pen.seq.cv object.

Usage

```
## S3 method for class 'rq.pen.seq.cv'
coef(
  object,
  septau = ifelse(object$fit$penalty != "gq", TRUE, FALSE),
  cvmin = TRUE,
  useDefaults = TRUE,
  tau = NULL,
  ...
)
```

Arguments

`object`	An rq.pen.seq.cv object.
`septau`	Whether tuning parameter should be optimized separately for each quantile.
`cvmin`	If TRUE then minimum error is used, if FALSE then one standard error rule is used.
`useDefaults`	Whether the default results are used. Set to FALSE if you you want to specify specific models and lambda values.
`tau`	Quantiles of interest.
`...`	Additional parameters sent to coef.rq.pen.seq()

Value

Returns coefficients

Author(s)

Ben Sherwood, <ben.sherwood@ku.edu>

Examples

```
## Not run:
set.seed(1)
x <- matrix(rnorm(800),nrow=100)
y <- 1 + x[,1] - 3*x[,5] + rnorm(100)
lassoModels <- rq.pen.cv(x,y,tau=seq(.1,.9,.1))
coefficients(lassoModels,septau=FALSE)
coefficients(lassoModels,cvmin=FALSE)

## End(Not run)
```

`plot.rq.pen.seq` *Plot of coefficients of rq.pen.seq object as a function of lambda*

Description

Plot of coefficients of rq.pen.seq object as a function of lambda

Usage

```
## S3 method for class 'rq.pen.seq'
plot(
  x,
  vars = NULL,
  logLambda = TRUE,
  tau = NULL,
  a = NULL,
  lambda = NULL,
  modelsIndex = NULL,
  lambdaIndex = NULL,
  main = NULL,
  ...
)
```

Arguments

`x`	rq.pen.seq object
`vars`	Variables of interest
`logLambda`	Whether lambda should be reported on the log scale
`tau`	Quantiles of interest
`a`	Tuning parameter a values of interest.
`lambda`	Values of lambda of interest.
`modelsIndex`	Specific models of interest.
`lambdaIndex`	Specific lambda values of interest.
`main`	Title of the plots. Can be a vector of multiple titles if multiple plots are created.
`...`	Additional arguments sent to plot

Value

Returns plot(s) of coefficients as they change with lambda.

Author(s)

Ben Sherwood, <ben.sherwood@ku.edu>

Examples

```
set.seed(1)
x <- matrix(rnorm(100*8,sd=10),ncol=8)
y <- 1 + x[,1] + 3*x[,3] - x[,8] + rt(100,3)
m1 <- rq.pen(x,y,tau=c(.1,.5,.7),penalty="SCAD",a=c(3,4))
plot(m1,a=3,tau=.7)
plot(m1)
mlist <- list()
for(i in 1:6){
mlist[[i]] <- paste("Plot",i)
}
plot(m1,main=mlist)
```

---

plot.rq.pen.seq.cv *Plots cross validation results from a rq.pen.seq.cv object*

Description

Provides plots of cross-validation results by lambda. If septau is set to TRUE then plots the cross-validation results for each quantile. If septau is set to FALSE then provides one plot for cross-validation results across all quantiles.

Usage

```
## S3 method for class 'rq.pen.seq.cv'
plot(
  x,
  septau = ifelse(x$fit$penalty != "gq", TRUE, FALSE),
  tau = NULL,
  logLambda = TRUE,
  main = NULL,
  ...
)
```

Arguments

x	The rq.pen.seq.cv object
septau	If set to true then optimal tuning parameters are selected seperately for each quantile and there will be a different plot for each quanitle.

tau	Quantiles of interest.
logLambda	Whether log(lambda) is used for the x-axis
main	Title to the plot
...	Additional parameters sent to the plot function.

Value

Plots of the cross validation results by lambda.

Author(s)

Ben Sherwood, <ben.sherwood@ku.edu>

Examples

```
set.seed(1)
x <- matrix(rnorm(100*8,sd=1),ncol=8)
y <- 1 + x[,1] + 3*x[,3] - x[,8] + rt(100,3)
m1 <- rq.pen.cv(x,y,tau=c(.1,.3,.7))
plot(m1)
plot(m1,septau=FALSE)
```

---

predict.qic.select *Predictions from a qic.select object*

Description

Predictions from a qic.select object

Usage

```
## S3 method for class 'qic.select'
predict(object, newx, ...)
```

Arguments

object	qic.select object
newx	Data matrix to make predictions from.
...	optional arguments

Value

A matrix of predicted values.

Author(s)

Ben Sherwood, <ben.sherwood@ku.edu>

Examples

```
x <- matrix(runif(800),ncol=8)
y <- 1 + x[,1] + x[,8] + (1+.5*x[,3])*rnorm(100)
m1 <- rq.pen(x,y,tau=c(.25,.75))
q1 <- qic.select(m1)
newx <- matrix(runif(80),ncol=8)
preds <- predict(q1,newx)
```

`predict.rq.pen.seq` *Predictions from rq.pen.seq object*

Description

Predictions from rq.pen.seq object

Usage

```
## S3 method for class 'rq.pen.seq'
predict(
  object,
  newx,
  tau = NULL,
  a = NULL,
  lambda = NULL,
  modelsIndex = NULL,
  lambdaIndex = NULL,
  ...
)
```

Arguments

`object`	rq.pen.seq object
`newx`	Matrix of predictors
`tau`	Quantile of interest. Default is NULL, which will return all quantiles. Should not be specified if modelsIndex is used.
`a`	Tuning parameter of a. Default is NULL, which returns coefficients for all values of a. Should not be specified if modelsIndex is used.
`lambda`	Tuning parameter of λ. Default is NULL, which returns coefficients for all values of λ.
`modelsIndex`	Index of the models for which coefficients should be returned. Does not need to be specified if tau or a are specified.
`lambdaIndex`	Index of the lambda values for which coefficients should be returned. Does not need to be specified if lambda is specified.
`...`	Additional parameters passed to coef.rq.pen.seq()

Value

A matrix of predictions for each tau and a combination

Author(s)

Ben Sherwood, <ben.sherwood@ku.edu>

Examples

```
x <- matrix(runif(800),ncol=8)
y <- 1 + x[,1] + x[,8] + (1+.5*x[,3])*rnorm(100)
m1 <- rq.pen(x,y,penalty="ENet",a=c(0,.5,1),tau=c(.25,.75),lambda=c(.1,.05,.01))
newx <- matrix(runif(80),ncol=8)
allCoefs <- predict(m1,newx)
targetCoefs <- predict(m1,newx,tau=.25,a=.5,lambda=.1)
idxApproach <- predict(m1,newx,modelsIndex=2)
bothIdxApproach <- predict(m1,newx,modelsIndex=2,lambdaIndex=1)
```

---

predict.rq.pen.seq.cv *Predictions from rq.pen.seq.cv object*

Description

Predictions from rq.pen.seq.cv object

Usage

```
## S3 method for class 'rq.pen.seq.cv'
predict(
  object,
  newx,
  tau = NULL,
  septau = ifelse(object$fit$penalty != "gq", TRUE, FALSE),
  cvmin = TRUE,
  useDefaults = TRUE,
  ...
)
```

Arguments

object	rq.pen.seq.cv object
newx	Matrix of predictors
tau	Quantile of interest. Default is NULL, which will return all quantiles. Should not be specified if modelsIndex is used.
septau	Whether tuning parameter should be optimized separately for each quantile.
cvmin	If TRUE then minimum error is used, if FALSE then one standard error rule is used.

useDefaults	Whether the default results are used. Set to FALSE if you you want to specify specific models and lambda values.
...	Additional parameters sent to coef.rq.pen.seq.cv().

Value

A matrix of predictions for each tau and a combination

Author(s)

Ben Sherwood, <ben.sherwood@ku.edu>

Examples

```
x <- matrix(runif(1600),ncol=8)
y <- 1 + x[,1] + x[,8] + (1+.5*x[,3])*rnorm(200)
m1 <- rq.pen.cv(x,y,penalty="ENet",a=c(0,.5,1),tau=c(.25,.75),lambda=c(.1,.05,.01))
newx <- matrix(runif(80),ncol=8)
cvpreds <- predict(m1,newx)
```

---

print.qic.select *Print a qic.select object*

Description

Print a qic.select object

Usage

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

Arguments

x	qic.select object
...	optional arguments

Value

Prints the coefficients of the qic.select object

Author(s)

Ben Sherwood, <ben.sherwood@ku.edu>

print.rq.pen.seq *Print a rq.pen.seq object*

Description

Print a rq.pen.seq object

Usage

```
## S3 method for class 'rq.pen.seq'
print(x, ...)
```

Arguments

x	rq.pen.seq object
...	optional arguments

Value

If only one model, prints a data.frame of the number of nonzero coefficients and lambda. Otherwise prints information about the quantiles being modeled and choices for a.

Author(s)

Ben Sherwood, <ben.sherwood@ku.edu>

print.rq.pen.seq.cv *Prints a rq.pen.seq.cv object*

Description

Prints a rq.pen.seq.cv object

Usage

```
## S3 method for class 'rq.pen.seq.cv'
print(x, ...)
```

Arguments

x	A req.pen.seq.cv object.
...	Additional arguments.

Value

Print of btr and gtr from a rq.pen.seq.cv object. If only one quantile is modeled then only btr is returned.

`qic.select` *Select tuning parameters using IC*

Description

Selects tuning parameter λ and a according to information criterion of choice. For a given $\hat{\beta}$ the information criterion is calculated as

$$\log(\sum_{i=1}^{n} w_i \rho_\tau(y_i - x_i^\top \hat{\beta})) + d * b/(2n),$$

where d is the number of nonzero coefficients and b depends on the method used. For AIC $b = 2$, for BIC $b = log(n)$ and for PBIC $d = log(n) * log(p)$ where p is the dimension of $\hat{\beta}$. If septau set to FALSE then calculations are made across the quantiles. Let $\hat{\beta}^q$ be the coefficient vector for the qth quantile of Q quantiles. In addition let d_q and b_q be d and b values from the qth quantile model. Note, for all of these we are assuming eqn and a are the same. Then the summary across all quantiles is

$$\sum_{q=1}^{Q} w_q[\log(\sum_{i=1}^{n} m_i \rho_\tau(y_i - x_i^\top \hat{\beta}^q)) + d_q * b_q/(2n)],$$

where w_q is the weight assigned for the qth quantile model.

Usage

```
qic.select(obj, ...)
```

Arguments

`obj`	A rq.pen.seq or rq.pen.seq.cv object.
`...`	Additional arguments see qic.select.rq.pen.seq() or qic.select.rq.pen.seq.cv() for more information.

Value

Returns a qic.select object.

Author(s)

Ben Sherwood, <ben.sherwood@ku.edu>

References

Lee ER, Noh H, Park BU (2014). "Model Selection via Bayesian Information Criterion for Quantile Regression Models." *Journal of the American Statistical Association*, **109**(505), 216–229. ISSN 01621459.

Examples

```
set.seed(1)
x <- matrix(runif(800),ncol=8)
y <- 1 + x[,1] + x[,8] + (1+.5*x[,3])*rnorm(100)
m1 <- rq.pen(x,y,penalty="ENet",a=c(0,.5,1),tau=c(.25,.75))
qic.select(m1)
```

`qic.select.rq.pen.seq` *Select tuning parameters using IC*

Description

Selects tuning parameter λ and a according to information criterion of choice. For a given $\hat{\beta}$ the information criterion is calculated as

$$\log(\sum_{i=1}^{n} w_i \rho_\tau(y_i - x_i^\top \hat{\beta})) + d * b/(2n),$$

where d is the number of nonzero coefficients and b depends on the method used. For AIC $b = 2$, for BIC $b = log(n)$ and for PBIC $d = log(n) * log(p)$ where p is the dimension of $\hat{\beta}$. If septau set to FALSE then calculations are made across the quantiles. Let $\hat{\beta}^q$ be the coefficient vector for the qth quantile of Q quantiles. In addition let d_q and b_q be d and b values from the qth quantile model. Note, for all of these we are assuming eqn and a are the same. Then the summary across all quantiles is

$$\sum_{q=1}^{Q} w_q[\log(\sum_{i=1}^{n} m_i \rho_\tau(y_i - x_i^\top \hat{\beta}^q)) + d_q * b_q/(2n)],$$

where w_q is the weight assigned for the qth quantile model.

Usage

```
## S3 method for class 'rq.pen.seq'
qic.select(
  obj,
  method = c("BIC", "AIC", "PBIC"),
  septau = ifelse(obj$penalty != "gq", TRUE, FALSE),
  tauWeights = NULL,
  ...
)
```

Arguments

`obj`	A rq.pen.seq or rq.pen.seq.cv object.
`method`	Choice of BIC, AIC or PBIC, a large p BIC.
`septau`	If optimal values of λ and a can vary with τ. Default is TRUE.
`tauWeights`	Weights for each quantile. Useful if you set septau to FALSE but want different weights for the different quantiles. If not specified default is to have $w_q = 1$ for all quantiles.
`...`	Additional arguments.

Value

coefficients Coefficients of the selected models.

ic Information criterion values for all considered models.

modelsInfo Model info for the selected models related to the original object obj.

gic Information criterion summarized across all quantiles. Only returned if septau set to FALSE

Author(s)

Ben Sherwood, <ben.sherwood@ku.edu>

References

Lee ER, Noh H, Park BU (2014). "Model Selection via Bayesian Information Criterion for Quantile Regression Models." *Journal of the American Statistical Association*, **109**(505), 216–229. ISSN 01621459.

Examples

```
set.seed(1)
x <- matrix(runif(800),ncol=8)
y <- 1 + x[,1] + x[,8] + (1+.5*x[,3])*rnorm(100)
m1 <- rq.pen(x,y,penalty="ENet",a=c(0,.5,1),tau=c(.25,.75))
qic.select(m1)
```

`qic.select.rq.pen.seq.cv`

Select tuning parameters using IC

Description

Selects tuning parameter λ and a according to information criterion of choice. For a given $\hat{\beta}$ the information criterion is calculated as

$$\log(\sum_{i=1}^{n} w_i \rho_\tau(y_i - x_i^\top \hat{\beta})) + d * b/(2n),$$

where d is the number of nonzero coefficients and b depends on the method used. For AIC $b = 2$, for BIC $b = log(n)$ and for PBIC $d = log(n) * log(p)$ where p is the dimension of $\hat{\beta}$. If septau set to FALSE then calculations are made across the quantiles. Let $\hat{\beta}^q$ be the coefficient vector for the qth quantile of Q quantiles. In addition let d_q and b_q be d and b values from the qth quantile model. Note, for all of these we are assuming eqn and a are the same. Then the summary across all quantiles is

$$\sum_{q=1}^{Q} w_q[\log(\sum_{i=1}^{n} \rho_\tau(y_i - x_i^\top \hat{\beta}^q)) + d_q * b_q/(2n)],$$

where w_q is the weight assigned for the qth quantile model.

Usage

```
## S3 method for class 'rq.pen.seq.cv'
qic.select(
  obj,
  method = c("BIC", "AIC", "PBIC"),
  septau = ifelse(obj$fit$penalty != "gq", TRUE, FALSE),
  weights = NULL,
  ...
)
```

Arguments

obj	A rq.pen.seq.cv object.
method	Choice of BIC, AIC or PBIC, a large p BIC.
septau	If optimal values of λ and a can vary with τ. Default is TRUE.
weights	Weights for each quantile. Useful if you set septau to FALSE but want different weights for the different quantiles. If not specified default is to have $w_q = 1$ for all quantiles.
...	Additional arguments.

Value

coefficients Coefficients of the selected models.

ic Information criterion values for all considered models.

modelsInfo Model info for the selected models related to the original object obj.

gic Information criterion summarized across all quantiles. Only returned if septau set to FALSE

Author(s)

Ben Sherwood, <ben.sherwood@ku.edu>

References

Lee ER, Noh H, Park BU (2014). "Model Selection via Bayesian Information Criterion for Quantile Regression Models." *Journal of the American Statistical Association*, **109**(505), 216–229. ISSN 01621459.

Examples

```
set.seed(1)
x <- matrix(runif(800),ncol=8)
y <- 1 + x[,1] + x[,8] + (1+.5*x[,3])*rnorm(100)
m1 <- rq.pen.cv(x,y,penalty="ENet",a=c(0,.5,1),tau=c(.25,.75))
qic.select(m1)
```

rq.gq.pen *Title Quantile regression estimation and consistent variable selection across multiple quantiles*

Description

Uses the group lasso penalty across the quantiles to provide consistent selection across all, K, modeled quantiles. Let β^q be the coefficients for the kth quantiles, β_j be the Q-dimensional vector of the jth coefficient for each quantile, and $\rho_\tau(u)$ is the quantile loss function. The method minimizes

$$\sum_{q=1}^{Q} \frac{1}{n} \sum_{i=1}^{n} m_i \rho_\tau(y_i - x_i^\top \beta^q) + \lambda \sum_{j=1}^{p} ||\beta_j||_{2,w}.$$

Uses a Huber approximation in the fitting of model, as presented in Sherwood and Li (2022). Where,

$$||\beta_j||_{2,w} = \sqrt{\sum_{k=1}^{K} w_k v_j \beta_{kj}^2},$$

where w_k is a quantile weight that can be specified by `tau.penalty.factor`, v_j is a predictor weight that can be assigned by `penalty.factor`, and m_i is an observation weight that can be set by `weights`.

Usage

```
rq.gq.pen(
  x,
  y,
  tau,
  lambda = NULL,
  nlambda = 100,
  eps = ifelse(nrow(x) < ncol(x), 0.01, 0.001),
  weights = NULL,
  penalty.factor = NULL,
  scalex = TRUE,
  tau.penalty.factor = NULL,
  gmma = 0.2,
  max.iter = 200,
  lambda.discard = TRUE,
  converge.eps = 1e-04,
  beta0 = NULL
)
```

Arguments

x	covariate matrix
y	a univariate response variable

tau	a sequence of quantiles to be modeled, must be of at least length 3.
lambda	shrinkage parameter. Default is NULL, and the algorithm provides a solution path.
nlambda	Number of lambda values to be considered.
eps	If not pre-specified the lambda vector will be from lambda_max to lambda_max times eps
weights	observation weights. Default is NULL, which means equal weights.
penalty.factor	weights for the shrinkage parameter for each covariate. Default is equal weight.
scalex	Whether x should be scaled before fitting the model. Coefficients are returned on the original scale.
tau.penalty.factor	weights for different quantiles. Default is equal weight.
gmma	tuning parameter for the Huber loss
max.iter	maximum number of iteration. Default is 200.
lambda.discard	Default is TRUE, meaning that the solution path stops if the relative deviance changes sufficiently small. It usually happens near the end of solution path. However, the program returns at least 70 models along the solution path.
converge.eps	The epsilon level convergence. Default is 1e-4.
beta0	Initial estimates. Default is NULL, and the algorithm starts with the intercepts being the quantiles of response variable and other coefficients being zeros.

Value

An rq.pen.seq object.

models: A list of each model fit for each tau and a combination.

n: Sample size.

p: Number of predictors.

alg: Algorithm used. Options are "huber" or any method implemented in rq(), such as "br".

tau: Quantiles modeled.

a: Tuning parameters a used.

modelsInfo: Information about the quantile and a value for each model.

lambda: Lambda values used for all models. If a model has fewer coefficients than lambda, say k. Then it used the first k values of lambda. Setting lambda.discard to TRUE will gurantee all values use the same lambdas, but may increase computational time noticeably and for little gain.

penalty: Penalty used.

call: Original call.

Each model in the models list has the following values.

coefficients: Coefficients for each value of lambda.

rho: The unpenalized objective function for each value of lambda.

PenRho: The penalized objective function for each value of lambda.

nzero: The number of nonzero coefficients for each value of lambda.

tau: Quantile of the model.

a: Value of a for the penalized loss function.

Author(s)

Shaobo Li and Ben Sherwood, <ben.sherwood@ku.edu>

References

Wang M, Kang X, Liang J, Wang K, Wu Y (2024). "Heteroscedasticity identification and variable selection via multiple quantile regression." *Journal of Statistical Computation and Simulation*, **94**(2), 297-314.

Sherwood B, Li S (2022). "Quantile regression feature selection and estimation with grouped variables using Huber approximation." *Statistics and Computing*, **32**(5), 75.

Examples

```
## Not run:
n<- 200
p<- 10
X<- matrix(rnorm(n*p),n,p)
y<- -2+X[,1]+0.5*X[,2]-X[,3]-0.5*X[,7]+X[,8]-0.2*X[,9]+rt(n,2)
taus <- seq(0.1, 0.9, 0.2)
fit<- rq.gq.pen(X, y, taus)
#use IC to select best model, see rq.gq.pen.cv() for a cross-validation approach
qfit <- qic.select(fit)

## End(Not run)
```

`rq.gq.pen.cv` *Title Cross validation for consistent variable selection across multiple quantiles.*

Description

Title Cross validation for consistent variable selection across multiple quantiles.

Usage

```
rq.gq.pen.cv(
  x = NULL,
  y = NULL,
  tau = NULL,
  lambda = NULL,
  nfolds = 10,
```

```
  cvFunc = c("rq", "se"),
  tauWeights = NULL,
  foldid = NULL,
  printProgress = FALSE,
  weights = NULL,
  ...
)
```

Arguments

x	covariate matrix. Not needed if model_obj is supplied.
y	univariate response. Not needed if model_obj is supplied.
tau	a sequence of tau to be modeled, must be at least of length 3.
lambda	Values of λ. Default will automatically select the λ values.
nfolds	number of folds
cvFunc	loss function to be evaluated for cross-validation. Supported loss functions include quantile ("rq") and squared loss("se"). Default is the quantile loss.
tauWeights	weights for different quantiles in calculating the cv error. Default is equal weight.
foldid	indices of pre-split testing obervations
printProgress	If set to TRUE prints which partition is being worked on.
weights	Weights for the quantile loss objective function.
...	other arguments for rq.gq.pen.cv sent to rq.gq.pen

Details

Let $y_{b,i}$ and $x_{b,i}$ index the observations in fold b. Let $\hat{\beta}^{-b}_{\tau,a,\lambda}$ be the estimator for a given quantile and tuning parameters that did not use the bth fold. Let n_b be the number of observations in fold b. Then the cross validation error for fold b is

$$\text{CV}(b,\tau) = \sum_{q=1}^{Q} \frac{1}{n_b} \sum_{i=1}^{n_b} m_{b,i} v_q \rho_\tau (y_{b,i} - x_{b,i}^\top \hat{\beta}^{-b}_{\tau_q,a,\lambda}).$$

Where, $m_{b,i}$ is the weight for the ith observation in fold b and v_q is a quantile specific weight. Note that $\rho_\tau()$ can be replaced squared error loss. Provides results about how the average of the cross-validation error changes with λ. Uses a Huber approximation in the fitting of model, as presented in Sherwood and Li (2022).

Value

An rq.pen.seq.cv object.

cverr: Matrix of cvSummary function, default is average, cross-validation error for each model, tau and a combination, and lambda.

cvse: Matrix of the standard error of cverr foreach model, tau and a combination, and lambda.

fit: The rq.pen.seq object fit to the full data.

btr: Let blank, unlike rq.pen.seq.cv() or rq.group.pen.cv(), because optmizes the quantiles individually does not make sense with this penalty.

gtr: A data.table for the combination of a and lambda that minimize the cross validation error across all tau.

gcve: Group, across all quantiles, cross-validation error results for each value of a and lambda.

call: Original call to the function.

Author(s)

Shaobo Li and Ben Sherwood, <ben.sherwood@ku.edu>

References

Wang M, Kang X, Liang J, Wang K, Wu Y (2024). "Heteroscedasticity identification and variable selection via multiple quantile regression." *Journal of Statistical Computation and Simulation*, **94**(2), 297-314.

Sherwood B, Li S (2022). "Quantile regression feature selection and estimation with grouped variables using Huber approximation." *Statistics and Computing*, **32**(5), 75.

Examples

```
## Not run:
n<- 200
p<- 10
X<- matrix(rnorm(n*p),n,p)
y<- -2+X[,1]+0.5*X[,2]-X[,3]-0.5*X[,7]+X[,8]-0.2*X[,9]+rt(n,2)
taus <- seq(0.1, 0.9, 0.2)
cvfit<- rq.gq.pen.cv(x=X, y=y, tau=taus)
cvCoefs <- coefficients(cvfit)

## End(Not run)
```

---

rq.group.pen *Fits quantile regression models using a group penalized objective function.*

Description

Let the predictors be divided into G groups with G corresponding vectors of coefficients, $\beta_1, \ldots, \beta_G$. Let $\rho_\tau(a) = a[\tau - I(a < 0)]$. Fits quantile regression models for Q quantiles by minimizing the penalized objective function of

$$\sum_{q=1}^{Q} \frac{1}{n} \sum_{i=1}^{n} m_i \rho_\tau(y_i - x_i^\top \beta^q) + \sum_{q=1}^{Q} \sum_{g=1}^{G} P(||\beta_g^q||_k, w_q * v_j * \lambda, a).$$

Where w_q and v_j are designated by penalty.factor and tau.penalty.factor respectively and m_i can be set by weights. The value of k is chosen by norm. Value of P() depends on the penalty. Briefly, but see references or vignette for more details,

Group LASSO (gLASSO) $P(||\beta||_k, \lambda, a) = \lambda||\beta||_k$

Group SCAD $P(||\beta||_k, \lambda, a) = SCAD(||\beta||_k, \lambda, a)$

Group MCP $P(||\beta||_k, \lambda, a) = MCP(||\beta||_k, \lambda, a)$

Group Adaptive LASSO $P(||\beta||_k, \lambda, a) = \frac{\lambda||\beta||_k}{|\beta_0|^a}$

Note if $k = 1$ and the group lasso penalty is used then this is identical to the regular lasso and thus function will stop and suggest that you use rq.pen() instead. For Adaptive LASSO the values of β_0 come from a Ridge solution with the same value of λ. If the Huber algorithm is used than $\rho_\tau(y_i - x_i^\top \beta)$ is replaced by a Huber-type approximation. Specifically, it is replaced by $h_\gamma^\tau(y_i - x_i^\top \beta)/2$ where

$$h_\gamma^\tau(a) = a^2/(2\gamma)I(|a| \leq \gamma) + (|a| - \gamma/2)I(|a| > \gamma) + (2\tau - 1)a.$$

Where if $\tau = .5$, we get the usual Huber loss function.

Usage

```
rq.group.pen(
  x,
  y,
  tau = 0.5,
  groups = 1:ncol(x),
  penalty = c("gLASSO", "gAdLASSO", "gSCAD", "gMCP"),
  lambda = NULL,
  nlambda = 100,
  eps = ifelse(nrow(x) < ncol(x), 0.05, 0.01),
  alg = c("huber", "br"),
  a = NULL,
  norm = 2,
  group.pen.factor = NULL,
  tau.penalty.factor = rep(1, length(tau)),
  scalex = TRUE,
  coef.cutoff = 1e-08,
  max.iter = 500,
  converge.eps = 1e-04,
  gamma = IQR(y)/10,
  lambda.discard = TRUE,
  weights = NULL,
  ...
)
```

Arguments

x	Matrix of predictors.
y	Vector of responses.
tau	Vector of quantiles.
groups	Vector of group assignments for predictors.
penalty	Penalty used, choices are group lasso ("gLASSO"), group adaptive lasso ("gAdLASSO"), group SCAD ("gSCAD") and group MCP ("gMCP")

lambda	Vector of lambda tuning parameters. Will be autmoatically generated if it is not set.
nlambda	The number of lambda tuning parameters.
eps	The value to be multiplied by the largest lambda value to determine the smallest lambda value.
alg	Algorithm used. Choices are Huber approximation ("huber") or linear programming ("lp").
a	The additional tuning parameter for adaptive lasso, SCAD and MCP.
norm	Whether a L1 or L2 norm is used for the grouped coefficients.
group.pen.factor	Penalty factor for each group. Default is 1 for all groups if norm=1 and square root of group size if norm=2.
tau.penalty.factor	Penalty factor for each quantile.
scalex	Whether X should be centered and scaled so that the columns have mean zero and standard deviation of one. If set to TRUE, the coefficients will be returned to the original scale of the data.
coef.cutoff	Coefficient cutoff where any value below this number is set to zero. Useful for the lp algorithm, which are prone to finding almost, but not quite, sparse solutions.
max.iter	The maximum number of iterations for the algorithm.
converge.eps	The convergence criteria for the algorithms.
gamma	The tuning parameter for the Huber loss.
lambda.discard	Whether lambdas should be discarded if for small values of lambda there is very little change in the solutions.
weights	Weights used in the quanitle loss objective function.
...	Additional parameters

Value

An rq.pen.seq object.

models A list of each model fit for each tau and a combination.

n Sample size.

p Number of predictors.

alg Algorithm used.

tau Quantiles modeled.

penalty Penalty used.

a Tuning parameters a used.

lambda Lambda values used for all models. If a model has fewer coefficients than lambda, say k. Then it used the first k values of lambda. Setting lambda.discard to TRUE will gurantee all values use the same lambdas, but may increase computational time noticeably and for little gain.

modelsInfo Information about the quantile and a value for each model.

call Original call.

Each model in the models list has the following values.

coefficients Coefficients for each value of lambda.

rho The unpenalized objective function for each value of lambda.

PenRho The penalized objective function for each value of lambda.

nzero The number of nonzero coefficients for each value of lambda.

tau Quantile of the model.

a Value of a for the penalized loss function.

Author(s)

Ben Sherwood, <ben.sherwood@ku.edu>, Shaobo Li <shaobo.li@ku.edu> and Adam Maidman

References

Peng B, Wang L (2015). "An iterative coordinate descent algorithm for high-dimensional nonconvex penalized quantile regression." *J. Comput. Graph. Statist.*, **24**(3), 676-694.

Examples

```
## Not run:
set.seed(1)
x <- matrix(rnorm(200*8,sd=1),ncol=8)
y <- 1 + x[,1] + 3*x[,3] - x[,8] + rt(200,3)
g <- c(1,1,1,2,2,2,3,3)
tvals <- c(.25,.75)
r1 <- rq.group.pen(x,y,groups=g)
r5 <- rq.group.pen(x,y,groups=g,tau=tvals)
#Linear programming approach with group SCAD penalty and L1-norm
m2 <- rq.group.pen(x,y,groups=g,alg="br",penalty="gSCAD",norm=1,a=seq(3,4))
# No penalty for the first group
m3 <- rq.group.pen(x,y,groups=g,group.pen.factor=c(0,rep(1,2)))
# Smaller penalty for the median
m4 <- rq.group.pen(x,y,groups=g,tau=c(.25,.5,.75),tau.penalty.factor=c(1,.25,1))

## End(Not run)
```

`rq.group.pen.cv` *Performs cross validation for a group penalty.*

Description

Performs cross validation for a group penalty.

Usage

```
rq.group.pen.cv(
  x,
  y,
  tau = 0.5,
  groups = 1:ncol(x),
  lambda = NULL,
  a = NULL,
  cvFunc = NULL,
  nfolds = 10,
  foldid = NULL,
  groupError = TRUE,
  cvSummary = mean,
  tauWeights = rep(1, length(tau)),
  printProgress = FALSE,
  weights = NULL,
  ...
)
```

Arguments

x	Matrix of predictors.
y	Vector of responses.
tau	Vector of quantiles.
groups	Vector of group assignments for the predictors.
lambda	Vector of lambda values, if set to NULL they will be generated automatically.
a	Vector of the other tuning parameter values.
cvFunc	Function used for cross-validation error, default is quantile loss.
nfolds	Number of folds used for cross validation.
foldid	Fold assignments, if not set this will be randomly created.
groupError	If errors are to be reported as a group or as the average for each fold.
cvSummary	The
tauWeights	Weights for the tau penalty only used in group tau results (gtr).
printProgress	If set to TRUE will print which fold the process is working on.
weights	Weights for the quantile loss function. Used in both model fitting and cross-validation.
...	Additional parameters that will be sent to rq.group.pen().

Details

Two cross validation results are returned. One that considers the best combination of a and lambda for each quantile. The second considers the best combination of the tuning parameters for all quantiles. Let $y_{b,i}$, $x_{b,i}$, and $m_{b,i}$ index the response, predictors, and weights of observations in fold

b. Let $\hat{\beta}^{-b}_{\tau,a,\lambda}$ be the estimator for a given quantile and tuning parameters that did not use the bth fold. Let n_b be the number of observations in fold b. Then the cross validation error for fold b is

$$\mathrm{CV}(b,\tau) = \frac{1}{n_b}\sum_{i=1}^{n_b} m_{b,i}\rho_\tau(y_{b,i} - x_{b,i}^\top \hat{\beta}^{-b}_{\tau,a,\lambda}).$$

Note that $\rho_\tau()$ can be replaced by a different function by setting the cvFunc parameter. The function returns two different cross-validation summaries. The first is btr, by tau results. It provides the values of `lambda` and `a` that minimize the average, or whatever function is used for `cvSummary`, of $\mathrm{CV}(b)$. In addition it provides the sparsest solution that is within one standard error of the minimum results.

The other approach is the group tau results, gtr. Consider the case of estimating Q quantiles of $\tau_1,\ldots,\tau_Q$ with quantile (tauWeights) of v_q. The gtr returns the values of `lambda` and `a` that minimizes the average, or again whatever function is used for `cvSummary`, of

$$\sum_{q=1}^{Q} v_q \mathrm{CV}(b,\tau_q).$$

If only one quantile is modeled then the gtr results can be ignored as they provide the same minimum solution as btr.

Value

An rq.pen.seq.cv object.

cverr Matrix of cvSummary function, default is average, cross-validation error for each model, tau and a combination, and lambda.

cvse Matrix of the standard error of cverr foreach model, tau and a combination, and lambda.

fit The rq.pen.seq object fit to the full data.

btr A data.table of the values of a and lambda that are best as determined by the minimum cross validation error and the one standard error rule, which fixes a. In btr the values of lambda and a are selected seperately for each quantile.

gtr A data.table for the combination of a and lambda that minimize the cross validation error across all tau.

gcve Group, across all quantiles, cross-validation error results for each value of a and lambda.

call Original call to the function.

Author(s)

Ben Sherwood, <ben.sherwood@ku.edu> and Shaobo Li <shaobo.li@ku.edu>

Examples

```
set.seed(1)
x <- matrix(rnorm(100*8,sd=1),ncol=8)
y <- 1 + x[,1] + 3*x[,3] - x[,8] + rt(100,3)
g <- c(1,1,1,1,2,2,3,3)
tvals <- c(.25,.75)
```

```
## Not run:
m1 <- rq.group.pen.cv(x,y,tau=c(.1,.3,.7),groups=g)
m2 <- rq.group.pen.cv(x,y,penalty="gAdLASSO",tau=c(.1,.3,.7),groups=g)
m3 <- rq.group.pen.cv(x,y,penalty="gSCAD",tau=c(.1,.3,.7),a=c(3,4,5),groups=g)
m4 <- rq.group.pen.cv(x,y,penalty="gMCP",tau=c(.1,.3,.7),a=c(3,4,5),groups=g)

## End(Not run)
```

`rq.pen` *Fit a quantile regression model using a penalized quantile loss function.*

Description

Let q index the Q quantiles of interest. Let $\rho_\tau(a) = a[\tau - I(a < 0)]$. Fits quantile regression models by minimizing the penalized objective function of

$$\frac{1}{n}\sum_{q=1}^{Q}\sum_{i=1}^{n} m_i \rho_\tau(y_i - x_i^\top \beta^q) + \sum_{q=1}^{Q}\sum_{j=1}^{p} P(\beta_p^q, w_q * v_j * \lambda, a).$$

Where w_q and v_j are designated by penalty.factor and tau.penalty.factor respectively, and m_i is designated by weights. Value of $P()$ depends on the penalty. See references or vignette for more details,

LASSO: $P(\beta, \lambda, a) = \lambda|\beta|$

SCAD: $P(\beta, \lambda, a) = SCAD(\beta, \lambda, a)$

MCP: $P(\beta, \lambda, a) = MCP(\beta, \lambda, a)$

Ridge: $P(\beta, \lambda, a) = \lambda\beta^2$

Elastic Net: $P(\beta, \lambda, a) = a * \lambda|\beta| + (1 - a) * \lambda * \beta^2$

Adaptive LASSO: $P(\beta, \lambda, a) = \frac{\lambda|\beta|}{|\beta_0|^a}$

For Adaptive LASSO the values of β_0 come from a Ridge solution with the same value of λ. Three different algorithms are implemented

huber: Uses a Huber approximation of the quantile loss function. See Yi and Huang 2017 for more details.

br: Solution is found by re-formulating the problem so it can be solved with the rq() function from quantreg with the br algorithm.

The huber algorithm offers substantial speed advantages without much, if any, loss in performance. However, it should be noted that it solves an approximation of the quantile loss function.

Usage

```
rq.pen(
  x,
  y,
  tau = 0.5,
  lambda = NULL,
  penalty = c("LASSO", "Ridge", "ENet", "aLASSO", "SCAD", "MCP"),
  a = NULL,
  nlambda = 100,
  eps = ifelse(nrow(x) < ncol(x), 0.05, 0.01),
  penalty.factor = rep(1, ncol(x)),
  alg = c("huber", "br", "QICD", "fn"),
  scalex = TRUE,
  tau.penalty.factor = rep(1, length(tau)),
  coef.cutoff = 1e-08,
  max.iter = 10000,
  converge.eps = 1e-07,
  lambda.discard = TRUE,
  weights = NULL,
  ...
)
```

Arguments

x	matrix of predictors
y	vector of responses
tau	vector of quantiles
lambda	vector of lambda, if not set will be generated automatically
penalty	choice of penalty
a	Additional tuning parameter, not used for lasso or ridge penalties. However, will be set to the elastic net values of 1 and 0 respectively. Defaults are ENet(0), aLASSO(1), SCAD(3.7) and MCP(3).
nlambda	number of lambda, ignored if lambda is set
eps	If not pre-specified the lambda vector will be from lambda_max to lambda_max times eps
penalty.factor	penalty factor for the predictors
alg	Algorithm used.
scalex	Whether x should be scaled before fitting the model. Coefficients are returned on the original scale.
tau.penalty.factor	A penalty factor for each quantile.
coef.cutoff	Some of the linear programs will provide very small, but not sparse solutions. Estimates below this number will be set to zero. This is ignored if a non-linear programming algorithm is used.

`max.iter`	Maximum number of iterations of non-linear programming algorithms.
`converge.eps`	Convergence threshold for non-linear programming algorithms.
`lambda.discard`	Algorithm may stop for small values of lambda if the coefficient estimates are not changing drastically. One example of this is it is possible for the LLA weights of the non-convex functions to all become zero and smaller values of lambda are extremely likely to produce the same zero weights.
`weights`	Weights for the quantile objective function.
`...`	Extra parameters.

Value

An rq.pen.seq object.

models: A list of each model fit for each tau and a combination.

n: Sample size.

p: Number of predictors.

alg: Algorithm used. Options are "huber" or any method implemented in rq(), such as "br".

tau: Quantiles modeled.

a: Tuning parameters a used.

modelsInfo: Information about the quantile and a value for each model.

lambda: Lambda values used for all models. If a model has fewer coefficients than lambda, say k. Then it used the first k values of lambda. Setting lambda.discard to TRUE will gurantee all values use the same lambdas, but may increase computational time noticeably and for little gain.

penalty: Penalty used.

call: Original call.

Each model in the models list has the following values.

coefficients: Coefficients for each value of lambda.

rho: The unpenalized objective function for each value of lambda.

PenRho: The penalized objective function for each value of lambda.

nzero: The number of nonzero coefficients for each value of lambda.

tau: Quantile of the model.

a: Value of a for the penalized loss function.

If the Huber algorithm is used than $\rho_\tau(y_i - x_i^\top \beta)$ is replaced by a Huber-type approximation. Specifically, it is replaced by $h_\gamma^\tau(y_i - x_i^\top \beta)/2$ where

$$h_\gamma^\tau(a) = a^2/(2\gamma)I(|a| \leq \gamma) + (|a| - \gamma/2)I(|a| > \gamma) + (2\tau - 1)a.$$

Where if $\tau = .5$, we get the usual Huber loss function. The Huber implementation calls the package hqreg which implements the methods of Yi and Huang (2017) for Huber loss with elastic net penalties. For non-elastic net penalties the LLA algorithm of Zou and Li (2008) is used to approximate those loss functions with a lasso penalty with different weights for each predictor.

Author(s)

Ben Sherwood, <ben.sherwood@ku.edu>, Shaobo Li, and Adam Maidman

References

Zou H, Li R (2008). "One-step sparse estimates in nonconcave penalized likelihood models." *Ann. Statist.*, **36**(4), 1509-1533.

Yi C, Huang J (2017). "Semismooth Newton Coordinate Descent Algorithm for Elastic-Net Penalized Huber Loss Regression and Quantile Regression." *J. Comput. Graph. Statist.*, **26**(3), 547-557.

Belloni A, Chernozhukov V (2011). "L1-Penalized quantile regression in high-dimensional sparse models." *Ann. Statist.*, **39**(1), 82-130.

Peng B, Wang L (2015). "An iterative coordinate descent algorithm for high-dimensional nonconvex penalized quantile regression." *J. Comput. Graph. Statist.*, **24**(3), 676-694.

Examples

```
n <- 200
p <- 8
x <- matrix(runif(n*p),ncol=p)
y <- 1 + x[,1] + x[,8] + (1+.5*x[,3])*rnorm(n)
r1 <- rq.pen(x,y) #Lasso fit for median
# Lasso for multiple quantiles
r2 <- rq.pen(x,y,tau=c(.25,.5,.75))
# Elastic net fit for multiple quantiles, which must use Huber algorithm
r3 <- rq.pen(x,y,penalty="ENet",a=c(0,.5,1),alg="huber")
# First variable is not penalized
r4 <- rq.pen(x,y,penalty.factor=c(0,rep(1,7)))
tvals <- c(.1,.2,.3,.4,.5)
#Similar to penalty proposed by Belloni and Chernouzhukov.
#To be exact you would divide the tau.penalty.factor by n.
r5 <- rq.pen(x,y,tau=tvals, tau.penalty.factor=sqrt(tvals*(1-tvals)))
```

---

rq.pen.cv *Does k-folds cross validation for rq.pen. If multiple values of a are specified then does a grid based search for best value of λ and a.*

Description

Does k-folds cross validation for rq.pen. If multiple values of a are specified then does a grid based search for best value of λ and a.

Usage

```
rq.pen.cv(
  x,
  y,
  tau = 0.5,
```

```
  lambda = NULL,
  penalty = c("LASSO", "Ridge", "ENet", "aLASSO", "SCAD", "MCP"),
  a = NULL,
  cvFunc = NULL,
  nfolds = 10,
  foldid = NULL,
  nlambda = 100,
  groupError = TRUE,
  cvSummary = mean,
  tauWeights = rep(1, length(tau)),
  printProgress = FALSE,
  weights = NULL,
  ...
)
```

Arguments

x	Matrix of predictors.
y	Vector of responses.
tau	Quantiles to be modeled.
lambda	Values of λ. Default will automatically select the λ values.
penalty	Choice of penalty between LASSO, Ridge, Elastic Net (ENet), Adaptive Lasso (aLASSO), SCAD and MCP.
a	Tuning parameter of a. LASSO and Ridge has no second tuning parameter, but for notation is set to 1 or 0 respectively, the values for elastic net. Defaults are Ridge ()
cvFunc	Loss function for cross-validation. Defaults to quantile loss, but user can specify their own function.
nfolds	Number of folds.
foldid	Ids for folds. If set will override nfolds.
nlambda	Number of lambda, ignored if lambda is set.
groupError	If set to false then reported error is the sum of all errors, not the sum of error for each fold.
cvSummary	Function to summarize the errors across the folds, default is mean. User can specify another function, such as median.
tauWeights	Weights for the different tau models. Only used in group tau results (gtr).
printProgress	If set to TRUE prints which partition is being worked on.
weights	Weights for the quantile loss objective function.
...	Additional arguments passed to rq.pen()

Details

Two cross validation results are returned. One that considers the best combination of a and lambda for each quantile. The second considers the best combination of the tuning parameters for all

quantiles. Let $y_{b,i}$, $x_{b,i}$, and $m_{b,i}$ index the response, predictors, and weights of observations in fold b. Let $\hat{\beta}^{-b}_{\tau,a,\lambda}$ be the estimator for a given quantile and tuning parameters that did not use the bth fold. Let n_b be the number of observations in fold b. Then the cross validation error for fold b is

$$\text{CV}(b,\tau) = \frac{1}{n_b}\sum_{i=1}^{n_b} m_{b,i}\rho_\tau(y_{b,i} - x_{b,i}^\top \hat{\beta}^{-b}_{\tau,a,\lambda}).$$

Note that $\rho_\tau()$ can be replaced by a different function by setting the cvFunc parameter. The function returns two different cross-validation summaries. The first is btr, by tau results. It provides the values of `lambda` and a that minimize the average, or whatever function is used for `cvSummary`, of $\text{CV}(b)$. In addition it provides the sparsest solution that is within one standard error of the minimum results.

The other approach is the group tau results, gtr. Consider the case of estimating Q quantiles of $\tau_1, \ldots, \tau_Q$ with quantile (tauWeights) of v_q. The gtr returns the values of `lambda` and a that minimizes the average, or again whatever function is used for `cvSummary`, of

$$\sum_{q=1}^{Q} v_q \text{CV}(b,\tau_q).$$

If only one quantile is modeled then the gtr results can be ignored as they provide the same minimum solution as btr.

Value

An rq.pen.seq.cv object.

cverr: Matrix of cvSummary function, default is average, cross-validation error for each model, tau and a combination, and lambda.

cvse: Matrix of the standard error of cverr foreach model, tau and a combination, and lambda.

fit: The rq.pen.seq object fit to the full data.

btr: A data.table of the values of a and lambda that are best as determined by the minimum cross validation error and the one standard error rule, which fixes a. In btr the values of lambda and a are selected seperately for each quantile.

gtr: A data.table for the combination of a and lambda that minimize the cross validation error across all tau.

gcve: Group, across all quantiles, cross-validation error results for each value of a and lambda.

call: Original call to the function.

Author(s)

Ben Sherwood, <ben.sherwood@ku.edu>

Examples

```
## Not run:
x <- matrix(runif(800),ncol=8)
y <- 1 + x[,1] + x[,8] + (1+.5*x[,3])*rnorm(100)
```

```
r1 <- rq.pen.cv(x,y) #lasso fit for median
# Elastic net fit for multiple values of a and tau
r2 <- rq.pen.cv(x,y,penalty="ENet",a=c(0,.5,1),tau=c(.25,.5,.75))
#same as above but more weight given to median when calculating group cross validation error.
r3 <- rq.pen.cv(x,y,penalty="ENet",a=c(0,.5,1),tau=c(.25,.5,.75),tauWeights=c(.25,.5,.25))
# uses median cross-validation error instead of mean.
r4 <- rq.pen.cv(x,y,cvSummary=median)
#Cross-validation with no penalty on the first variable.
r5 <- rq.pen.cv(x,y,penalty.factor=c(0,rep(1,7)))

## End(Not run)
```

---

rqPen *rqPen: A package for estimating quantile regression models using penalized objective functions.*

Description

The package estimates a quantile regression model using LASSO, Adaptive LASSO, SCAD, MCP, elastic net, and their group counterparts, with the exception of elastic net for which there is no group penalty implementation.

rqPen functions

The most important functions are rq.pen(), rq.group.pen(), rq.pen.cv() and rq.group.pen.cv(). These functions fit quantile regression models with individual or group penalties. The cv functions automate the cross-validation process for selection of tuning parameters.

Index