Package ‘rqPen’

May 1, 2019

Type Package
Title Penalized Quantile Regression
Version 2.1
Date 2019-5-1
Author Ben Sherwood [aut, cre], Adam Maidman [ctb]
Depends R (>= 3.0.0), quantreg, regpro
Suggests splines
Maintainer Ben Sherwood <ben.sherwood@ku.edu>
Description Performs penalized quantile regression for LASSO, SCAD and MCP functions including group penalties. Provides a function that automatically generates lambdas and evaluates different models with cross validation or BIC, including a large p version of BIC.
ByteCompile TRUE
License MIT + file LICENSE
NeedsCompilation yes
Repository CRAN
Date/Publication 2019-05-01 21:20:03 UTC

R topics documented:

rqPen-package .......................................................... 2
beta_plots ............................................................... 3
check ................................................................. 4
coef.cv rq.group.pen ............................................... 5
coef.cv rq.pen ....................................................... 6
cv rq.group.pen .................................................... 6
cv rq.pen ............................................................ 8
cv_plots ............................................................. 10
getRho ............................................................... 11
groupMultLambda .................................................. 11
group_derivs ......................................................... 13
kernel_estimates ..................................................... 13
## Index

| rqPen-package | Penalized quantile regression for LASSO, SCAD, and MCP penalty functions including group penalties |

### Description

This package provides functions to find solutions to penalized quantile regression problems. Throughout this package, the estimated coefficients are the minimizers of the penalized quantile regression objective function:

\[
\beta = \frac{1}{n} \sum_{i=1}^{n} \rho_{\tau}(y_i - x_i^T \beta) + \sum_{j=1}^{p} p_{\lambda}(|\beta_j|)
\]

where

\[
\rho_{\tau}(u) = u(\tau - I(u < 0))
\]
This package can handle three different penalty functions with $\lambda > 0$:

**LASSO:**
\[ p_\lambda(|\beta_j|) = \lambda |\beta_j| \]

**SCAD:**
\[ p_\lambda(|\beta_j|) = \lambda |\beta_j| I(0 \leq |\beta_j| < \lambda) + \frac{a\lambda |\beta_j| - (\beta_j^2 + \lambda^2)/2}{a - 1} I(\lambda \leq |\beta_j| \leq a\lambda) + \frac{(a + 1)\lambda^2}{2} I(|\beta_j| > a\lambda), \]
for $a > 2$

**MCP:**
\[ p_\lambda(|\beta_j|) = \lambda (|\beta_j| - \frac{\beta_j^2}{2a\lambda}) I(0 \leq |\beta_j| \leq a\lambda) + \frac{a\lambda^2}{2} I(|\beta_j| > a\lambda), \]
for $a > 1$.

---

**beta_plots**

*Plots of Betas*

**Description**

Plots how the beta estimates changes with the different values of lambda.

**Usage**

\[ \text{beta} \_ \text{plots}(\text{model}, \text{voi}=\text{NULL}, \text{logLambda}=\text{TRUE}, \text{loi}=\text{NULL}, \ldots) \]

**Arguments**

- **model**: "cv.rq.pen" object.
- **voi**: Index of betas to include. Default is all of the lambdas from "cv.rq.pen" object.
- **logLambda**: Plot of lambdas is on the log scale.
- **loi**: Index of lambdas to include. Default is all of the lambdas from "cv.rq.pen" object.
- **...**: Additional arguments to be sent to plot.

**Value**

Plot of how beta estimates change with lambda.

**Author(s)**

Ben Sherwood

**Examples**

\[ x \leftarrow \text{matrix(rnorm(800), nrow=100)} \]
\[ y \leftarrow 1 + x[,1] - 3 \times x[,5] + \text{rnorm(100)} \]
\[ \text{lassoModels} \leftarrow \text{cv.rq.pen}(x, y) \]
\[ \text{b} \_ \text{plot} \leftarrow \text{beta} \_ \text{plots}(\text{lassoModels}) \]
Quantile check function

Description

Evaluates the check function for quantile \( \tau \) at value \( x \). Check function is the objective function defined in Koenker and Bassett (1978).

Usage

\[ \text{check}(x, \tau) \]

Arguments

- \( x \) Number to be evaluated.
- \( \tau \) Number between 0 and 1 for quantile of interest.

Value

\[ x \times (\tau - \mathbb{1}(x < 0)) \]

Author(s)

Ben Sherwood

References


Examples

- \text{check}(2, .5)
- \text{check}(-2, .5)
- \text{check}(2, .2)
- \text{check}(2, .8)
coef.cv.rq.group.pen  Group Penalized Quantile Regression Coefficients

Description

Returns coefficients for a cv.rq.pen object with default values being coefficients associated with minimum cross-validation value.

Usage

## S3 method for class 'cv.rq.group.pen'
coef(object, lambda='min',...)

Arguments

object  cv.rq.group.pen object

lambda  Tuning parameter lambda. Default is to select minimum lambda from cross-validation method. User can also select a specific value of lambda, but it needs to be a lambda that was part of the fit of cv.rq.pen object.

...  Additional arguments, but currently not used.

Value

Coefficients for selected value of lambda.

Author(s)

Ben Sherwood

Examples

## Not run:
x <- matrix(rnorm(400),nrow=100)
y <- 1 + x[,1] - 3*x[,3] + rnorm(100)
cv_model <- cv.rq.group.pen(x,y,groups=c(rep(1,2),rep(2,2)))
coefficients(cv_model)

## End(Not run)
coef.cv.rq.pen  

*Penalized Quantile Regression Coefficients*

**Description**

Returns coefficients for a cv.rq.pen object with default values being coefficients associated with minimum cross-validation value.

**Usage**

```r
## S3 method for class 'cv.rq.pen'
coef(object, lambda='min',...)
```

**Arguments**

- `object`: cv.rq.pen object
- `lambda`: Tuning parameter lambda. Default is to select minimum lambda from cross-validation method. User can also select a specific value of lambda, but it needs to be a lambda that was part of the fit of cv.rq.pen object.
- `...`: Additional arguments, currently not used.

**Value**

Coefficients for selected value of lambda.

**Author(s)**

Ben Sherwood

**Examples**

```r
x <- matrix(rnorm(100),nrow=20)
y <- 1 + x[,1] - 3*x[,5] + rnorm(20)
cv_model <- cv.rq.pen(x,y)
coefficients(cv_model)
```

---

**cv.rq.group.pen**

*Cross Validated quantile regression with group penalty*

**Description**

Similar to cv.rq.pen function, but uses group penalty. Group penalties use the L1 norm instead of L2 for computational convenience. QICD is a group penalty extension of the algorithm presented by Peng and Wang (2015). LP does a linear programming version of the group penalty.
**Usage**

cv.rq.group.pen(x, y, groups, tau = 0.5, lambda = NULL, penalty = "LASSO",
                  intercept = TRUE, criteria = "CV", cvFunc = "check", nfolds = 10,
                  foldid = NULL, nlambda = 100, eps = 1e-04, init.lambda = 1,alg="QICD",
                  penGroups=NULL, ...)

**Arguments**

- **x** Matrix of predictors.
- **y** Vector of response values.
- **groups** Vector assigning columns of x to groups.
- **tau** Conditional quantile being modelled.
- **lambda** Vector of lambdas. Default is for lambdas to be automatically generated.
- **penalty** Type of penalty: "LASSO", "SCAD" or "MCP".
- **intercept** Whether model should include an intercept. Constant does not need to be included in "x".
- **criteria** How models will be evaluated. Either cross-validation "CV", BIC "BIC" or large P BIC "PBIC".
- **cvFunc** If cross-validation is used how errors are evaluated. Check function "check", "SqErr" (Squared Error) or "AE" (Absolute Value).
- **nfolds** K for K-folds cross-validation.
- **foldid** Group id for cross-validation. Function will randomly generate groups if not specified.
- **nlambda** Number of lambdas for which models are fit.
- **eps** Smallest lambda used.
- **init.lambda** Initial lambda used to find the maximum lambda. Not needed if lambda values are set.
- **alg** Algorithm used for fit. "QICD" or "LP".
- **penGroups** Specify which groups will be penalized. Default is to penalize all groups.
- **...** Additional arguments to be sent to rq.group.fit or groupQICDMultLambda.

**Value**

Returns the following:

- **beta** Matrix of coefficients for different values of lambda
- **residuals** Matrix of residuals for different values of lambda.
- **rho** Vector of rho, unpenalized portion of the objective function, for different values of lambda.
- **cv** Data frame with "lambda" and second column is the evaluation based on the criteria selected.
- **lambda.min** Lambda which provides the smallest statistic for the selected criteria.
- **penalty** Penalty selected.
- **intercept** Whether intercept was included in model.
- **groups** Group structure for penalty function.
Author(s)

Ben Sherwood

References


Examples

```r
## Not run:
x <- matrix(rnorm(800), nrow=100)
y <- 1 + x[,1] - 3*x[,5] + rnorm(100)
cv_model <- cv.rq.group.pen(x,y,groups=c(rep(1L,rep(4L)))

## End(Not run)
```

---

cv.rq.pen  Cross Validated quantile regression

Description

Produces penalized quantile regression models for a range of lambdas and penalty of choice. If lambda is unselected than an iterative algorithm is used to find a maximum lambda such that the penalty is large enough to produce an intercept only model. Then range of lambdas goes from the maximum lambda found to "eps" on the log scale. For non-convex penalties local linear approximation approach used by Wang, Wu and Li to extend LLA as proposed by Zou and Li (2008) to the quantile regression setting.

Usage

```
cv.rq.pen(x,y,tau=.5,lambda=NULL,weights=NULL,penalty="LASSO",
           intercept=TRUE,criteria="CV",cvFunc="check",nfolds=10,
           foldid=NULL,nlambda=100,eps=.0001,init.lambda=1,penVars=NULL,
           alg = ifelse(ncol(x) < 50, "LP", "QICD"),...)
```

Arguments

- **x**: Matrix of predictors.
- **y**: Vector of response values.
- **tau**: Conditional quantile being modelled.
- **lambda**: Vector of lambdas. Default is for lambdas to be automatically generated.
weights Weights for the objective function.
penalty Type of penalty: "LASSO", "SCAD" or "MCP".
intercept Whether model should include an intercept. Constant does not need to be included in "x".
criteria How models will be evaluated. Either cross-validation "CV", BIC "BIC" or large P BIC "PBIC".
cvFunc If cross-validation is used how errors are evaluated. Check function "check", "SqErr" (Squared Error) or "AE" (Absolute Value).
nfolds K for K-folds cross-validation.
foldid Group id for cross-validation. Function will randomly generate groups if not specified.
nlambda Number of lambdas for which models are fit.
eps Smallest lambda used.
init.lambda Initial lambda used to find the maximum lambda. Not needed if lambda values are set.
penVars Variables that should be penalized. With default value of NULL all variables are penalized.
alg Algorithm that will be used, either linear programming (LP) or coordinate descent (QICD) algorithm from Peng and Wang (2015).
... Additional arguments to be sent to rq.lasso.fit or rq.nc.fit.

Value

Returns the following:
models List of penalized models fit. Number of models will match number of lambdas and correspond to cv$lambda.
cv Data frame with "lambda" and second column is the evaluation based on the criteria selected.
lambda.min Lambda which provides the smallest statistic for the selected criteria.
penalty Penalty selected.

Author(s)

Ben Sherwood

References

cv_plots

Plots of Cross-validation results

Description

Slightly misnamed as user could choose BIC as a criteria for "cv.rq.pen" object. Function is able to discern between the two types of evaluation criteria and provides appropriate labels for the plot.

Usage

cv_plots(model, logLambda=TRUE, loi=NULL,...)

Arguments

model "cv.rq.pen" object.
logLambda Plot of lambdas is on the log scale.
loi Index of lambdas to be plotted. Default is all of the lambdas from "cv.rq.pen" object.
... Additional items to be sent to plot function.

Value

Plot of how cross validation statistic changes with lambda.

Author(s)

Ben Sherwood

Examples

```r
x <- matrix(rnorm(100), nrow=20)
y <- 1 + x[,1] - 3*x[,5] + rnorm(20)
lassoModels <- cv.rq.pen(x, y)
cv_plot <- cv_plots(lassoModels)
```
getRho

**Objective Function Value**

**Description**

Returns unpenalized portion of the objective function for a penalized quantile regression object.

**Usage**

getRho(model)

**Arguments**

- **model**
  - Object with rho as an attribute

**Value**

Rho is the value of the unpenalized portion of the objective function.

**Author(s)**

Ben Sherwood

**Examples**

```r
x <- matrix(rnorm(800),nrow=100)
y <- 1 + x[,1] - 3*x[,5] + rnorm(100)
lassoModel <- rq.lasso.fit(x,y,lambda=.5)
getRho(lassoModel)
```

---

**groupMultLambda**

**Quantile Regression with Group Penalty for multiple lambdas**

**Description**

Fit multiple models with L1 group penalty. QICD algorithm is using an adaptation of the algorithm presented by Peng and Wang (2015).

**Usage**

```r
groupMultLambda(x, y, groups, tau = 0.5, lambda, intercept = TRUE, penalty="LASSO", alg="QICD_warm", penGroups=NULL, ...)
```
Arguments

- `x` Matrix of predictors.
- `y` Vector of response values.
- `groups` Vector assigning columns of `x` to groups.
- `tau` Conditional quantile being modelled.
- `lambda` Vector of lambdas. Default is for lambdas to be automatically generated.
- `intercept` Whether model should include an intercept. Constant does not need to be included in "x".
- `penalty` Type of penalty: "LASSO", "SCAD" or "MCP".
- `alg` "QICD" for QICD implementation. Otherwise linear programming approach is implemented.
- `penGroups` Specify which groups will be penalized. Default is to penalize all groups.
- `...` Additional parameters to be sent to rq.group.fit.

Value

Returns a list of rq.group.pen objects. Each element of the list is a fit for a different value of lambda.

Author(s)

Ben Sherwood

References


Examples

```r
## Not run:
x <- matrix(rnorm(400),nrow=100)
y <- 1 + x[,1] - 3*x[,3] + rnorm(100)
cv_model <- groupMultLambda(x,y,groups=c(rep(1,2),rep(2,2)),lambda=seq(.1,.5,))
## End(Not run)
```
**group_derivs**

*Derivative of a group penalty*

**Description**

Used to estimate non-convex group penalties.

**Usage**

```r
group_derivs(deriv_func, groups, coefs, lambda, a=3.7)
```

**Arguments**

- `deriv_func`: Target derivative function.
- `groups`: Vector assigning columns of x to groups.
- `coefs`: Coefficients.
- `lambda`: Lambda value for deriv_func.
- `a`: Additional parameter for deriv_func.

**Value**

Returns the derivative of the L1 group penalty function.

**Author(s)**

Ben Sherwood

---

**kernel_estimates**

*Kernel based estimates of Y|X*

**Description**

Provides fitted values of Y.

**Usage**

```r
kernel_estimates(x, y, h, ...)
```

**Arguments**

- `x`: matrix of predictors
- `y`: Vector of response
- `h`: Scalar bandwidth tuning parameter
- `...`: Additional arguments to be sent to kernesti.regr from regrepo package.
**Value**

Estimates a conditional density. For future use of implementing inverse probability weights (IPW) to handle missing data.

**Author(s)**

Ben Sherwood

**Examples**

```r
x <- matrix(rnorm(800), nrow=100)
y <- rbinom(100, 1, .5)
cond_fit <- kernel_estimates(x, y, 1)
```

<table>
<thead>
<tr>
<th>kernel_weights</th>
<th>Nonparametric estimate of IPW weights</th>
</tr>
</thead>
</table>

**Description**

This is for downstream analysis for fitting models with missing data. Future work is to fully incorporate these into penalized models. Tuning parameter for conditional density is estimated using approach of Chen, Wan and Zhou (2015), which is a simplified approach of Sepanski et al. (1994).

**Usage**

```r
kernel_weights(obs_data, obs_ind, ...)
```

**Arguments**

- `obs_data` Matrix of variables with complete observations
- `obs_ind` Vector of whether sample is observed or not (1-observed, 0-not)
- `...` Additional arguments to be sent to `kernel_estimates`.

**Value**

Estimates of weights.

**Author(s)**

Ben Sherwood

**References**

Examples

```r
x <- matrix(rnorm(800), nrow=100)
y <- rbinom(100, 1, .5)
wts <- kernel_weights(x, y)
```

---

lasso  

Lasso

Description

LASSO penalty function.

Usage

```r
lasso(x, lambda)
```

Arguments

- `x`: Number to be evaluated
- `lambda`: Tuning parameter lambda

Value

`lambda*abs(x)`

Author(s)

Ben Sherwood

References


Examples

```r
lasso(3, 1)
lasso(-3, 1)
lasso(-3, 2)
```
LASSO.fit

LASSO Penalized Quantile Regression

Description

LASSO.fit obtains coefficient estimates for Lasso penalized quantile regression. It is called by the QICD and QICD.group functions to obtain initial estimates when they are not provided.

Usage

LASSO.fit(y, x, tau, lambda, intercept, coef.cutoff, weights=NULL)

Arguments

- **y**: Vector of responses.
- **x**: n x p matrix of covariates.
- **tau**: Conditional quantile being modelled.
- **lambda**: Tuning parameter. Must be positive.
- **intercept**: If TRUE, an intercept is included in the model. If FALSE, no intercept is included.
- **coef.cutoff**: Coefficients with magnitude less than this value are set to 0.
- **weights**: If not NULL, weights must be a vector of length n with a positive weight for each observation. This is used for the linear programming solution for the SCAD and MCP penalties.

Details

This is a barebones function that only provides coefficient estimates. It will not provide any warnings or errors, so you need to check that inputs are accurate and appropriate. The rq.lasso.fit function should be used to obtain more information from the Lasso fit.

Value

Returns a vector containing the intercept (if intercept=TRUE) and the estimated coefficients for each column in x.

Author(s)

Adam Maidman

References

Examples

```r
n = 50
p = 100
x <- matrix(rnorm(n*p), nrow=n)
y <- 0 + x[,1] - 3*x[,5] + rnorm(n)
fit1 <- LASSO.fit(y, x, tau=.5, lambda=1, intercept=TRUE, coef.cutoff=1e-08)
fit2 <- LASSO.fit(y, x, tau=.5, lambda=.1, intercept=TRUE, coef.cutoff=1e-08)
```

**Description**

LASSO.fit.nonpen obtains coefficient estimates for Lasso penalized quantile regression with some nonpenalized coefficients. It is called by the QICD.nonpen function to obtain initial estimates when they are not provided.

**Usage**

```r
LASSO.fit.nonpen(y, x, z, tau, lambda, intercept, coef.cutoff, weights=NULL)
```

**Arguments**

- `y` Vector of responses.
- `x` $n \times p$ matrix of covariates.
- `z` $n \times q$ matrix of observations with each row corresponding to one observation. Penalties (and variable selection) WILL NOT be applied to these coefficients. Do not include column of 1’s; set intercept=TRUE, if intercept is desired.
- `tau` Conditional quantile being modelled.
- `lambda` Tuning parameter. Must be positive.
- `intercept` If TRUE, an intercept is included in the model. If FALSE, no intercept is included.
- `coef.cutoff` Coefficients with magnitude less than this value are set to 0.
- `weights` If not NULL, weights must be a vector of length $n$ with a positive weight for each observation. This is used for the linear programming solution for the SCAD and MCP penalties.

**Details**

This is a barebones function that only provides coefficient estimates. It will not provide any warnings or errors, so you need to check that inputs are accurate and appropriate. The rq.lasso.fit function should be used to obtain more information from the Lasso fit.
Value

Returns a vector containing the intercept (if intercept=TRUE) and the estimated coefficients for each column in x and z.

Author(s)

Adam Maidman

References


Examples

```r
library(splines)
n = 50
p = 100
x <- matrix(rnorm(n*p),nrow=n)
z1 <- runif(n)
z <- bs(z1)
y <- 0 + x[,1] - 3*x[,5] + z1^3 + rnorm(n)
fit1 <- LASSO.fit.nonpen(y,x,z, tau=.5, lambda=1, intercept=TRUE, coef.cutoff=1e-08)
fit2 <- LASSO.fit.nonpen(y,x,z, tau=.5, lambda=.1, intercept=TRUE, coef.cutoff=1e-08)
```

Description

MCP function as described in Fan and Li (2001).

Usage

```r
mcp(x, lambda, a)
```

Arguments

- `x` Number to be evaluated
- `lambda` Tuning parameter lambda
- `a` Tuning parameter a

Value

MCP function with tuning parameters lambda and "a" evaluated at "x".
**mcp_deriv**

**Author(s)**

Ben Sherwood

**References**


**Examples**

```r
mcp(3,1)
mcp(-3,1)
mcp(.001,2)
```

---

**Description**

Derivative of MCP function as described in Fan and Li (2001).

**Usage**

```r
mcp_deriv(x, lambda, a)
```

**Arguments**

- `x` Number to be evaluated
- `lambda` Tuning parameter lambda
- `a` Tuning parameter a

**Value**

Derivative of MCP function with tuning parameters lambda and "a" evaluated at "x".

**Author(s)**

Ben Sherwood

**References**


**Examples**

```r
mcp(3,1)
mcp(-3,1)
mcp(.001,2)
```
model_eval

Model Evaluation

Description

Used for cross-validation. For a model of class "rqPen" it provides the average prediction error given the evaluation function of choice.

Usage

model_eval(model, test_x, test_y, test_w=NULL, func="check", ...)

Arguments

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".

model  Model of class "rqPen".
nonzero

nonzero  Nonzero

Description

Calls nonzero function based on object.

Usage

nonzero(obj)

Arguments

obj  Model.

Value

Returns if coefficients or groups are nonzero or not. TRUE if they are not zero and FALSE if they are.

Author(s)

Ben Sherwood

Examples

## Not run:
x <- matrix(rnorm(400),nrow=100)
y <- 1 + x[,1] - 3*x[,3] + rnorm(100)
cv_model <- cv.rq.group.pen(x,y,groups=c(rep(1,2),rep(2,2)))
nonzero(cv_model)

## End(Not run)

nonzero.cv.rq.group.pen

nonzero.cv.rq.group.pen  Nonzero

Description

Calls nonzero function.

Usage

## S3 method for class 'cv.rq.group.pen'
nonzero(obj)
Arguments

obj  cv.rq.group.pen object.

Value

Returns true if all elements in a group are non-zero and FALSE if they are not. Chooses the model associated with lambda.min.

Author(s)

Ben Sherwood

Examples

## Not run:
x <- matrix(rnorm(400),nrow=100)
y <- 1 + x[,1] - 3*x[,3] + rnorm(100)
cv_model <- cv.rq.group.pen(x,y,groups=c(rep(1,2),rep(2,2)))
nonzero(cv_model)

## End(Not run)

plot.cv.rq.group.pen  Plot cv.rq.group.pen

Description

Plots the validation criteria against the lambda values.

Usage

## S3 method for class 'cv.rq.group.pen'
plot(x,y=NULL,...)

Arguments

x  cv.rq.group.pen object.
y  holder value to match up with default plot program
...  Additional values to function, but not currently used

Value

Plots the validation criteria against the lambda values.

Author(s)

Ben Sherwood
### Examples

```r
## Not run:
x <- matrix(rnorm(400), nrow=100)
y <- 1 + x[,1] - 3*x[,3] + rnorm(100)
cv_model <- cv.rq.group.pen(x, y, groups=c(rep(1,2), rep(2,2)))
plot(cv_model)

## End(Not run)
```

---

<table>
<thead>
<tr>
<th>pos_part</th>
<th>Positive part</th>
</tr>
</thead>
</table>

### Description

Returns \( \min(0, x) \)

### Usage

`pos_part(x)`

### Arguments

- `x`: Number to be evaluated

### Value

\( \min(0, x) \)

### Author(s)

Ben Sherwood

### Examples

```r
pos_part(5)
pos_part(-5)
```
predict.cv.rq.pen  Prediction from a cv quantile regression penalized model

Description

Returns predicted values from "rqPen" model associated with lambda for "newx" covariates.

Usage

## S3 method for class 'cv.rq.pen'
predict(object, newx, lambda,...)

Arguments

- **object**: "cv.rq.pen" object.
- **newx**: Matrix of covariates used for prediction.
- **lambda**: Lambda associated with the model from which predictions should be made. Default is to use the lambda that provides the minimum criteria (cross-validation or BIC) that was selected by cv.rq.pen.
- **...**: Needed for consistency with generic predict.

Value

Returns predicted values from the model for the selected lambda.

Author(s)

Ben Sherwood

Examples

```r
x <- matrix(rnorm(800),nrow=100)
y <- 1 + x[,1] - 3*x[,5] + rnorm(100)
cv_model <- cv.rq.pen(x,y)
n2 <- matrix(rnorm(80),nrow=10)
preds <- predict(cv_model,n2)
```
### predict.rq.pen

**Prediction from a quantile regression penalized model**

**Description**

Returns predicted values from "rq.pen" object for "newx" covariates.

**Usage**

```r
## S3 method for class 'rq.pen'
predict(object, newx, ...) 
```

**Arguments**

- `object`: rq.pen object.
- `newx`: Matrix of covariates used for prediction.
- `...`: Needed for consistency with generic predict.

**Value**

Returns predicted values from the model.

**Author(s)**

Ben Sherwood

**Examples**

```r
x <- matrix(rnorm(800), nrow=100)
y <- 1 + x[,1] - 3*x[,5] + rnorm(100)
lassoModel <- rq.lasso.fit(x, y, lambda=1)
n2 <- matrix(rnorm(80), nrow=10)
preds <- predict(lassoModel, n2)
```

### print.cv.rq.pen

**Print cv.rq.pen object**

**Description**

Prints the cross validation (or bic) results and reports coefficients for a cv.rq.pen object. Default coefficients are for value that minimizes cross validation or BIC.

**Usage**

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

x  Object to be printed.
... Additional arguments for coefficient function

Author(s)

Ben Sherwood

print.rq.pen  Print rq.pen object

Description

Reports coefficients for a rq.pen object.

Usage

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

Arguments

x  Object to be printed.
... Additional arguments for coefficient function

Author(s)

Ben Sherwood

qaSIS  Quantile Adaptive Sure Independence Screening

Description

Implements quantile adaptive screening as outlined by He, Wang and Hong

Usage

```r
qaSIS(x,y,tau,linear=FALSE,...)
```
**Arguments**

- `x`  Matrix of predictors.
- `y` Vector of response values.
- `tau` Conditional quantile being modelled.
- `linear` If true linear screening will be done, otherwise splines will be fit to each column vector.
- `...` Additional items to be sent to bs function from splines package.

**Value**

Returns the ranking of important predictors, from highest to lowest.

**Author(s)**

Ben Sherwood

**References**


**Examples**

```r
x <- pnorm(matrix(rnorm(8000), nrow=100))
y <- 1 + 2*cos(2*pi*x[,1])+exp(2*x[,2]) + rnorm(100, sd=.1)
var_ranks <- qasis(x,y)
```

**Description**

Quantile regression BIC with large p alternative as described in Lee, Noh and Park (2013).

**Usage**

`qbic(model, largeP=FALSE)`

**Arguments**

- `model` Model of class "rqPen".
- `largeP` Large P version using an additional penalty factor of log(s) where "s" is the total number of covariates considered.

**Value**

Numeric value representing BIC of selected model.
Author(s)

Ben Sherwood

References


Examples

```r
x <- matrix(rnorm(800), nrow=100)
y <- 1 + x[,1] - 3*x[,5] + rnorm(100)
l_model <- rq.lasso.fit(x, y, lambda=1)
nc_model <- rq.nc.fit(x, y, lambda=1)
qbic(l_model)
qbic(nc_model)
qbic(l_model, largeP=TRUE)
qbic(nc_model, largeP=TRUE)
```

---

**QICD**

*Penalized Quantile Regression with QICD Algorithm*

**Description**

QICD produces penalized quantile regression estimates using the QICD algorithm. If no initial values are given, LASSO estimates will be used. This function can handle the LASSO, SCAD, and MCP penalties.

**Usage**

```r
QICD(y, x, tau=.5, lambda, intercept=TRUE, penalty="SCAD",
    initial_beta=NULL, maxin=100, maxout=20, eps = 1e-05, coef.cutoff=1e-08,
    a=3.7, ...)
```

**Arguments**

- `y` Vector of response values.
- `x` $n \times p$ matrix of observations with each row corresponding to one observation. Do not include column of 1's; set intercept=TRUE, if intercept is desired.
- `tau` Conditional quantile being modelled.
- `lambda` Tuning parameter for LASSO, SCAD, and MCP penalties. Must be positive.
- `intercept` If TRUE, an intercept is included in the model. If FALSE, no intercept is included.
- `penalty` Penalty function for the coefficients. Either "SCAD", "MCP", or "LASSO". See details for description of penalties.
**QICD**

**initial_beta**  Vector of initial values for QICD algorithm. The vector should contain the intercept first (if intercept=TRUE) and then the p coefficients. If NULL, exact LASSO estimates will be computed and used as initial values.

**maxin**  Maximum number of iterations on the minimization step of the QICD algorithm.

**maxout**  Maximum number of iterations on the majorization step of the QICD algorithm.

**eps**  Threshold for convergence of algorithm.

**coef.cutoff**  Coefficients with magnitude less than this value are set to 0.

**a**  Second tuning parameter for SCAD and MCP penalties. Must be greater than 2 for SCAD and greater than 1 for MCP.

**...**  The extra arguments will not be used.

**Details**

The QICD algorithm should only be used for the LASSO penalty if initial_beta can be set to LASSO estimates with a similar lambda (similar to a "warm start"). Otherwise, exact LASSO estimates will be used as initial values for the QICD algorithm: this will cause unnecessary computations and could lead to less accurate estimates.

**Value**

Returns a vector containing the intercept (if intercept=TRUE) and the estimated coefficients for each column in x.

**Author(s)**

Adam Maidman

**References**


**Examples**

```r
n = 50
p = 5
x <- matrix(rnorm(n*p),nrow=n)
y <- 0 + x[,1] - 3*x[,5] + rnorm(n)
fit1 <- QICD(y,x, tau=.5, lambda=1, intercept=TRUE, penalty="SCAD")
fit2 <- QICD(y,x, tau=.7, lambda=1, intercept=TRUE, penalty="SCAD")
```
Description

QICD.group produces group penalized quantile regression estimates using the QICD algorithm. If no initial values are given, LASSO estimates will be used. This function can handle the LASSO, SCAD, and MCP penalties.

Usage

QICD.group(y, x, groups, tau=.5, lambda, intercept=TRUE, penalty="SCAD",
            initial_beta=NULL, maxin=100, maxout=20, eps = 1e-05,
            coef.cutoff=1e-08, a=3.7, ...)  

Arguments

y  Vector of response values.

x  n x p matrix of observations with each row corresponding to one observation. 
   Do not include column of 1's; set intercept=TRUE, if intercept is desired.

groups  Vector of length p with the group number of the corresponding coefficient. Co-
         efficients in the same group will either all be 0, or none will be 0.

tau  Conditional quantile being modelled.

lambda  Tuning parameter for LASSO, SCAD, and MCP penalties. Must be positive.

intercept  If TRUE, an intercept is included in the model. If FALSE, no intercept is in-
            cluded.

penalty  Penalty function for the coefficients. Either "SCAD", "MCP", or "LASSO". See 
         details for description of penalties.

initial_beta  Vector of initial values for QICD algorithm. The vector should contain the in-
               tercept first (if intercept=TRUE) and then the p coefficients. If NULL, exact 
               LASSO estimates will be computed and used as initial values.

maxin  Maximum number of iterations on the minimization step of the QICD algorithm.

maxout  Maximum number of iterations on the majorization step of the QICD algorithm.

eps  Threshold for convergence of algorithm.

coef.cutoff  Coefficients with magnitude less than this value are set to 0.

a  Second tuning parameter for SCAD and MCP penalties. Must be greater than 2 
   for SCAD and greater than 1 for MCP.

...  The extra arguments will not be used.

Details

The QICD algorithm should only be used for the LASSO penalty if initial_beta can be set to LASSO 
estimates with a similar lambda (similar to a "warm start"). Otherwise, exact LASSO estimates will 
be used as initial values for the QICD algorithm: this will cause unnecessary computations and 
could lead to less accurate estimates.
Value

Returns a vector containing the intercept (if intercept=TRUE) and the estimated coefficients for each column in x.

Author(s)

Adam Maidman

References


Examples

```r
library(splines)
n = 50
p = 100
x <- matrix(rnorm(n*p), nrow=n)
z1 <- runif(n)
z2 <- runif(n)
x <- cbind(x, bs(z1), bs(z2))
groups <- c(1:p, rep(101,3), rep(102,3) )
y <- 0 + x[,1] - 3*x[,5] + z1^3 + rnorm(n)
fit1 <- QICD.group(y,x, groups, tau=.5, lambda=1, intercept=TRUE, penalty="SCAD")
fit2 <- QICD.group(y,x, groups, tau=.7, lambda=1, intercept=TRUE, penalty="SCAD")
```

QICD.master

Master QICD Function for Regular QICD, group QICD, and Partially Penalized QICD with Multiple Lambdas

Description

QICD.master produces penalized quantile regression estimates for all three cases of the QICD algorithm (QICD, QICD.nonpen, QICD.group). This function will find estimates for multiple lambdas.
Usage

QICD.master(y, x, z=NULL, groups=NULL, tau=.5, lambda, intercept=TRUE, penalty="SCAD", initial_beta, maxin=100, maxout=20, eps = 1e-05, coef.cutoff=1e-08, a=3.7, ...)

Arguments

y Vector of response values.

x n x p matrix of observations with each row corresponding to one observation. Penalties (and variable selection) will be applied to these coefficients. Do not include column of 1's; set intercept=TRUE, if intercept is desired.

z n x q matrix of observations with each row corresponding to one observation. Penalties (and variable selection) WILL NOT be applied to these coefficients. Do not include column of 1's; set intercept=TRUE, if intercept is desired. Set to NULL if all coefficients (except for intercept) should be penalized. Currently no support for z and groups.

groups Vector of length p with the group number of the corresponding coefficient. Coefficients in the same group will either all be 0, or none will be 0. Set to NULL if no groups. Currently no support for groups and z.

tau Conditional quantile being modelled.

lambda Tuning parameters for LASSO, SCAD, and MCP penalties. Must be positive.

intercept If TRUE, an intercept is included in the model. If FALSE, no intercept is included.

penalty Penalty function for the coefficients. Either "SCAD", "MCP", or "LASSO". See details for description of penalties.

initial_beta Vector of initial values for QICD algorithm. The vector should contain the intercept first (if intercept=TRUE) and then the p coefficients corresponding to x. Initial values for the coefficients corresponding to z can be passed after the the x coefficients, but will be ignored. These initial values will be used for all values of lambda. It is recommended to use LASSO estimates (with appropriately chosen lambda) as initial_beta.

maxin Maximum number of iterations on the minimization step of the QICD algorithm.

maxout Maximum number of iterations on the majorization step of the QICD algorithm.

eps Threshold for convergence of algorithm.

coef.cutoff Coefficients with magnitude less than this value are set to 0.

a Second tuning parameter for SCAD and MCP penalties. Must be greater than 2 for SCAD and greater than 1 for MCP.

... For partially penalized penalties, the method can be changed. See QICD.nonpen function.

Value

Returns the following:
coefficients Matrix of estimated coefficients corresponding to each value of lambda. The ith column corresponds to the ith lambda value in lambda.

lambda Unique values of lambda sorted into ascending order.

Author(s)
Adam Maidman

References

Examples
```r
n = 50
p = 100
x <- matrix(rnorm(n*p), nrow=n)
y <- 0 + x[,1] - 3*x[,5] + rnorm(n)
lambda <- exp(-20:0)
# fit1 <- QICD.master(y,x, tau=.5, lambda=lambda, intercept=TRUE, penalty="SCAD")
# fit2 <- QICD.master(y,x, tau=.7, lambda=lambda, intercept=TRUE, penalty="SCAD")
# head(fit1)
# fit2
```

QICD.nonpen Penalized Quantile Regression with some nonpenalized coefficients with QICD Algorithm

Description
QICD.nonpen produces penalized quantile regression estimates with some nonpenalized coefficients using the QICD algorithm. If no initial values are given, LASSO estimates will be used. This function can handle the LASSO, SCAD, and MCP penalties. This can be useful when you would like to perform variable selection only on some covariates and would like to guarantee that other covariates remain in the model.

Usage
```r
QICD.nonpen(y, x, z, tau=.5, lambda, intercept=TRUE, penalty="SCAD",
        initial_beta=NULL, maxin=100, maxout=20, eps = 1e-05,
        coef.cutoff=1e-08, a=3.7, method="br", …)
```
Arguments

\( y \) Vector of response values.

\( x \) \( n \times p \) matrix of observations with each row corresponding to one observation. Penalties (and variable selection) will be applied to these coefficients. Do not include column of 1’s; set intercept=TRUE, if intercept is desired.

\( z \) \( n \times q \) matrix of observations with each row corresponding to one observation. Penalties (and variable selection) WILL NOT be applied to these coefficients. Do not include column of 1’s; set intercept=TRUE, if intercept is desired.

\( \tau \) Conditional quantile being modelled.

\( \lambda \) Tuning parameter for LASSO, SCAD, and MCP penalties. Must be positive.

\( \text{intercept} \) If TRUE, an intercept is included in the model. If FALSE, no intercept is included.

\( \text{penalty} \) Penalty function for the coefficients. Either "SCAD", "MCP", or "LASSO". See details for description of penalties.

\( \text{initial\_beta} \) Vector of initial values for QICD algorithm. The vector should contain the intercept first (if intercept=TRUE) and then the p coefficients corresponding to x. Initial values for the coefficients corresponding to z can be passed after the the x coefficients, but will be ignored. If NULL, exact LASSO estimates will be computed and used as initial values.

\( \text{maxin} \) Maximum number of iterations on the minimization step of the QICD algorithm.

\( \text{maxout} \) Maximum number of iterations on the majorization step of the QICD algorithm.

\( \text{eps} \) Threshold for convergence of algorithm.

\( \text{coef\_cutoff} \) Coefficients with magnitude less than this value are set to 0.

\( a \) Second tuning parameter for SCAD and MCP penalties. Must be greater than 2 for SCAD and greater than 1 for MCP.

\( \text{method} \) Method used in QICD algorithm. Default is "br" method. When sample size is several thousand, method "fn" should be used for increased speed. See quantreg package for more details.

\(...\) The extra arguments will not be used.

Details

The QICD algorithm should only be used for the LASSO penalty if initial\_beta can be set to LASSO estimates with a similar lambda (similar to a "warm start"). Otherwise, exact LASSO estimates will be used as initial values for the QICD algorithm: this will cause unnecessary computations and could lead to less accurate estimates.

Value

Returns a vector containing the intercept (if intercept=TRUE) and the estimated coefficients for each column in x and z.

Author(s)

Adam Maidman
randomly_assign

References


Examples

```r
library(splines)
n = 50
p = 100
x <- matrix(rnorm(n*p), nrow=n)
z1 <- runif(n)
z2 <- runif(n)
z <- cbind(bs(z1), bs(z2))
y <- 0 + x[,1] - 3*x[,5] + z1^3 + sin(2*pi*z2) + rnorm(n)
fit1 <- QICD.nonpen(y,x,z, tau=.5, lambda=1, intercept=TRUE, penalty="SCAD")
fit2 <- QICD.nonpen(y,x,z, tau=.7, lambda=1, intercept=TRUE, penalty="SCAD")
```

---

randomly_assign

*Randomly Assign*

Description

Randomly assign n samples into k groups

Usage

`randomly_assign(n,k)`

Arguments

- `n`: Number of samples.
- `k`: Number of groups.

Value

A vector of length n having entries of 1,...,k.

Author(s)

Ben Sherwood
Examples

```r
randomly_assign(37,5)
randomly_assign(11,3)
```

**rq.group.fit**  
Quantile Regression with Group Penalty

**Description**

Similar to `cv.rq.pen` function, but uses group penalty. Group penalties use the L1 norm instead of L2 for computational convenience. We use a group penalty extension of the QICD algorithm presented by Peng and Wang (2015).

**Usage**

```r
rq.group.fit(x, y, groups, tau = 0.5, lambda, intercept = TRUE,
penalty = "lasso", alg = "qicd", a=3.7, penGroups=NULL, ...)
```

**Arguments**

- `x`  
  Matrix of predictors.
- `y`  
  Vector of response values.
- `groups`  
  Vector assigning columns of `x` to groups.
- `tau`  
  Conditional quantile being modelled.
- `lambda`  
  Vector of lambdas. Default is for lambdas to be automatically generated.
- `intercept`  
  Whether model should include an intercept. Constant does not need to be included in "x".
- `penalty`  
  Type of penalty: "LASSO", "SCAD" or "MCP".
- `alg`  
  If set to QICD algorithm will use coordinate descent algorithm. Otherwise, will use a linear programming algorithm, which is an extension of algorithm used in `rq.lasso.fit`.
- `a`  
  The additional tuning parameter for SCAD and MCP.
- `penGroups`  
  Specify which groups will be penalized. Default is to penalize all groups.
- `...`  
  Additional arguments to be sent to `rq.lasso.fit` or `groupQICD`.

**Value**

Returns the following:

- `coefficients`  
  Coefficients of the model.
- `residuals`  
  Residuals from the fitted model.
- `rho`  
  Unpenalized portion of the objective function.
- `tau`  
  Quantile being modelled.
- `n`  
  Sample size.
- `intercept`  
  Whether intercept was included in model.
Author(s)
Ben Sherwood; Adam Maidman

References

Examples

```r
x <- matrix(rnorm(800), nrow=100)
y <- 1 + x[,1] - 3*x[,5] + rnorm(100)
cv_model <- rq.group.fit(x, y, groups=c(rep(1,4), rep(2,4)), lambda=1, penalty="SCAD")
```

Description
Linear programming implementation of quantile regression with a group penalty.

Usage

```r
rq.group.lin.prog(x, y, groups, tau, lambda, intercept=TRUE, eps=1e-05, penalty="SCAD", a=3.7, coef.cutoff=1e-08, initial_beta=NULL, iterations=10, converge_criteria=0.01, penGroups=NULL,...)
```

Arguments

- `x` Matrix of predictors.
- `y` Vector of response values.
- `groups` Vector assigning columns of `x` to groups.
- `tau` Conditional quantile being modelled.
- `lambda` Vector of lambdas. Default is for lambdas to be automatically generated. Default is for lambdas to be automatically generated.
- `intercept` Whether model should include an intercept. Constant does not need to be included in "x".
- `eps` Multiplier for smallest lambda.
- `penalty` Type of penalty: "LASSO", "SCAD" or "MCP".
- `a` Additional parameter for non-convex penalties.
- `coef.cutoff` Estimates below this value are set to zero.
initial_beta  Initial beta estimate.
iterations  Maximum number of iterations.
converge_criteria
  Convergence criteria
penGroups  Specify which groups will be penalized. Default is to penalize all groups.
...  Additional arguments to be sent to rq.lasso.fit.

Value

Returns the following:

coefficients  Coefficients of the model.
residuals  Residuals from the fitted model.
rho  Unpenalized portion of the objective function.
tau  Quantile being modelled.
n  Sample size.
intercept  Whether intercept was included in model.
penalty  Penalty used for fitting the model.
class  rqPen and rqNC

Author(s)

Ben Sherwood

Examples

```r
## Not run:
x <- matrix(rnorm(800), nrow=100)
y <- 1 + x[,1] - 3*x[,5] + rnorm(100)
cv_model <- rq.group.lin.prog(x, y, groups=c(rep(1,4), rep(2,4)), tau=.5, lambda=1)
```

rq.lasso.fit  Quantile Regression with LASSO penalty

Description

Fits a quantile regression model with the LASSO penalty. Algorithm is similar to LASSO code presented in Koenker and Mizera (2014).

Usage

```r
rq.lasso.fit(x, y, tau=.5, lambda=NULL, weights=NULL,
  intercept=TRUE, coef_cutoff=.0000001,
  method="br", penVars=NULL, ...)
```
Arguments

- **x**: Matrix of predictors.
- **y**: Vector of response values.
- **tau**: Conditional quantile being modelled.
- **lambda**: Tuning parameter.
- **weights**: Weights for the objective function.
- **intercept**: Whether model should include an intercept. Constant does not need to be included in "x".
- **coef.cutoff**: Coefficients below this value will be set to zero.
- **method**: Use method "br" or "fn" as outlined in quantreg package. We have found "br" to be more stable for penalized regression problems.
- **penVars**: Variables that should be penalized. With default value of NULL all variables are penalized.
- **...**: Additional items to be sent to rq. Note this will have to be done carefully as rq is run on the augmented data to account for penalization and could provide strange results if this is not taken into account.

Value

Returns the following:

- **coefficients**: Coefficients from the penalized model.
- **PenRho**: Penalized objective function value.
- **residuals**: Residuals from the model.
- **rho**: Objective function evaluation without the penalty.
- **tau**: Conditional quantile being modelled.
- **n**: Sample size.

Author(s)

- Ben Sherwood

References


Examples

```r
x <- matrix(rnorm(800),nrow=100)
y <- 1 + x[,1] - 3*x[,5] + rnorm(100)
lassoModel <- rq.lasso.fit(x,y,lambda=1)
```
Fit Quantile Regression model for varying quantiles with LASSO penalty

Description

Fits quantile regression models for multiple quantiles with the LASSO penalty. Algorithm is similar to LASSO code presented in Koenker and Mizera (2014).

Usage

rq.lasso.fit.mult(x, y, tau_seq=c(.1, .3, .5, .7, .9), lambda=NULL,
                  weights=NULL, intercept=TRUE, coef.cutoff=.0000001,...)

Arguments

x Matrix of predictors.
y Vector of response values.
tau_seq Vector of quantiles of interest
lambda Tuning parameter.
weights Weights for the objective function.
intercept Whether model should include an intercept. Constant does not need to be included in "x".
coef.cutoff Coefficients below this value will be set to zero.
... Additional items to be sent to rq. Note this will have to be done carefully as rq is run on the augmented data to account for penalization and could provide strange results if this is not taken into account.

Value

Returns a list of rq.pen, rqLASSO objects.

Author(s)

Ben Sherwood

References

Examples

```r
x <- matrix(rnorm(800), nrow=100)
y <- 1 + x[,1] - 3*x[,5] + rnorm(100)
lassoModel <- rq.lasso.fit.mult(x, y, lambda=1)
```

**rq.nc.fit**

Non-convex penalized quantile regression

**Description**

Produces penalized quantile regression models for a range of lambdas and penalty of choice. If lambda is unselected than an iterative algorithm is used to find a maximum lambda such that the penalty is large enough to produce an intercept only model. Then range of lambdas goes from the maximum lambda found to “eps” on the log scale. Local linear approximation approach used by Wang, Wu and Li to extend LLA as proposed by Zou and Li (2008) to the quantile regression setting.

**Usage**

```r
rq.nc.fit(x, y, tau=.5, lambda=NULL, weights=NULL, intercept=TRUE,
penalty="SCAD", a=3.7, iterations=10,
converge_criteria=1e-06, alg=ifelse(p<50,"LP","QICD"),
penVars=NULL, ...)
```

**Arguments**

- **x**: Matrix of predictors.
- **y**: Vector of response values.
- **tau**: Conditional quantile being modeled.
- **lambda**: Tuning parameter.
- **weights**: Weights for the objective function.
- **intercept**: Whether model should include an intercept. Constant does not need to be included in "x".
- **penalty**: MCP or SCAD.
- **a**: Second tuning parameter.
- **iterations**: Number of iterations to be done for iterative LLA algorithm.
- **converge_criteria**: Difference in betas from iteration process that would satisfy convergence.
- **alg**: Defaults for small p to linear programming (LP), see Wang, Wu and Li (2012) for details. Otherwise a coordinate descent algorithm is used (QICD), see Peng and Wang (2015) for details. Both methods rely on the One-step sparse estimates algorithm.
- **penVars**: Variables that should be penalized. With default value of NULL all variables are penalized.
- **...**: Additional items to be sent to rq.lasso.fit.
Value

Returns the following:

- coefficients: Coefficients from the penalized model.
- PenRho: Penalized objective function value.
- residuals: Residuals from the model.
- rho: Objective function evaluation without the penalty.
- tau: Conditional quantile being modeled.
- n: Sample size.
- penalty: Penalty used, SCAD or MCP.

Author(s)

Ben Sherwood; Adam Maidman

References


Examples

```r
x <- matrix(rnorm(800), nrow = 100)
y <- 1 + x[, 1] - 3 * x[, 5] + rnorm(100)
scadModel <- rq.nc.fit(x, y, lambda = 1)
```

Description

SCAD penalty function as described in Fan and Li (2001).

Usage

```r
scad(x, lambda, a)
```
**Arguments**

- `x`: Number to be evaluated
- `lambda`: Tuning parameter lambda
- `a`: Tuning parameter a

**Value**

SCAD penalty function with tuning parameters lambda and "a" evaluated at "x".

**Author(s)**

Ben Sherwood

**References**


**Examples**

```r
scad(3,1)
scad(-3,1)
scad(.001,2)
```

---

**Description**

Derivative of SCAD penalty function as described in Fan and Li (2001).

**Usage**

`scad_deriv(x, lambda, a)`

**Arguments**

- `x`: Number to be evaluated
- `lambda`: Tuning parameter lambda
- `a`: Tuning parameter a. Default value of 3.7 as suggested in Fan and Li (2001)

**Value**

Derivative SCAD penalty function with tuning parameters lambda and "a" evaluated at "x".

**Author(s)**

Ben Sherwood
References


Examples

```r
scad_deriv(3, 1)
scad_deriv(-3, 1)
scad_deriv(.001, 2)
```

---

**square**  
*Square function*

Description

Square value of a number

Usage

```r
square(x)
```

Arguments

- `x`  
  Number to be squared.

Value

`x^2`

Author(s)

Ben Sherwood

Examples

```r
square(4)
square(-4)
square(2)
```
Index

Index entries:

- beta_plots, 3
- check, 4
- coef.cv.rq.group.pen, 5
- coef.cv.rq.pen, 6
- cv.rq.group.pen, 6
- cv.rq.pen, 8
- cv_plots, 10
- getRho, 11
- group_derivs, 13
- groupMultLambda, 11
- kernel_estimates, 13
- kernel_weights, 14
- lasso, 15
- LASSO.fit, 16
- LASSO.fit.nonpen, 17
- mcp, 18
- mcp_deriv, 19
- model_eval, 20
- nonzero, 21
- nonzero.cv.rq.group.pen, 21
- plot.cv.rq.group.pen, 22
- pos_part, 23
- predict.cv.rq.pen, 24
- predict.rq.pen, 25
- print.cv.rq.pen, 25
- print.rq.pen, 26
- qaSIS, 26
- qbic, 27
- QICD, 28
- QICD.group, 30
- QICD.master, 31
- QICD.nonpen, 33
- randomly_assign, 35
- rq.group.fit, 36
- rq.group.lin.prog, 37
- rq.lasso.fit, 38
- rq.lasso.fit.mult, 40
- rq.nc.fit, 41
- rqPen-package, 2
- scad, 42
- scad_deriv, 43
- square, 44