rqPen: An R package for penalized quantile regression

Ben Sherwood
School of Business
University of Kansas

Shaobo Li
School of Business
University of Kansas

Adam Maidman
School of Statistics
University of Minnesota

Abstract

Quantile regression directly models a conditional quantile of interest. The R package rqPen provides penalized quantile regression for lasso, elastic net, adaptive lasso, SCAD and MCP penalties, along with extensions to group penalties. The backbone of the package is the elastic net and group lasso methods, which are of interest in their own right and can be used to approximate the nonconvex penalties. The traditional approach to solving penalized quantile regression problems is to frame it as a linear programming, similar to what is done with unpenalized quantile regression. For large data sets this approach can be computationally burdensome. The package **rqPen** provides these traditional linear programming solutions, along with coordinate descent algorithms and algorithms based on approximating the quantile loss with a Huber-type approximation.

Keywords: quantile regression, penalized regression, model selection.

1. Introduction

The package **rqPen** allows users to model conditional quantiles using a penalized quantile regression approach. It supports a wide range of penalties and includes cross-validation and information criterion approaches for selecting tuning paramters. Koenker and Bassett (1978) proposed quantile regression as a robust alternative to mean regression that directly models a conditional quantile of interest without the need for assumptions about the distribution or moment conditions for the error term. Since the seminal paper of (Tibshirani 1996), introducing the lasso penalty, investigating the combination of different penalties and loss functions has been an active area of interest. Penalties provided in **rqPen** are lasso, elastic net (Zou and Hastie 2005), SCAD (Fan and Li 2001), MCP (Zhang 2010), adaptive lasso (Zou 2006), group lasso (Yuan and Lin 2005) and other group extensions of the previously stated penalties (Wang, Chen, and Li 2007; Huang, Breheny, and Ma 2012; Breheny and Huang 2009), with the exception of elastic net for which a group generalization has not been implemented. Extending the theoretical results of penalized estimators to the quantile regression setting has been an active area of research. Examples include deriving the rate of convergence for lasso (Belloni and Chernozhukov 2011) and group lasso (Kato 2012) and deriving oracle properties for non-convex penalties such as SCAD and MCP (Wang, Wu, and Li 2012). Discussed in these papers is how minimizing the penalized objective functions for quantile regression can be framed as linear programming problems or, in the case of group lasso, second order cone programming problems. The linear programming formulation is particularly familiar to researchers in quantile regression because this is the most common approach for solving quantile regression problems, including in the **quantreg** package (Koenker and D'Orey 1987, 1994). While a second order cone programming problem can be solved using convex optimization software, including the R package **Rmosek** (Koenker and Mizera 2014).

The ability to analyze large data sets is one of the major appeals of penalized regression methods. However, linear programming and second order cone programming becomes computationally burdensome for large data sets. Further complicating matters, is the quantile loss function is non-differentiable, while popular algorithms for penalized objective functions rely on a differentiable loss function, for instance Friedman, Hastie, and Tibshirani (2010) (elastic net), Breheny and Huang (2011) (non-convex penalties), Breheny and Huang (2015) (group non-convex penalties), and Yang and Zou (2015) (group lasso). Yi and Huang (2017) proposed using a Huber-type approximation of quantile loss and coordinate descent algorithm for solving elastic net penalized quantile regression which is implemented in the R package **hqreg**. Peng and Wang (2015) proposed a coordinate descent algorithm (QICD) for non-convex penalties that relies on an initial estimator being provided, implemented in **qicd**. For unpenalized and penalized quantile regression He, Pan, Tan, and Zhou (2023) and Tan, Wang, and Zhou (2022), respectively, proposed approximating the quantile loss with a convolution of the quantile loss function and a Kernel function that provides a twice differentiable loss function. This approach is implemented in **conquer**, available on CRAN. Other approaches, albeit without active R packages, for penalized quantile regression include using ADMM algorithms (Yu and Lin 2017; Yu, Lin, and Wang 2017; Gu, Fan, Kong, Ma, and Zou 2018).

The package **rqPen** provides implementation of the Huber based approximation, linear programming and QICD. It allows users to fit quantile regression models with all of the penalty functions discussed in the first paragraph. It also provides tools for using cross validation or information criterion for parameter selection. In addition, it provides plots for how cross validation results and coefficient estimation change with λ, the main sparsity tuning parameter. The package allows for estimation of multiple quantiles, enabling a visualization ofhow coefficient values change with the different quantiles being modeled. The packages **quantreg**, **conquer**, **hrqglaso** and **hqreg** are other alternatives for penalized quantile regression in R. However, there are some substantial differences between **rqPen** and these packages. The package **rqPen** allows users to fit quantile regression models using the elastic net, SCAD, MCP, adaptive lasso, group lasso, group SCAD, group MCP and group adaptive lasso penalties, along with allowing users to choose between an L_1 or L_2 norm for all group penalties. In addition, users can fit models for multiple quantiles, use cross-validation or IC approaches to choose tuning parameters and use plotting functions to create plots of how coefficients change with tuning parameters or by quantiles. Also, commonly used functions such as `predict()`, `coef()` and `plot()` can be used with the '`rq.pen.seq`' and '`rq.pen.seq.cv`' objects created by **rqPen** functions. The package **quantreg** is the gold standard for fitting unpenalized quantile regression models and provides the SCAD and lasso penalty. For both penalized and unpenalized quantile regression, **conquer** offers computationally efficient estimation for large n or p data sets. Users of **conquer** can choose between lasso, elastic net, group lasso, sparse-group(Simon, Friedman, Hastie, and Tibshirani 2013), group SCAD and group MCP. To the best of our knowledge **conquer** the only package that offers the sparse-group lasso penalty for quantile regression. While, **hrqglaso** provides penalized group lasso, it does not provide any of the other group penalties, nor does it allow for simultaneous estimation of multiple quantiles. On the other hand that package and **hqreg** are also setup to allow for robust mean regression using the Huber loss function, something **rqPen** is not setup to do. Both packages are required for **rqPen** and provide the backbone for the algorithms that use a Huber approximation for the group, **hrqglaso**, and non-group, **hqreg**, penalties.

2. Penalized estimation of quantile regrssion

Consider observations $\{y_i, \mathbf{x}_i\}_{i=1}^n$, where $\mathbf{x}_i \in \mathbb{R}^p$, and the model

$$y_i = \mathbf{x}_i^\top \boldsymbol{\beta}_0^\tau + \epsilon_i, \tag{1}$$

where $P(\epsilon_i < 0|\mathbf{x}_i) = \tau$. Define $\rho_\tau(u) = u[\tau - I(u < 0)]$ and Koenker and Bassett (1978) proposed estimating (1) by minimizing

$$\sum_{i=1}^n \rho_\tau(y_i - \mathbf{x}_i^\top \boldsymbol{\beta}), \tag{2}$$

which is available in the package **quantreg**. The package **rqPen** provides functions for estimating $\boldsymbol{\beta}_0^\tau$ by minimizing

$$\frac{1}{n}\sum_{i=1}^n \rho_\tau(y_i - \mathbf{x}_i^\top \boldsymbol{\beta}) + P_{\lambda,a}(\boldsymbol{\beta}), \tag{3}$$

where $\lambda > 0$ and the plausible values of a depend on the penalty being used. The penalty function $P_{\lambda,a}(\boldsymbol{\beta})$ can take the form of a group or individual penalty.

2.1. Individual Penalties

For individual penalties (3) has the form of

$$\frac{1}{n}\sum_{i=1}^n \rho_\tau(y_i - \mathbf{x}_i^\top \boldsymbol{\beta}) + \sum_{j=1}^p p_{w_j\lambda,a}(|\beta_j|) \tag{4}$$

The package **rqPen** supports four different forms of individual penalties: elastic net, lasso, SCAD and MCP. The SCAD, $p^s_{\lambda,a}()$, and MCP, $p^m_{\lambda,a}()$ penalty functions are

$$\begin{aligned} p^s_{\lambda,a}(|x|) &= \lambda|x|I(0 \le |x| < \lambda) + \frac{a\lambda|x| - (x^2+\lambda^2)/2}{a-1}I(\lambda \le |x| \le a\lambda) + \frac{(a+1)\lambda^2}{2}I(|x| > a\lambda), \\ p^m_{\lambda,a}(|x|) &= \lambda(|x| - \frac{x^2}{2a\lambda})I(0 \le |x| < a\lambda) + \frac{a\lambda^2}{2}I(|x| \ge a\lambda), \end{aligned}$$

where $a > 2$ for the SCAD penalty function and $a > 1$ for MCP.

The following provides the definition of $P_{\lambda,a}(\boldsymbol{\beta})$ for the four different penalty functions, plus two important special cases.

1. Elastic net: $p_{w_j\lambda,a}(|\beta_j|) = \lambda w_j \left[\alpha|\beta_j| + (1-\alpha)\beta_j^2\right]$, where $a \in [0, 1]$.
   (a) LASSO: special case with $a = 1$.
   (b) Ridge: special case with $a = 0$.

2. Adaptive LASSO: $p_{w_j\lambda,a}(|\beta_j|) = \lambda w_j|\tilde{\beta}_j|^{-a}|\beta_j|$, where $a > 0$ and $\tilde{\beta}_j$ is the Ridge estimator for the same value of λ.
3. SCAD: $p_{w_j\lambda,a}(|\beta_j|) = p^s_{w_j\lambda,a}(|\beta_j|)$, where $a > 2$.
4. MCP: $p_{w_j\lambda,a}(|\beta_j|) = p^m_{w_j\lambda,a}(|\beta_j|)$, where $a > 1$.

The weights, w_j, allow for different weights for each predictor and most be non-negative. If $w_j = 0$ then that variable will be unpenalized and thus is guaranteed to be included in the model. In **rqPen** these weights are refereed to as *penalty.factors* or *group.penalty.factors*. The LASSO estimator provides the backbone for the algorithm of the three non-elastic net penalties. As $\rho_\tau(x) + \rho_\tau(-x) = |x|$, the LASSO estimator minimizes,

$$\frac{1}{n}\sum_{i=1}^{n}\rho_\tau(y_i - \mathbf{x}_i^\top\boldsymbol{\beta}) + \sum_{j=1}^{p}\rho_\tau(\lambda w_j\beta_j) + \rho_\tau(-\lambda w_j\beta_j). \tag{5}$$

For $i \in \{1,\ldots,n\}$ define $\tilde{y}_i = y_i$ and $\tilde{\mathbf{x}}_i = \mathbf{x}_i$. Let $\mathbf{e}_j \in \mathbb{R}^{p+1}$ represent a unit vector with a value of one in the jth position and zero in all other entries. For $i \in \{n+1,\ldots,n+2p\}$, $\tilde{\mathbf{x}}_i = -n\lambda w_j\mathbf{e}_j$ or $\tilde{\mathbf{x}}_i = n\lambda w_j\mathbf{e}_j$, where each definition is used once for each value of $j \in \{1,\ldots,p\}$. While, $\tilde{y}_i = 0$ for $i \in \{n+1,\ldots,n+2p\}$. Then minimizing (5) is equivalent to minimizing

$$\frac{1}{n}\sum_{i=1}^{n+2p}\rho_\tau(\tilde{y}_i - \tilde{\mathbf{x}}_i^\top\boldsymbol{\beta}), \tag{6}$$

which, with the exception of the scaling constant of $\frac{1}{n}$, has the same form as (2). This approach of creating the augmented 2p samples and then using standard quantile regression is implemented in **rqPen** where the problem is solved using the `rq` function from **quantreg**. Note this approach is different than using `rq(method="lasso",...)` within **quantreg**, which uses a linear programming approach but does not formulate the problem using augmented data.

For large values of n and p the linear programming algorithms become computationally burdensome. To decrease computational complexity, Yi and Huang (2017) proposed approximating the quantile loss function with a Huber-like function and proposed a new coordinate descent algorithm that requires a differentiable loss function. Define the Huber loss function proposed by Huber (1964) as

$$h_\gamma(t) = \begin{cases} \dfrac{t^2}{2\gamma}, & \text{if } |t| \le \gamma, \\ |t| - \dfrac{\gamma}{2}, & \text{if } |t| > \gamma. \end{cases}$$

Note $\rho_\tau(u) = u[\tau - I(u < 0)] = \frac{1}{2}(|u| + (2\tau - 1)u)$ and for sufficiently small γ, $|u| \approx h_\gamma(u)$. We define the Huber-approximated quantile loss as

$$h^\tau_\gamma(u) = h_\gamma(u) + (2\tau - 1)u, \tag{7}$$

and for small γ, $\rho_\tau(u) \approx \frac{1}{2}h^\tau_\gamma(u)$. The package **hqreg** implements the approach of Yi and Huang (2017) and the function `hqreg()`, with `method="quantile"` solves the problem of

$$\frac{1}{2n}\sum_{i=1}^{n}h^\tau_\gamma(y_i - \mathbf{x}_i^\top\boldsymbol{\beta}) + \lambda\sum_{j=1}^{p}w_j|\beta_j|. \tag{8}$$

In **rqPen** (8) is solved by calling `hqreg::hqreg()`, and thus **hqreg** is a required package for **rqPen**. The Huber approximation is the default and will be used if the algorithm is not specified. For the adaptive lasso the connection is straight forward, as it is a special case of a lasso problem with different weights for each coefficients. The initial estimators necessary for the weights are determined by a ridge estimator with the same value of λ. The SCAD and MCP functions are approximated by a local linear approximation (LLA) as proposed by Zou and Li (2008). Let $p_{w_j\lambda,a}(|\beta_j|)$ represent a generic penalty function and $p'_{w_j\lambda,a}(|\beta_j|)$ be the derivative with respect to β_j. In addition let $\bar{\beta}_j$ be the lasso estimator for the same value of λ and weights. The LLA approach uses the following approximation,

$$\frac{1}{n}\sum_{i=1}^{n}\rho_\tau(y_i - \mathbf{x}_i^\top \boldsymbol{\beta}) + \sum_{j=1}^{p} p_{w_j\lambda,a}(|\beta_j|) \approx \frac{1}{n}\sum_{i=1}^{n}\rho_\tau(y_i - \mathbf{x}_i^\top \boldsymbol{\beta}) + \sum_{j=1}^{p} p'_{w_j\lambda,a}(|\bar{\beta}_j|)|\beta_j|. \tag{9}$$

Again, the problem becomes a special case of a lasso estimator with specific weights for each predictor. Thus all the non-group penalties, except for elastic net, can be solved using the linear programming or the Huber approximation algorithms. Peng and Wang (2015) proposed a coordinate descent algorithm for penalized objective functions with quantile loss and a non-convex penalty (QICD). This approach has been implemented for the SCAD and MCP functions. That proposed algorithm requires a good initial estimator and **rqPen** uses a lasso estimator. For this reason the algorithm was not implemented for the lasso penalty. For larger values of p and n this algorithm is faster than linear programming approaches. The Huber based algorithm will be faster than QICD, but the trade-off there is using an approximation for the quantile loss function.

The elastic net penalty of

$$\frac{1}{n}\sum_{i=1}^{n}\rho_\tau(y_i - \mathbf{x}_i^\top \boldsymbol{\beta}) + \lambda\sum_{j=1}^{p} w_j\left[a|\beta_j| + (1-a)\beta_j^2\right], \tag{10}$$

cannot be framed as a linear programming problem because of the ridge penalty. Thus for $a \neq 1$, the **rqPen** implementation of elastic net only uses the Huber approximation approach provided in **hqreg**. While **hqreg** provides a computational backbone, **rqPen** provides SCAD, MCP and adaptive lasso penalties that are not provided in `hqreg()`. In addition `hqreg()` does not include any group penalties.

2.2. Group Penalties

Group penalties are useful when there is a group structure to the predictors. Common examples of this are non-binary categorical variables or polynomial transformations of single predictor. This section assumes the p predictors are partitioned into G groups and $\boldsymbol{\beta}_g$ represents the coefficients associated with the gth group of predictors. Group penalized quantile regression estimators minimize

$$\frac{1}{n}\sum_{i=1}^{n}\rho_\tau(y_i - \mathbf{x}_i^\top \boldsymbol{\beta}) + \sum_{g=1}^{G} p_{w_g\lambda,a}(||\boldsymbol{\beta}_g||_q). \tag{11}$$

In **rqPen** user can choose four penalty functions for $p_{w_g,\lambda,a}()$: (1) lasso; (2) Adaptive lasso; (3) SCAD; and (4) MCP. Specifically,

1. lasso: $\sum_{g=1}^{G} p_{w_g\lambda,a}(||\boldsymbol{\beta}_g||_2) = \lambda \sum_{g=1}^{G} w_g||\boldsymbol{\beta}_g||_2$.
2. Adaptive lasso: $\sum_{g=1}^{G} p_{w_g\lambda,a}(||\boldsymbol{\beta}_g||_q) = \lambda \sum_{g=1}^{G} w_g||\tilde{\boldsymbol{\beta}}_g||_q^{-a}||\boldsymbol{\beta}_g||_q$, where $a > 0$ and $\tilde{\beta}_j$ is the Ridge estimator for the same value of λ.
3. SCAD: $P_a(\boldsymbol{\beta}) = \sum_{g=1}^{G} p^s_{w_j\lambda,a}(||\boldsymbol{\beta}_g||_q)$, where $a > 2$.
4. MCP: $P_a(\boldsymbol{\beta}) = \sum_{g=1}^{G} p^m_{w_j\lambda,a}(||\boldsymbol{\beta}_g||_q)$, where $a > 1$.

Currently, there is no implementation of a group elastic net or ridge penalty. A group ridge penalty could be done by choosing appropriate groups of weights for predictors. A group elastic net would be a convex combination of a group lasso penalty and ridge. Implementing this would require generalizing the group lasso penalty algorithm and has not been done, yet. The choice q is limited to $q \in \{1, 2\}$. If $q = 2$ then group variable selection will be all-or-nothing, that is all variables within a group will be selected or none of them will be. The choice of $q = 1$, allows for bi-level variable selection, that is it is possible to have zero and non-zero estimates for coefficients within a group. Consider the case of the lasso penalty for $q = 1$ (11) is

$$\frac{1}{n}\sum_{i=1}^{n} \rho_\tau(y_i - \mathbf{x}_i^\top \boldsymbol{\beta}) + \lambda \sum_{g=1}^{G} w_g||\boldsymbol{\beta}_g||_1, \tag{12}$$

while for $q = 2$ it is

$$\frac{1}{n}\sum_{i=1}^{n} \rho_\tau(y_i - \mathbf{x}_i^\top \boldsymbol{\beta}) + \lambda \sum_{g=1}^{G} w_g||\boldsymbol{\beta}_g||_2. \tag{13}$$

Note that (12) is a special case of the individual lasso penalty where the weights within each group are the same, while (13) is the standard group lasso penalty introduced by Yuan and Lin (2005). The estimator that minimizes (12) is not guaranteed to have variable selection agreement within a group, where this is guaranteed for the estimator that minimizes (13). Due to (12) being a special case of the individual lasso estimator it is not implemented as a group penalty, but for all other penalty functions users can choose between $q = 1$ or $q = 2$. For instance when using the SCAD penalty for $q = 1$ (11) is

$$\frac{1}{n}\sum_{i=1}^{n} \rho_\tau(y_i - \mathbf{x}_i^\top \boldsymbol{\beta}) + \sum_{g=1}^{G} p^s_{w_g\lambda,a}\left(||\boldsymbol{\beta}_g||_1\right), \tag{14}$$

while for $q = 2$ it is,

$$\frac{1}{n}\sum_{i=1}^{n} \rho_\tau(y_i - \mathbf{x}_i^\top \boldsymbol{\beta}) + \sum_{g=1}^{G} p^s_{w_g\lambda,a}\left(||\boldsymbol{\beta}_g||_2\right). \tag{15}$$

For an excellent review of group penalties, including a discussion of the use of $q = 1$ the reader is referred to Huang *et al.* (2012). Breheny and Huang (2009) provides a more detailed exploration of group penalties with $q = 1$. Motivation for the SCAD penalty was to find a penalty that provides the oracle property. For penalized likelihood based approaches, with a differentiable loss function, using a SCAD or MCP penalty the oracle properties for $q = 1$ and $q = 2$ are the same (Sherwood, Molstad, and Singha 2020).

Choosing $q = 1$ for quantile regression estimators has the benefit of being solved using linear programming, the most commonly used approach for quantile regression problems. As mentioned previously the group lasso estimator becomes a special case of the individual lasso estimator, and this is also true for the adaptive lasso penalty. For the SCAD and MCP penalties the same LLA will be used. Let $\bar{\boldsymbol{\beta}}_g$ be an initial group lasso estimator and the penalized objective function is approximated by

$$\frac{1}{n}\sum_{i=1}^{n}\rho_\tau(y_i - \mathbf{x}_i^\top\boldsymbol{\beta}) + \sum_{g=1}^{G} p'_{w_g\lambda,a}(||\bar{\boldsymbol{\beta}}_g||_q)||\boldsymbol{\beta}_g||_q. \tag{16}$$

For the case of $q = 1$ this becomes an individual lasso problem and the algorithms discussed in the previous section apply, though with some minor differences that will be discussed in the following section. While for $q = 2$ this becomes a special case of the group lasso problem. These group lasso problems are not linear programming problems, but second-order cone programming problems. Therefore they can be solved by existing convex optimization software such as **Rmosek** (Koenker and Mizera 2014). However, there are some barriers to a user in using **Rmosek**. A user needs to correctly define all the variables of the convex optimization problem and they need to have a copy of Mosek installed on their computer. In addition, similar to linear programming problems, second-order cone programming problems can be computationally burdensome for large values of n or p. For $q = 2$, the Huber approximation described in the previous subsection is used. However, **hqreg** cannot be used to solve this problem because the approach of Yi and Huang (2017) is not for a group penalty. Instead the algorithm of Yang and Zou (2015) is implemented. See Sherwood and Li (2022) for specific details regarding the application of this algorithm to the asymmetric Huber regression setting. This approach is available in the package **hrqglas**, which is maintained by two of the authors, and is used as the computational backbone for the L_2 group penalties.

2.3. Estimation of Multiple Quantiles

Say there are B quantiles, $(\tau_1, \ldots, \tau_B)$ of interest and let $\boldsymbol{\beta}_0^{\tau_b}$ be the true regression coefficients for the bth quantile. For a set of quantiles of interest, $(\tau_1, \ldots, \tau_B)$, Belloni and Chernozhukov (2011) propose minimizing,

$$\frac{1}{n}\sum_{b=1}^{B}\sum_{i=1}^{n}\rho_{\tau_b}\left(y_i - \mathbf{x}_i^\top\boldsymbol{\beta}^{\tau_b}\right) + \frac{\tilde{\lambda}}{n}\sum_{b=1}^{B}\sum_{j=1}^{p}\sqrt{\tau_b(1-\tau_b)}\hat{\sigma}_j|\beta_j|, \tag{17}$$

where $\hat{\sigma}_j = n^{-1}\sum_{i=1}^{n} x_{ij}^2$. We use $\tilde{\lambda}$ in (17) to emphasize that the scaling of the tuning parameter is different than that used in **rqPen**, but for $\lambda = \frac{\tilde{\lambda}}{n}$ these approaches are equivalent. More important, is highlighting that (17) provides a quantile specific weight of $\sqrt{\tau_b(1-\tau_b)}$ and a predictor specific weight of $\hat{\sigma}_j$, while λ is assumed to have the same value. This approach is generalized to allow for different penalties and user choices of the quantile and predictor weights. For individual penalties the estimator minimizes,

$$\frac{1}{n}\sum_{b=1}^{B}\sum_{i=1}^{n}\rho_{\tau_b}\left(y_i - \mathbf{x}_i^\top\boldsymbol{\beta}^{\tau_b}\right) + \sum_{b=1}^{B}\sum_{j=1}^{p} p_{w_j d_b\lambda,a}(|\beta_j^{\tau_b}|). \tag{18}$$

While for group penalties the penalized objective function is

$$\frac{1}{n}\sum_{b=1}^{B}\sum_{i=1}^{n}\rho_{\tau_b}\left(y_i - \mathbf{x}_i^\top\boldsymbol{\beta}^{\tau_b}\right) + \sum_{b=1}^{B}\sum_{g=1}^{G} p_{w_g d_b\lambda,a}(||\boldsymbol{\beta}_g||_q). \tag{19}$$

Note, the $w_j d_b \lambda$ term allows users to specify specific weights for a predictor, w_j, and specific weights for a quantile, d_b. For instance, if a user chooses the lasso penalty with $w_j = n^{-1}\sum_{i=1}^n x_{ij}^2$ and $d_b = n^{-1}\sqrt{\tau_b(1-\tau_b)}$ then (18) is equivalent to (17). The optimization of (18) and (19) consist of B different optimization problems that have been discussed in Sections 2.1 and 2.2. In Belloni and Chernozhukov (2011) they suggest choosing the same value of λ for all the quantiles, albeit with the quantile dependent weights. In the following, we present how users can either choose a single value of λ and coefficient estimates come from (18) or (19). Alternatively, users can choose B different values of λ and then the solutions come from B different optimizations of (4) or (11). This later approach is equivalent to a data-driven approach for estimate the optimal quantile specific weights, d_b.

2.4. Tuning parameter selection

There are two tuning parameters, λ and a, that need to be set. The value of a defaults to commonly used values for each penalty: (1) elastic-net (a=1); (2) adaptive lasso (a=1); (3) SCAD (a=3.7); and (4) MCP (a=3), while for Ridge ($a = 0$) and lasso ($a = 1$), the a parameter cannot be changed. Users can also specify their own value of a or a sequence of potential values of a to be considered. Typically the value of a is not as much of a concern as the value of λ, a potential notable exception to this would be the elastic net penalty. Users can specify their own sequence for values of λ, otherwise a sequence will be automatically generated. Define $H_\gamma^\tau(\boldsymbol{\beta}) = \frac{1}{2n}\sum_{i=1}^n h_\gamma^\tau(y_i - \mathbf{x}_i^\top\boldsymbol{\beta})$. Define $\tilde{a} = a$ if the elastic net penalty is used and one otherwise. If $a = 0$, that is the ridge penalty is being used, then $\tilde{a} = .001$. The default value for individual penalties is $\lambda_{\max} = \max_{j,b} 1.05 \left|\frac{\partial}{\partial\beta_j} H_\gamma^{\tau_b}(\mathbf{0}_{p+1})\right| (w_j d_b \tilde{a})^{-1}$ and for the group penalties is $\lambda_{\max} = \max_{g,b} 1.05 \left|\left|\frac{\partial}{\partial\boldsymbol{\beta}_g} H_\gamma^{\tau_b}(\mathbf{0}_{p+1})\right|\right|_q (w_g d_b)^{-1}$. Without the 1.05 multiplier, these values of $\lambda_{\max}$ provide a totally sparse solution for lasso and group lasso under the KKT conditions of the Huber approximation. However, not all algorithms use the Huber approximation and the 1.05 multiplier is used to make it more likely that $\lambda_{\max}$ will provide a totally sparse solution for any estimator. If any predictors or groups are unpenalized at a given quantile then they are excluded from these calculations.

Both cross-validation and information criterion are provided as tools for selecting the optimal pair of (λ, a). For a given quantile τ and pair of tuning parameters (λ, a) with estimator $\hat{\boldsymbol{\beta}}_{\lambda,a}^\tau$ with $k_{\lambda,a}^\tau$ non-zero coefficients, including the intercept, define the quantile information criterion (QIC) as

$$QIC(\tau,\lambda,a) = \log\left[\sum_{i=1}^n \rho_\tau(y_i - \mathbf{x}_i^\top \hat{\boldsymbol{\beta}}_{\lambda,a}^\tau)\right] + m k_{\lambda,a}^\tau, \tag{20}$$

where the value of m depends on the criterion being used. In **rqPen** the user can select between AIC ($m = 2$), BIC [$m = \log(n)$] and a version of the large p BIC proposed by Lee, Noh, and Park (2014) [$m = \log(n)\log(p)$].

Cross-validation is the other implemented approach for choosing the optimal pair of (λ, a). Consider the case of K folds, let n_k be the number of observations in the kth fold, y_i^k and $\mathbf{x}_i^k$ be the ith response and predictor vector for the kth fold and $\hat{\boldsymbol{\beta}}_{-k,\lambda,a}^{\tau_b}$ be the fit for the bth quantile excluding the kth fold for given values of λ and a. By default, even if Huber approximations are used, cross-validation is done with respect to quantile loss. However, this can be changed by setting *cvFunc* in the later described functions `rq.group.pen.cv()`

or `rq.pen.cv()`. For instance, `cvFunc=abs` will use absolute value loss regardless of the quantile being modeled. For simplicity of presentation the following assumes quantile loss is used. The average quantile loss at a given fold is

$$C_k^\tau(\lambda, a) = \frac{1}{n_k} \sum_{i=1}^{n_k} \rho_\tau(y_i^k - \mathbf{x}_i^{k\top} \hat{\boldsymbol{\beta}}_{-k,\lambda,a}^\tau). \tag{21}$$

The cross-validation functions return two summaries of the cross validation for selecting (λ, a). The return value *btr* provides a table of the values of a and λ that minimize

$$C^\tau(\lambda, a) = \frac{1}{K} \sum_{k=1}^{K} C_k^\tau(\lambda, a). \tag{22}$$

In addition it provides the λ that provides the sparsest solution that is within one standard error of the value that minimizes (22). The standard error for a given pair of (λ, a) is calculated as

$$\sqrt{\frac{1}{K-1} \sum_{k=1}^{K} \left[C_k^\tau(\lambda, a) - C^\tau(\lambda, a)\right]^2}. \tag{23}$$

The average summary can be replaced by changing the parameter *cvSummary*, for instance *cvSummary=median* would use the median values of $C_k^\tau(\lambda, a)$. The return value *gtr* provides results for the value of λ and a that minimize

$$C(\lambda, a) = \sum_{b=1}^{B} w_b C^{\tau_b}(\lambda, a). \tag{24}$$

Users can choose between a single pair of (λ, a) that minimizes (24) or B, possibly different, pairs that minimize (22) separately for each quantile. The results also include results for the one standard error approach, where the standard error for a fixed (λ, a) pair is

$$\sqrt{\frac{1}{K-1} \sum_{k=1}^{K} \sum_{b=1}^{B} w_b \left[C_k^\tau(\lambda, a) - C^\tau(\lambda, a)\right]^2}. \tag{25}$$

The one standard error rule finds the sparsest solution, by increasing λ, where the value of (22) or (24) are within one standard error of the optimal value, assuming the optimal a is fixed.

By setting *groupError=FALSE* each individual cross-validation error is weighted equally and (22) and (24) are replaced by

$$\sum_{k=1}^{K} n_k C_k^\tau(\lambda, a), \tag{26}$$

and

$$\sum_{b=1}^{B} w_b \sum_{k=1}^{K} n_k C_k^\tau(\lambda, a), \tag{27}$$

respectively.

Tuning parameter selection for multiple quantiles using information criterion

For the case of multiple, B, quantiles being modeled, **rqPen** offers two different approaches for selecting the tuning parameters. One approach is to select B pairs of (λ, a) that minimize the B different versions of (20). Define B weights of $(w_1, \ldots, w_B)$. The other approach is to select a single pair of (λ, a) that minimizes,

$$\sum_{b=1}^{B} z_b QIC(\tau_b, \lambda, a). \tag{28}$$

The weights offer flexibility to a user who wants to provide more weight to a specific quantile. The default value for all the weights is one thus giving equal weight to each quantile.

2.5. Additional notes on Huber approximation

For Huber approximation an appropriate value of γ is needed. If the value is too large then the estimator will have a large amount of bias, but if the value is too small the algorithms will become unstable. The values of γ are updated for each value of λ. Let $\mathbf{r}^k$ be the vector of residuals corresponding to the estimator using the kth value of λ, λ^k, and $Q_{.1}(|\mathbf{r}^k|)$ be the 10th percentile of the absolute values of these residuals. For the individual penalties we use the approach outlined by Yi and Huang (2017), with differences only coming from later changes that were made in **hqreg**,

$$\begin{aligned} \gamma^k &= I(|1-\tau| < .05) \max\left\{.0001, \min\left[\gamma^{k-1}, Q_{.01}(|\mathbf{r}^k|)\right]\right\} \\ &\quad + I(|1-\tau| >= .05) \max\left\{.001, \min\left[\gamma^{k-1}, Q_{.1}(|\mathbf{r}^k|)\right]\right\}. \end{aligned}$$

For the group penalties, as presented in Sherwood and Li (2022),

$$\gamma^k = \min\left\{4, \max\left[.001, Q_{.1}(|\mathbf{r}^k|)\right]\right\}.$$

Generally speaking, γ needs to be small, but not too small. For larger values of γ the approximation becomes closer to a least squares or expectile loss function (Newey and Powell 1987). As expectile regression is not the same as quantile regression and least squares is only the same if the errors are symmetric, this can result in biased estimators. For values of γ that are too small the algorithms become unstable as the differentiable loss function becomes for practical purposes non-differentiable.

One reason the linear and second-order cone programming problems are computationally slow for larger values of p is they optimize with respect to all the potential covariates. Tibshirani *et al.* (2012) proposed a strong rule for discarding predictors that can greatly decrease the computational complexity of penalized optimization. The strong rule assumes a differentiable loss function and thus is only implemented for the Huber approximations. In addition the Huber approximation approaches use warm starts across a path of potential λ values. It starts with the largest potential value of λ then uses that solution as the initial value for the next largest potential value of λ. This iterative process is continued until all solutions have been found. The linear programming approaches rely on `quantreg::rq()` that does not have an option for initial values. Thus warm starts are not implemented for those approaches. For the individual penalties, the calculations of the strong rule are done in the package **hqreg**, and details can be found in Yi and Huang (2017). A similar approach has been used for the group penalties (Sherwood and Li 2022).

3. Description of functions

Below are the main functions of **rqPen** and an S3 method that is unique to **rqPen**, `bytau.plot`,

- `rq.pen()` provides estimates of a quantile regression model, for potentially multiple quantiles, using an individual penalty of either elastic-net, adaptive lasso, SCAD or MCP. If not specified, will automatically generate a sequence of λ values. Returns an '`rq.pen.seq`' S3 object that works `bytau.plot`, `coef`, `plot`, `predict`, and `print` methods.

- `rq.group.pen()` is a group penalty version of `rq.pen()`. Users have access to group versions of lasso, adaptive lasso, SCAD and MCP. The lasso penalty is restricted to the L2-norm, but for other penalties users can choose between the L1 or L2 norm. It also returns an an '`rq.pen.seq`' S3 object and can be used with the same methods listed above.

- `rq.pen.cv()` automates K-folds cross-validation for selection of the tuning parameters λ and a. A sequence of λ will be automatically generated. However, for a a sequence needs to be supplied, otherwise a single default value will be used. If multiple quantiles are modeled then finds the optimal pair of (λ, a) for each quantile and the optimal pair across all quantiles. Returns an '`rq.pen.seq.cv`' S3 object that works with `bytau.plot`, `coef`, `plot`, `predict`, and `print` methods.

- `rq.group.pen.cv()` is the group version of `rq.pen.cv()`. It does everything listed above, but for group penalties. It also returns an '`rq.pen.seq.cv`' object.

- `qic.select()` takes a '`rq.pen.seq`' or '`rq.pen.seq.cv`' object and provides the optimal values of (λ, a) using information criterion explained in the previous section. It returns a '`qic.select`' S3 object that works with `coef`, `predict` and `print` methods.

- `bytau.plot()` plots coefficients estimates for each predictor as a function of τ.

The `rq.pen()` function provides estimates of conditional quantiles from minimizing a penalized objective function for a sequence of λ values.

```
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 = ifelse(sum(dim(x)) < 200, "br",
"huber"), scalex = TRUE, tau.penalty.factor = rep(1, length(tau)), coef.cutoff
= 1e-08, max.iter = 10000, converge.eps = 1e-07, lambda.discard = TRUE,...)
```

The function `rq.pen()` requires a design matrix `x` and vector of response `y`. The predictors in the design matrix will be centered to have mean zero and standard deviation when minimizing the penalized objective function unless `scalex` is set to FALSE. While the predictors are scaled, the coefficients are returned with respect to the original scale of `x`. The quantiles modeled are set with `tau`. Users can choose the `penalty` function, with the default being lasso. Users can choose to set their own lambda sequence, `lambda`, and choose the number of lambda values `nlambda`. If the user does not specify a sequence of λ values then $\lambda_{min} = \epsilon\lambda_{max}$, where ϵ can be set by `eps`. Though very small values of λ may be discarded if the coefficient estimates are not changing much with smaller values of λ, unless `lambda.discard=FALSE`. For non-ridge or non-lasso penalties, A sequence of values can also be selected for `a`. If `a` is not set then default values are elastic-net ($a = 0$), adaptive lasso ($a = 1$), SCAD ($a = 3.7$) and MCP ($a = 3$). Penalty factors can be set for predictors, `penalty.factor`, or quantiles, `tau.penalty.factor`. Sometime the linear programming approaches can provide very small, but non-zero estimates, and these coefficients are set to zero if they are below `coef.cutoff`. For algorithm, `alg`, users can choose between linear programming approaches of "br" or "fn", "huber" to use the Huber approximation or "qicd" to use the quantile coordinate descent algorithm for SCAD or MCP. In addition, they can set the maximum number of iterations, `max.iter`, or converge criteria, `converge.eps`, though these only apply to the "huber" and "qicd" algorithms.

The function `rq.pen.cv()` provides cross-validation for a model that can be fit using `rq.pen`.

```
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, ...)
```

The arguments `x`, `y`, `tau`, `lambda`, `penalty`, `a` and `nlambda` are the same as in `rq.pen()`, and the remaining arguments in `rq.pen()` can also be set in `rq.pen.cv()`. The loss function for cross-validation can be specified in `cvFunc`, where the default is to use quantile loss. The number of folds can be specified, `nfolds`, or users can specify their own partitioning of the data with `foldid`. If `groupError=FALSE` then each observation will contribute equally to the cross-validation error, otherwise each fold contributes equally. The default is to report the mean across the folds, but another function, such as `median()` can be set in `cvSummary`. The partition being used for testing data will be printed on if `printProgress=TRUE`. If `rq=rq.pen.cv`, the underlying '`rq.pen.seq`' object can be accessed by `rq$fit`.

The function `rq.group.pen()` fits a quantile regression model using a group penalty.

```
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", "qicd"), a = NULL, norm = 2, group.pen.factor
= rep(1, length(unique(groups))), tau.penalty.factor = rep(1, length(tau)), scalex
= TRUE, coef.cutoff = 1e-08, max.iter = 10000, converge.eps = 1e-07, gamma = IQR(y)/10,
lambda.discard = TRUE, ...)
```

The arguments `x`, `y`, `penalty`, `tau`, `lambda`, `nlambda`, `eps`, `alg`, `a`, `tau.penalty.factor`, `scalex`, `coef.cutoff`, `max.iter`, `converge.eps`, `lambda.discard` are the same as in `rq.pen()`, though the types of penalties are different. The `groups` need to be specified, the default is to have p singleton groups. The q-norm can be set to 1 or 2 in `norm`. Group penalty factors are set with `group.pen.factor`. The initial value of `gamma` for the Huber approximation can be set, but that algorithm adaptively tunes this value.

The function `rq.group.pen()` provides cross-validation for a model that can be fit using `rq.group.pen()`.

```
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, ...)
```

All of the arguments to `rq.group.pen.cv()` have been covered in the discussion of the three previous functions.

Information criterions provide an alternative to cross-validation for selecting tuning parameters and this can be done using `qic.select()`, which takes an '`rq.pen.seq`' or '`rq.pen.seq.cv`' object and returns a '`qic.select`' object. Below are the arguments for `qic.select()` with a '`rq.pen.seq`' object.

```
qic.select.rq.pen.seq(obj, method = c("BIC", "AIC", "PBIC"), septau = TRUE, weights
= NULL, ...)
```

The `object` in this case would be a '`rq.pen.seq`' object, but either '`rq.pen.seq`' or '`rq.pen.seq.cv`' objects can be passed to `qic.select()`. For the IC '`method`' users can choose between BIC, AIC or large p BIC ("PBIC"). If `septau=TRUE` then tuning parameter pairs of (λ, a) are minimized separately for each quantile. If `septau=FALSE` users can choose to weight the different quantiles using `weights`.

The last function unique to **rqPen** is `bytau.plot()` which provides a plot of how coefficient values change with quantiles being modeled. First, we discuss how `coef()` works with '`rq.pen.seq`', '`rq.pen.seq.cv`' and '`qic.select`' objects. For a '`qic.select`' object, say "qc", then `coef(qc)` will return the coefficients for each quantile that were selected according to the IC criteria provided to `qic.select()`. For '`rq.pen.seq`' and '`rq.pen.seq.cv`' objects there are more options.

```
coef.rq.pen.seq(object, tau = NULL, a = NULL, lambda = NULL, modelsIndex = NULL,
lambdaIndex = NULL, ...)
```

For an '`rq.pen.seq`' object users can specify the quantiles, `tau`, and tuning parameters, `a` and `lambda`, that they wish to extra coefficient values for. Alternatively, they can directly specify the models and lambda values using `modelsIndex` and `lambdaIndex`, respectively. If none of these values are set then a matrix of coefficients for all quantiles and tuning parameters will be returned.

```
coef.rq.pen.seq.cv(object, septau = TRUE, cvmin = TRUE, useDefaults = TRUE, tau
= NULL, ...)
```

With a '`rq.pen.seq.cv`' object the `coef()` function returns coefficients based on the cross validation results. If `cvmin=TRUE` then the tuning parameters associated with the minimum cross validation error is returned, otherwise the one standard error rule is used. When septau=TRUE then the tuning parameters are optimized individually for each quantile. Users can identify specific the quantiles, tau, they wish to return, the default is to return results for all quantiles. Setting useDefaults=FALSE ignores the cross validation results and users can use the arguments in coef.rq.pen.seq() to select coefficients. For instance, if useDefaults=FALSE and no other arguments are provided then all coefficients are returned. Function bytau.plot() has a similar form for the 'rq.pen.seq' and 'rq.pen.seq.cv' objects.

```
bytau.plot.rq.pen.seq(x, a = NULL, lambda = NULL, lambdaIndex = NULL, ...)
```

```
bytau.plot.rq.pen.seq.cv(x, septau = TRUE, cvmin = TRUE, useDefaults = TRUE, ...)
```

These functions will produce a plot of how coefficients change with quantiles. The arguments in bytau.plot.rq.pen.seq() and bytau.plot.rq.pen.seq.cv() are the same as those covered for coef(), but with one major difference. Only one vector of coefficients can be provided for each quantile. When using a 'rq.pen.seq' object users must specify a single pair of λ. While a 'rq.pen.seq.cv' object can rely on the default tools for selecting coefficients or specify a single pair of λ and a.

Both 'rq.pen.seq' and 'rq.pen.seq.cv' objects have plot() functions. Using plot with a 'rq.pen.seq.cv' object provides a plot of how the cross validation error changes with λ for each quantile.

```
plot.rq.pen.seq.cv(x, septau = TRUE, tau = NULL, logLambda = FALSE, main = NULL,
...)
```

If septau=TRUE then a separate plot is created for each quantile, otherwise one plot is created that combines the cross validation error across all quantiles. Users can choose a subset of quantiles, tau, to plot results from. If logLambda=TRUE then the x-axis will be $\log(\lambda)$. The main text can be set otherwise the main title depends on the name of the variable and the value of septau.

While, plot() for a 'rq.pen.seq' object creates a plot of the coefficients as the change with λ.

```
plot.rq.pen.seq(x, vars = NULL, logLambda = TRUE, tau = NULL, a = NULL, lambda
= NULL, modelsIndex = NULL, lambdaIndex = NULL, main = NULL, ...)
```

The plot() function will create a plot for each combination of τ and a. Users can specify specific values of tau and a if they only want to consider a subset. They can also limit the values of lambda. Choice of models and λ values can also be done using modelsIndex and lambdaIndex. The main text can be set. If logLambda=TRUE then the λ values are reported on the natural log scale.

4. Applications

This section details fitting using the aforementioned barro data set and the Ames housing data (De Cock 2011). In the barro data set functions rq.pen() and rq.pen.cv() are used, while for the Ames data set `rq.group.pen()` and `rq.group.pen.cv()` are used.

4.1. barro

The barro data set, available in **quantreg**, contains GPD growth rates and 13 other potential explanatory variables. See Koenker and Machado (1999) for more details.

Using different algorithms

First we look at fitting a model with the SCAD penalty for $\tau = .5$ using the four different algorithms.

```
R> library(rqPen)
R> #quantreg is required for rqPen, but call directly here
R> #because we need the barro data set
R> library(quantreg)
R> data(barro)
R> y <- barro$y.net
R> x <- as.matrix(barro[,-1])
R> qbr <- rq.pen(x,y,alg="br", penalty="SCAD")
R> qfn <- rq.pen(x,y,alg="fn", penalty="SCAD")
R> qhuber <- rq.pen(x,y,alg="huber",penalty="SCAD")
R> qqicd <- rq.pen(x,y,alg="QICD", penalty="SCAD")
```

Where “br” and “fn” are the Barrodale and Roberts(Koenker and D’Orey 1987, 1994) and Frisch-Newton(Portnoy and Koenker 1997) interior point method for linear programming problems implemented in `quantreg:::rq()`. The following code takes the 25th value in the sequence of λ values and compares the coefficients for the four different approaches creating four different ‘`rq.pen.seq`’ objects.

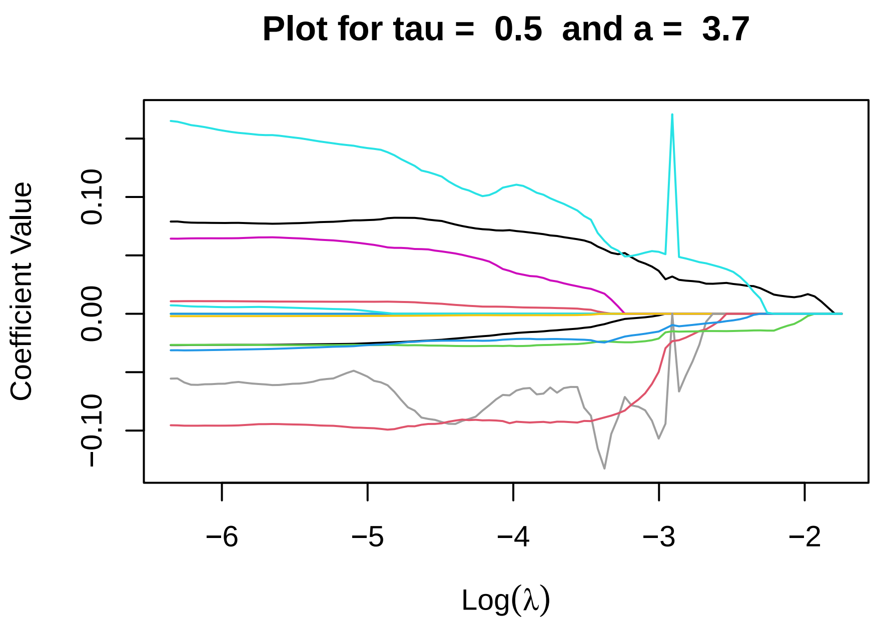
```
R> targetLambda <- qbr$lambda[25]
R> brCoef <- coefficients(qbr,lambda=targetLambda)
R> fnCoef <- coefficients(qfn, lambda=targetLambda)
R> huberCoef <- coefficients(qhuber, lambda=targetLambda)
R> qicdCoef <- coefficients(qqicd, lambda=targetLambda)
R> coefDf <- cbind(brCoef,fnCoef,huberCoef,qicdCoef)
R> colnames(coefDf) <- c("br","fn","huber","qicd")
R> coefDf
```

```
                  br         fn      huber         qicd
intercept 0.02018841 0.02018852 0.02136612  0.040130199
lgdp2     0.00000000 0.00000000 0.00000000 -0.002284152
mse2      0.00000000 0.00000000 0.00000000  0.000000000
fse2      0.00000000 0.00000000 0.00000000  0.000000000
fhe2      0.00000000 0.00000000 0.00000000  0.000000000
mhe2      0.00000000 0.00000000 0.00000000  0.000000000
lexp2     0.00000000 0.00000000 0.00000000  0.000000000
```

```
lintr2     0.00000000  0.00000000  0.00000000  0.000000000
gedy2     -0.01773640 -0.01773726 -0.06654418 -0.007484307
Iy2        0.02366121  0.02366100  0.02905822  0.031260853
gcony2    -0.02007166 -0.02007206 -0.02261462 -0.049204738
lblakp2   -0.01433958 -0.01433962 -0.01527611 -0.014414122
pol2      -0.01112295 -0.01112292 -0.01071062 -0.012459128
ttrad2     0.04986245  0.04986267  0.04860164  0.053067947
```

The four different algorithms provide four different coefficient estimates that are qualitatively similar. For larger data sets, with respect to either sample size or variables, we recommend using the Huber approximation because this greatly reduces computational time while providing a reasonable solution (Yi and Huang 2017). The following presents a plot of how the coefficient values from `qhuber` change with λ.

```
R> plot(qhuber)
```



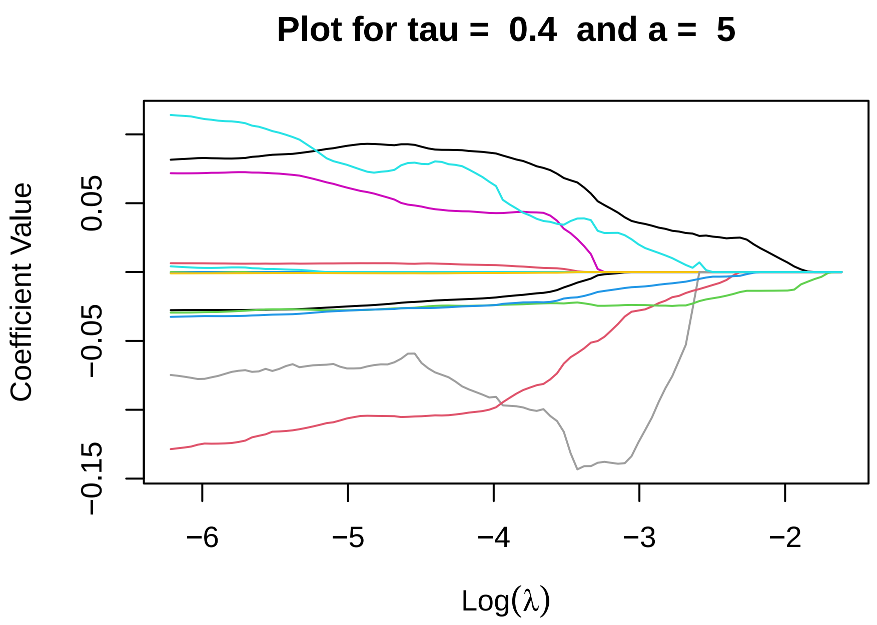
Multiple quantiles

Next, we fit a model for multiple quantiles of $\tau \in \{.1, .2, \ldots, .9\}$ and will also consider values of $a \in \{3, 4, 5\}$ for the additional tuning parameter for SCAD. A fit with multiple quantiles remains a '`rq.pen.seq`' object.

```
R> qmult <- rq.pen(x,y,tau=seq(.1,.9,.1),a=seq(3,5),penalty="SCAD")
```

The following code provides a plot of the coefficients for $\tau = .4$ and $a = 5$.

```
R> plot(qmult, tau=.4,a=5)
```



Using IC to select tuning parameters

Below is code to select the tuning parameters of (λ, a) for the `qmult` model using AIC. This code will create two '`qic.select`' objects, `qic_sep` and `qic_joint`.

```
R> qic_sep <- qic.select(qmult,method="AIC")
R> qic_joint <- qic.select(qmult, method="AIC", septau=FALSE)
```

Below provides the tuning parameters selected by the two different approaches. The `modelsInfo` attribute provides information on the model selected for each quantile. Coefficients can be extracted using `coef()`.

```
R> qic_sep$modelsInfo

   tau modelIndex a   minQIC lambdaIndex       lambda
1: 0.1          1 3 3.683767          97 0.002294493
2: 0.2          4 3 4.986045         100 0.001995633
3: 0.3          9 5 5.433769          98 0.002190205
4: 0.4         10 3 5.564932         100 0.001995633
5: 0.5         15 5 5.589880         100 0.001995633
6: 0.6         18 5 5.738133          98 0.002190205
7: 0.7         19 3 5.480456         100 0.001995633
8: 0.8         22 3 4.760914          95 0.002518203
9: 0.9         25 3 2.978646          99 0.002090657
```

```
R> qic_joint$modelsInfo

         modelIndex a tau   minQIC lambdaIndex      lambda
tau0.1a3          1 3 0.1 3.683982         100 0.001995633
tau0.2a3          4 3 0.2 4.986045         100 0.001995633
tau0.3a3          7 3 0.3 5.458126         100 0.001995633
tau0.4a3         10 3 0.4 5.564932         100 0.001995633
tau0.5a3         13 3 0.5 5.601752         100 0.001995633
tau0.6a3         16 3 0.6 5.756723         100 0.001995633
tau0.7a3         19 3 0.7 5.480456         100 0.001995633
tau0.8a3         22 3 0.8 4.761124         100 0.001995633
tau0.9a3         25 3 0.9 2.978665         100 0.001995633

R> coef(qic_sep)

               tau=0.1      tau=0.2      tau=0.3      tau=0.4      tau=0.5
intercept  0.011325407 -0.017459218 -0.014496871 -0.053056801 -0.038046008
lgdp2     -0.028132891 -0.025860545 -0.025532048 -0.027709382 -0.026691105
mse2       0.013288598  0.008775746  0.006468634  0.006829600  0.010711403
fse2       0.000978223  0.004775398  0.002571248  0.000000000  0.000000000
fhe2      -0.022975351 -0.024707772  0.007507245  0.000000000  0.000000000
mhe2       0.010217203  0.008563773  0.000000000  0.003684845  0.006581601
lexp2      0.054850467  0.058927699  0.056538780  0.071027197  0.065132350
lintr2    -0.001133734 -0.001909570 -0.001621095 -0.001009784 -0.002055311
gedy2     -0.373880872 -0.295789636 -0.087072194 -0.072942962 -0.061636582
Iy2        0.081368094  0.077582802  0.088175767  0.081262842  0.077450448
gcony2    -0.204386365 -0.190029809 -0.150795760 -0.129824786 -0.094919513
lblakp2   -0.023736703 -0.025787696 -0.029525690 -0.029501489 -0.026655910
pol2      -0.028651420 -0.030021577 -0.031453741 -0.032774733 -0.031023563
ttrad2     0.116612338  0.114958988  0.089690003  0.112472383  0.158627515
               tau=0.6      tau=0.7      tau=0.8       tau=0.9
intercept -0.029887183 -0.024698529 -0.081557007 -0.0393325613
lgdp2     -0.028207627 -0.028064396 -0.030604396 -0.0334852076
mse2       0.013886241  0.022198000  0.022884492  0.0212605769
fse2       0.000000000 -0.005718433 -0.009768302 -0.0079038518
fhe2       0.001973450  0.000000000 -0.010097068  0.0000000000
mhe2       0.013613919  0.006321253  0.000000000  0.0000000000
lexp2      0.067195909  0.065457541  0.084482901  0.0823329536
lintr2    -0.003229847 -0.003205599 -0.002027544 -0.0023088733
gedy2     -0.115178952 -0.080049606  0.124954829 -0.0241901389
Iy2        0.077267579  0.072604810  0.055315818  0.0576063671
gcony2    -0.102761017 -0.104151382 -0.083788007 -0.0911038215
lblakp2   -0.029386457 -0.027236468 -0.030516963 -0.0335016439
pol2      -0.024401767 -0.018913389 -0.006792098  0.0005022515
ttrad2     0.191920505  0.216779484  0.224462314  0.2262907484

R> coef(qic_joint)
```

```
                tau=0.1      tau=0.2      tau=0.3      tau=0.4      tau=0.5
intercept  0.011848020 -0.017459218 -0.009590736 -0.053056801 -0.032738451
lgdp2     -0.028021094 -0.025860545 -0.025554042 -0.027709382 -0.026790676
mse2       0.013102269  0.008775746  0.006642524  0.006829600  0.010731908
fse2       0.001355050  0.004775398  0.002756279  0.000000000  0.000000000
fhe2      -0.024024384 -0.024707772  0.008194420  0.000000000  0.000000000
mhe2       0.010638228  0.008563773  0.000000000  0.003684845  0.006945651
lexp2      0.054457129  0.058927699  0.055392908  0.071027197  0.063819347
lintr2    -0.001189216 -0.001909570 -0.001734245 -0.001009784 -0.002045861
gedy2     -0.371296136 -0.295789636 -0.089666730 -0.072942962 -0.051230003
Iy2        0.080968618  0.077582802  0.088423947  0.081262842  0.079605243
gcony2    -0.203600349 -0.190029809 -0.152798978 -0.129824786 -0.096184478
lblakp2   -0.023520034 -0.025787696 -0.029636950 -0.029501489 -0.026566220
pol2      -0.028668797 -0.030021577 -0.031676568 -0.032774733 -0.031165598
ttrad2     0.116795353  0.114958988  0.090868194  0.112472383  0.165241719
                tau=0.6      tau=0.7      tau=0.8       tau=0.9
intercept -0.029389067 -0.024698529 -0.080452949 -0.0392609848
lgdp2     -0.028299800 -0.028064396 -0.030573146 -0.0334838681
mse2       0.014010752  0.022198000  0.023250123  0.0212971527
fse2       0.000000000 -0.005718433 -0.010266364 -0.0079567091
fhe2       0.002677534  0.000000000 -0.011098353  0.0000000000
mhe2       0.013468792  0.006321253  0.000000000  0.0000000000
lexp2      0.067270194  0.065457541  0.083996617  0.0823082688
lintr2    -0.003276889 -0.003205599 -0.001996580 -0.0023048683
gedy2     -0.114576164 -0.080049606  0.143533627 -0.0235707659
Iy2        0.077010441  0.072604810  0.056374224  0.0576359178
gcony2    -0.103758949 -0.104151382 -0.086830831 -0.0912776256
lblakp2   -0.029274274 -0.027236468 -0.030441125 -0.0335009792
pol2      -0.024684074 -0.018913389 -0.006540822  0.0004966526
ttrad2     0.192454318  0.216779484  0.221634421  0.2261160322
```

When jointly optimizing the tuning parameter selection the same values of λ and a will be selected for each quantile, minimizing (28), otherwise the optimal tuning parameters will be selected by minimizing (20) for each quantile separately. Users can specify `method="AIC"`, `method="BIC"` or `method="PBIC"` for a large p BIC proposed by Lee *et al.* (2014). When doing joint selection, users can also specify `weights`, the values of z_b in (28). The default is to provide an equal weight, of one, for all quantiles.

Predictions from models

Users can make predictions from either the '`qic.select`' or '`rq.pen.seq`' objects. Prediction from a '`qic.select`' object will return predictions for all quantiles modeled.

```
R> # creating new data using the mean of all variables
R> newData <- apply(barro,2,mean)[-1] #removing response
R> predict(qic_sep,newData)
```

```
           tau=0.1     tau=0.2   tau=0.3    tau=0.4    tau=0.5    tau=0.6
```

```
[1,] -0.0001469578 0.004810161 0.0108926 0.01405191 0.01924728 0.02428196
        tau=0.7    tau=0.8    tau=0.9
[1,] 0.02763724 0.03275538 0.03906455
```

The default for `predict()` for a '`rq.pen.seq`' object is to return predictions for all values of λ, a and τ. For `qmult` this results in 2700 predictions for one row of predictors because there are 100 λ values, 3 a values and 9 quantiles being modeled.

```
R> allPreds <- predict(qmult,newData)
R> allPreds[,1:5] #Present the first five predictions
```

```
tau0.1a3 L1 tau0.1a3 L2 tau0.1a3 L3 tau0.1a3 L4 tau0.1a3 L5
-0.01338804 -0.01338804 -0.01338804 -0.01338804 -0.01338804
```

Following the `coef.rq.pen.seq` code presented in the previous section, users can specify predictions from specific values of λ, τ or a.

```
R> somePreds <- predict(qmult,newData, tau=c(.1,.5,.9),a=4,lambda=qmult$lambda[25])
R> somePreds
```

```
        tau0.1a4   tau0.5a4   tau0.9a4
[1,] -0.009712623 0.01760586 0.05166048
```

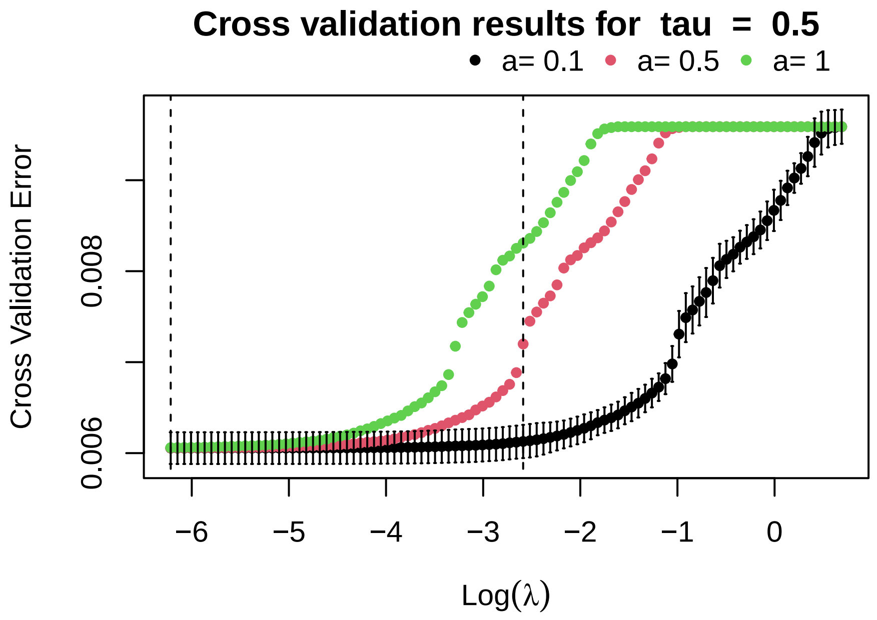
Cross-validation for tuning parameter selection

The function `rq.pen.cv()` creates a '`rq.pen.seq.cv`' object that can be used to select tuning parameters. Below we consider a similar model, but use elastic net instead of SCAD. To set the weights of (24) `tauWeights` is set to be $\tau(1-\tau)$, which provides more weights to error at the median and less to the more extreme quantiles.

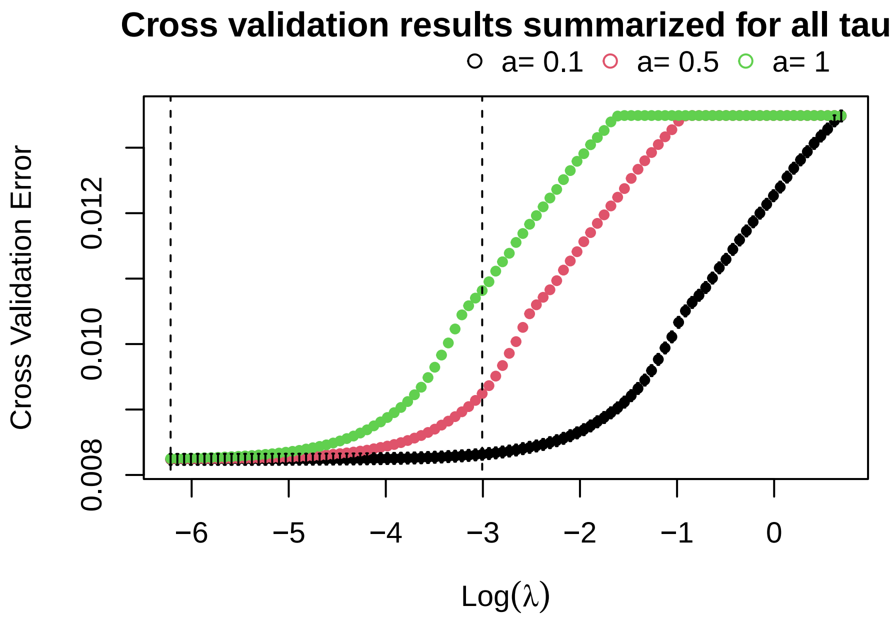
```
R> tauVals <- seq(.1,.9,.1)
R> qcv <- rq.pen.cv(x,y,tau=tauVals,a=c(.1,.5,1),penalty="ENet",
+                   tauWeights = tauVals*(1-tauVals))
```

The `plot()` function with a '`rq.pen.seq.cv`' object can create a plot of cross-validation error for a single quantile, (21), or across all quantiles, (24).

```
R> plot(qcv,tau=.5)
```



```
R> plot(qcv,septau=FALSE)
```



The first dashed line is at the value of λ that minimizes the cross-validation error, while the second dashed line is the value of λ that is one standard error above the minimum value.

For a 'rq.pen.seq.cv' object the predict() function relies on the coef() function. Below are four types of predictions that can easily provided, using the four combinations of septau and cvmin.

```
R> p1 <- predict(qcv,newData)
R> p2 <- predict(qcv,newData,septau = FALSE)
R> p3 <- predict(qcv,newData, cvmin=FALSE)
R> p4 <- predict(qcv,newData, septau=FALSE,cvmin=FALSE)
R> cbind(t(p1),t(p2),t(p3),t(p4))
```

```
                    [,1]        [,2]         [,3]         [,4]
tau0.1a0.1 0.0001875151 0.000187527 -0.002058590 -0.002283364
tau0.2a0.1 0.0050622767 0.005062277  0.005732399  0.005798122
tau0.3a0.1 0.0108911830 0.010891183  0.010762942  0.010785846
tau0.4a0.1 0.0141624668 0.014162467  0.014841379  0.014881769
tau0.5a0.1 0.0192869943 0.019286994  0.018918003  0.018917174
tau0.6a0.1 0.0243272552 0.024327255  0.023525021  0.023886754
tau0.7a0.1 0.0276171641 0.027617164  0.027683506  0.027828709
tau0.8a0.1 0.0327623472 0.032762347  0.032954817  0.032994521
tau0.9a0.1 0.0391546241 0.039154624  0.038912496  0.038578777
```

For cvMin=TRUE, the default setting, the model predictions are the same for septau=TRUE and septau=FALSE, because the optimal (λ, a) pair is the same for each quantile, and thus will be the optimal pair across all quantiles. However, there are some differences in the one standard error approach. The print() of a 'rq.pen.seq.cv' object provides information about the cross validation results.

```
R> qcv
```

```
Cross validation tuning parameter optimized for each quantile
   tau       minCv      lambda lambdaIndex  lambda1se lambda1seIndex   a
1: 0.1 0.002484347 0.002139852          99 0.04299461             56 0.1
2: 0.2 0.004125078 0.001995633         100 0.05683634             52 0.1
3: 0.3 0.005234138 0.001995633         100 0.06534799             50 0.1
4: 0.4 0.005882223 0.001995633         100 0.07513430             48 0.1
5: 0.5 0.006055034 0.001995633         100 0.07513430             48 0.1
6: 0.6 0.005801171 0.001995633         100 0.06094375             51 0.1
7: 0.7 0.005141526 0.001995633         100 0.06094375             51 0.1
8: 0.8 0.004020829 0.001995633         100 0.04299461             56 0.1
9: 0.9 0.002384353 0.001995633         100 0.03487425             59 0.1
           cvse modelsIndex nonzero nzse
1: 9.012905e-05           1      14   13
2: 1.167327e-04           4      14   13
3: 1.442722e-04           7      14   11
4: 1.704024e-04          10      14   11
```

```
5: 1.741708e-04          13      13   11
6: 1.406351e-04          16      14   12
7: 1.268395e-04          19      14   13
8: 9.010966e-05          22      14   14
9: 6.840411e-05          25      14   13

Cross validation tuning parameter optimized across all quantiles
   tau       minCv      lambda lambdaIndex  lambda1se lambda1seIndex   a
1: 0.1 0.008238191 0.001995633         100 0.04943335             54 0.1
2: 0.2 0.008238191 0.001995633         100 0.04943335             54 0.1
3: 0.3 0.008238191 0.001995633         100 0.04943335             54 0.1
4: 0.4 0.008238191 0.001995633         100 0.04943335             54 0.1
5: 0.5 0.008238191 0.001995633         100 0.04943335             54 0.1
6: 0.6 0.008238191 0.001995633         100 0.04943335             54 0.1
7: 0.7 0.008238191 0.001995633         100 0.04943335             54 0.1
8: 0.8 0.008238191 0.001995633         100 0.04943335             54 0.1
9: 0.9 0.008238191 0.001995633         100 0.04943335             54 0.1
           cvse modelsIndex nonzero nzse
1: 8.363941e-05           1      14   13
2: 8.363941e-05           4      14   13
3: 8.363941e-05           7      14   12
4: 8.363941e-05          10      14   11
5: 8.363941e-05          13      13   12
6: 8.363941e-05          16      14   12
7: 8.363941e-05          19      14   13
8: 8.363941e-05          22      14   13
9: 8.363941e-05          25      14   10
```

The table provides the following output for results optimized at each quantile, and across all quantiles.

- *tau*: The quantile being modeled.
- *minCv*: The value of the minimum cross validation error.
- *lambda*: The value of λ at the minimum cross-validation error.
- *lambdaIndex*: The position in the sequence for the preceding λ value.
- *lambda1se*: The λ value using the one standard error rule.
- *a*: The optimal value of a decided by the minimum cross validation error, and the same value is used for the one standard error λ value.
- *cvse*: The standard error of the cross validation for the values of (λ, a) that provide the smallest cross validation error.
- *modelsIndex*: The position of the corresponding '`rq.pen.seq`' object in the `$fit$models` list.

- *nonzero*: The number of nonzero coefficients at the minimum value.

- *nzse*: The number of nonzero coefficients at the one standard error rule.

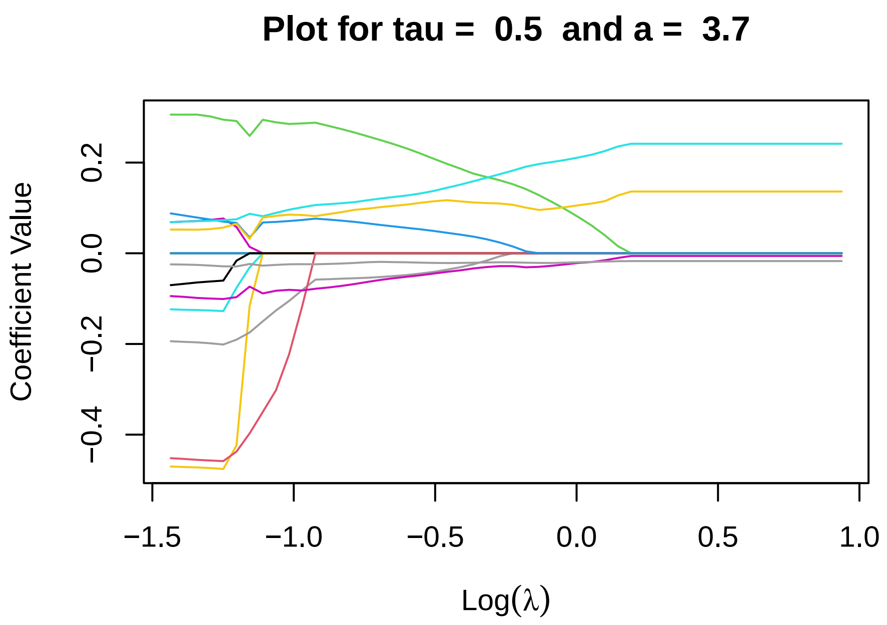
4.2. Group Penalty Example

The Ames Housing data contains many factor variables and is available in **AmesHousing**. For simplicity of presentation we consider only Lot_Shape, Garage_Type, Full_Bath, Fireplaces and Lot_Config as potential predictors. Some of these are categorical predictors, and thus a group penalty could be used to account for the group structure. The function `rq.group.pen()` creates an '`rq.pen.seq`' object, where the models are fit with a group penalty. In the below example, we fit a model for log sale price using these predictors, but do not penalize the Lot_Config variable. In addition, quantile specific penalties of $\tau(1-\tau)$ are used. Two '`rq.pen.seq`' objects are fit, one with $q=1$, `norm=1`, and the other with $q=2$, `norm=2`.

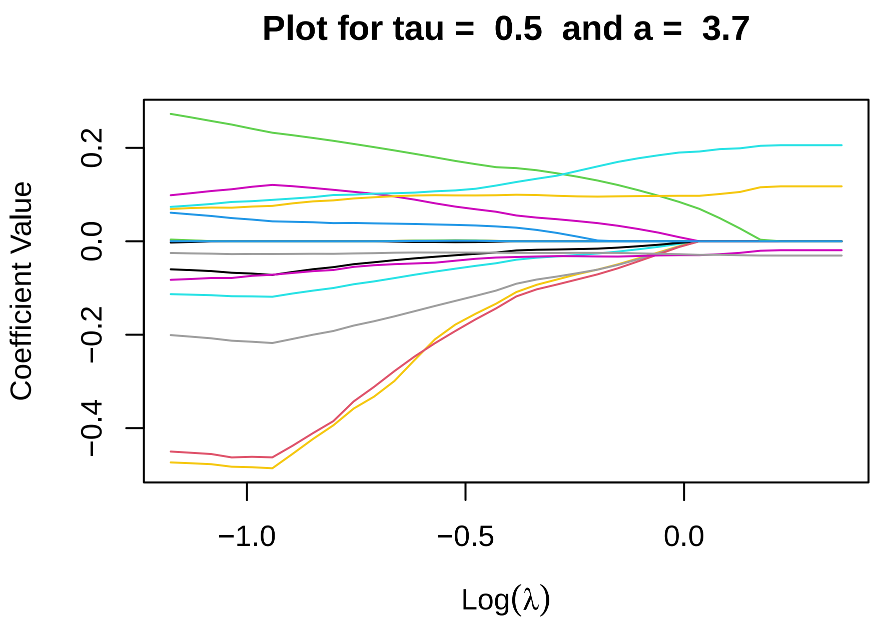
```
R> library(AmesHousing)
R> ames <- make_ames()
R> x_g <- cbind(model.matrix(~ Lot_Shape+Garage_Type+Full_Bath
+                            +Fireplaces+Lot_Config - 1,ames))
R> y_g <- log(ames$Sale_Price)
R> g <-  c(rep(1,4),rep(2,6),3,4,rep(5,4))
R> taus <- seq(.1,.9,.1)
R> qAmes1 <- rq.group.pen(x_g,y_g,groups=g, group.pen.factor = c(1,1,1,1,0), tau=taus,
+                        penalty="gSCAD",norm=1,tau.penalty.factor=taus*(1-taus))
R> qAmes2 <- rq.group.pen(x_g,y_g,groups=g, group.pen.factor = c(1,1,1,1,0), tau=taus,
+                        penalty="gSCAD",tau.penalty.factor=taus*(1-taus))
```

Plotting how the median model coefficient estimates change with τ provide two insights about `qAmes1` and `qAmes2`. First, for both models the unpenalized coefficients for `Lot_Config` are never set to zero. Second, the group coefficients for `qAmes2` go to zero as a group. While the grouped coefficients for `qAmes1` tend to go to zero as a group, but do not have to converge to zero at the same time.

```
R> plot(qAmes1,tau=.5)
```



```
R> plot(qAmes2,tau=.5)
```



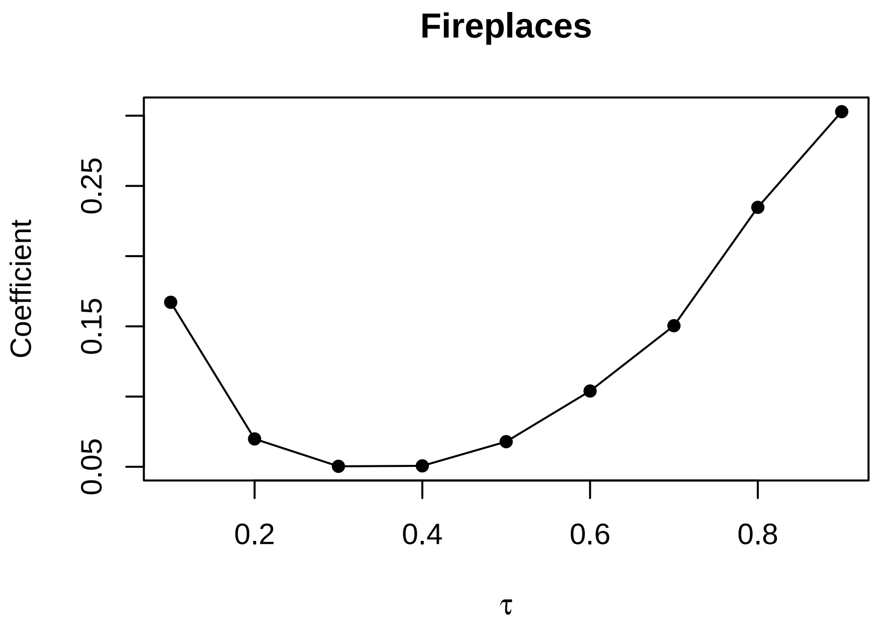
For instance, for $\tau = .5$ and using the 45th value of λ the garage coefficients from `qAmes1` provide a mix of zero and non-zero coefficients. This would not occur if the `norm=2`, the default.

```
R> coefficients(qAmes1,tau=.5, lambdaIndex=45)
```

```
NULL
```

The following plot presents how the coefficients for Fireplaces changes with τ at the 45th λ value.

```
R> bytau.plot(qAmes1,vars=13,lambdaIndex=45)
```

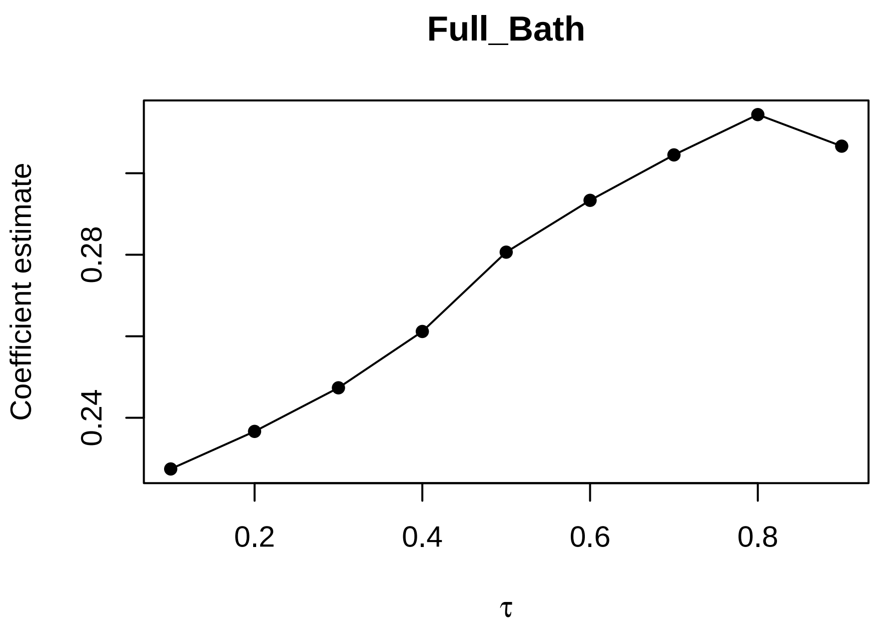


The `bytau.plot()` can also be used with a '`rq.pen.seq.cv`' object. The below code creates a '`rq.pen.seq.cv`' object using the group lasso penalty.

```
R> qgl <- rq.group.pen.cv(x_g,y_g,g,tau=tauVals)
```

Below code provides the plot for the coefficient of Full_Bath using the one standard error rule and the λ tuning parameter is selected to be optimal across all quantiles.

```
R> bytau.plot(qgl,vars=12,septau=FALSE,cvmin=FALSE)
```



Other functions such as `predict()`, `coef()` and `qic.select()` work the same for the group penalty methods as they do for the individual penalties, presented in the `barro` data set example. In fact, `rq.group.pen()` and `rq.pen()` both return '`rq.pen.seq`' objects and `rq.group.pen.cv` and `rq.pen.cv` both return '`rq.pen.seq.cv`' objects.

5. Conclusion

The **rqPen** provides tools for penalized estimators of quantile regression models. The package supports elastic-net, adaptive lasso, SCAD and MCP penalty functions. In addition users have access to group versions of lasso, adaptive lasso, SCAD and MCP penalties. It provides three different approaches for estimating these models: (1) linear programming; (2) QICD based approaches; and (3) Huber-like approximation of the quantile loss function with coordinate descent algorithms. The package is currently available on CRAN and github, see `https://github.com/bssherwood/rqpen` for the most recent updates.

References

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

Breheny P, Huang J (2009). "Penalized methods for bi-level variable selection." *Statistics and its Interface*, **2**, 369–380.

Breheny P, Huang J (2011). "Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection." *The Annals of Applied Statistics*, **5**(1), 232 – 253.

Breheny P, Huang J (2015). “Group descent algorithms for nonconvex penalized linear and logistic regression models with grouped predictors.” *Stat. Comput.*, **25**, 173–187.

De Cock D (2011). “Ames, Iowa: Alternative to the Boston housing data as an end of semester regression project.” *J. Stat. Educ.*, **19**(3).

Fan J, Li R (2001). “Variable selection via nonconcave penalized likelihood and its oracle properties.” *J. Amer. Statist. Assoc.*, **96**(456), 1348–1360.

Friedman J, Hastie T, Tibshirani R (2010). “Regularization Paths for Generalized Linear Models via Coordinate Descent.” *J. Stat. Softw.*, **33**(1), 1–22.

Gu Y, Fan J, Kong L, Ma S, Zou H (2018). “ADMM for High-Dimensional Sparse Penalized Quantile Regression.” *Technometrics*, **60**(3), 319–331.

He X, Pan X, Tan KM, Zhou WX (2023). “Smoothed quantile regression with large-scale inference.” *Journal of Econometrics*, **232**(2), 367–388. ISSN 0304-4076. doi:https://doi.org/10.1016/j.jeconom.2021.07.010. URL https://www.sciencedirect.com/science/article/pii/S0304407621001950.

Huang J, Breheny P, Ma S (2012). “A selective review of group selection in high-dimensional models.” *Statistical Science*, **27**(4), 481–499.

Huber PJ (1964). “Robust Estimation of a Location Parameter.” *Ann. Math. Statist.*, **35**(1), 73–101.

Kato K (2012). “Group Lasso for high dimensional sparse quantile regression models.” Https://arxiv.org/pdf/1103.1458.

Koenker R, Bassett G (1978). “Regression Quantiles.” *Econometrica*, **46**(1), 33–50.

Koenker R, D’Orey V (1994). “A Remark on Algorithm AS 229: Computing Dual Regression Quantiles and Regression Rank Score.” *J. R. Stat. Soc. Ser. C. Appl. Stat.*, **43**(2), 410–414.

Koenker R, Machado JAF (1999). “Goodness of Fit and Related Inference Processes for Quantile Regression.” *Journal of the American Statistical Association*, **94**(448), 1296–1310. doi:10.1080/01621459.1999.10473882. https://www.tandfonline.com/doi/pdf/10.1080/01621459.1999.10473882, URL https://www.tandfonline.com/doi/abs/10.1080/01621459.1999.10473882.

Koenker R, Mizera I (2014). “Convex Optimization in R.” *J. Stat. Softw.*, **60**(5), 1–23. ISSN 1548-7660.

Koenker RW, D’Orey V (1987). “Computing Regression Quantiles.” *J. R. Stat. Soc. Ser. C. Appl. Stat.*, **36**(3), 383–393.

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. URL http://www.jstor.org/stable/24247149.

Newey WK, Powell JL (1987). “Asymmetric Least Squares Estimation and Testing.” *Econometrica*, **55**(4), 819–847. ISSN 00129682, 14680262.

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.

Portnoy S, Koenker R (1997). "The Gaussian hare and the Laplacian tortoise: computability of squared-error versus absolute-error estimators." *Statist. Sci.*, **12**(4), 279–300.

Sherwood B, Li S (2022). "Quantile regression feature selection and estimation with grouped variables using Huber approximation." *Statistics and Computing*, **32**(5), 75. ISSN 1573-1375.

Sherwood B, Molstad AJ, Singha S (2020). "Asymptotic properties of concave L1-norm group penalties." *Statistics & Probability Letters*, **157**, 108631. ISSN 0167-7152.

Simon N, Friedman J, Hastie T, Tibshirani R (2013). "A Sparse-Group Lasso." *Journal of Computational and Graphical Statistics*, **22**(2), 231–245. doi:10.1080/10618600.2012.681250. https://doi.org/10.1080/10618600.2012.681250, URL https://doi.org/10.1080/10618600.2012.681250.

Tan KM, Wang L, Zhou WX (2022). "High-dimensional quantile regression: Convolution smoothing and concave regularization." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **84**(1), 205–233. doi:https://doi.org/10.1111/rssb.12485. https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/rssb.12485, URL https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/rssb.12485.

Tibshirani R (1996). "Regression Shrinkage and Selection via the Lasso." *J. R. Stat. Soc. Ser. B. Stat. Methodol.*, **58**(1), 267–288.

Tibshirani R, *et al.* (2012). "Strong rules for discarding predictors in lasso-type problems." *J. R. Stat. Soc. Ser. B. Stat. Methodol.*, **74**(2), 245–266.

Wang L, Chen G, Li H (2007). "Group SCAD regression analysis for microarray time course gene expression data." *Bioinformatics*, **23**(12), 1486–1494.

Wang L, Wu Y, Li R (2012). "Quantile regression of analyzing heterogeneity in ultra-high dimension." *J. Amer. Statist. Assoc.*, **107**(497), 214–222.

Yang Y, Zou H (2015). "A fast unified algorithm for solving group-lasso penalize learning problems." *Stat. Comput.*, **25**(6), 1129–1141. ISSN 1573-1375.

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.

Yu L, Lin N (2017). "ADMM for Penalized Quantile Regression in Big Data." *International Statistical Review*, **85**(3), 494–518.

Yu L, Lin N, Wang L (2017). "A Parallel Algorithm for Large-Scale Nonconvex Penalized Quantile Regression." *Journal of Computational and Graphical Statistics*, **26**(4), 935–939.

Yuan M, Lin Y (2005). "Model selection and estimation in regression with grouped variables." *J. R. Stat. Soc. Ser. B. Stat. Methodol.*, **68**(1), 49–67.

Zhang CH (2010). "Nearly unbiased variable selection under minimax concave penalty." *Ann. Statist.*, **38**(2), 894–942.

Zou H (2006). "The adaptive Lasso and its oracle properties." *J. Amer. Statist. Assoc.*, **101**(476), 1418–1429.

Zou H, Hastie T (2005). "Regularization and variable selection via the Elastic Net." *J. R. Stat. Soc. Ser. B. Stat. Methodol.*, **67**, 301–320.

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

Affiliation:

Ben Sherwood
School of Business
University of Kansas
1654 Naismith Drive
Lawrence, KS, USA 66045
E-mail: `ben.sherwood@ku.edu`
URL: `https://business.ku.edu/people/ben-sherwood`

Shaobo Li
School of Business
University of Kansas
1654 Naismith Drive
Lawrence, KS, USA 66045

Adam Maidman
School of Statistics
University of Minnesota
224 Church St SE
Minneapolis, MN