rqPen: An R package for penalized quantile regression

Ben Sherwood
University of Kansas

Shaobo Li
University of Kansas

Adam Maidman
Airbnb

Abstract

Quantile regression directly models a conditional quantile of interest. A wide variety of penalties have been introduced to improve regression estimation and many all provide simultaneous estimation and model selection. 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

? proposed quantile regression as robust alternative to mean regression that directly models a conditional quantile of interest without the need for assumptions about the distribution or variance of the error term. Since the seminal paper of (**?**), 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 (**?**), SCAD (**?**), MCP (**?**), adaptive lasso (**?**), group lasso (**?**) and other group extensions of the previously stated penalties (**???**), 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 (**?**) and group lasso (**?**) and deriving oracle properties for non-convex penalties such as SCAD and MCP (**?**). 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 (**??**). While a second order cone programming problem can be solved using convex optimization software, including the R package **Rmosek** (**?**).

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 **?** (elastic net), **?** (non-convex penalties), **?** (group non-convex penalties), and **?** (group lasso). **?** 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**. **?** proposed a coordinate descent algorithm (QICD) for nonconvex penalties that relies on an initial estimator being provided, implemented in **qicd**. Other approaches, albeit without active R packages, for penalized quantile regression include using ADMM algorithms (**???**).

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 and provides plots for how coefficient values change with the quantile being modeled, τ. The packages **quantreg** and **hqreg** are other alternative for penalized quantile regression in R. The package **rqPen** provides access to the same penalties these packages provide, plus additional penalties. For instance, neither of these packages include group penalties. In addition, **rqPen** provides more tools that are directly catered to penalized regression. Such as the previously mentioned trace plots and using information criterion for parameter selection. 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.

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 **?** 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

$$\frac{1}{n}\sum_{i=1}^{n}\rho_\tau(y_i-\mathbf{x}_i^\top\boldsymbol{\beta})+\lambda P_a(\boldsymbol{\beta}).$$

The penalty function $P_a(\boldsymbol{\beta})$ can take on the form of a group or individual penalty.

2.1. Individual Penalties

The package **rqPen** supports four different forms of individual penalties. 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 are the four different penalty functions, plus two important special cases.

1. Elastic net: $P_a(\boldsymbol{\beta})=\lambda\sum_{j=1}^p w_j\left[\alpha|\beta_j|+(1-\alpha)\beta_j^2\right]$, where $a\in[0,1]$.
   (a) LASSO: special case with $a=1$.
   (b) Ridge: special case with $a=0$.
2. Adaptive LASSO: $P_a(\boldsymbol{\beta})=\lambda\sum w_j|\tilde{\beta}_j|^{-a}|\beta_j|$, where $a>0$.
3. SCAD: $P_a(\boldsymbol{\beta})=\sum_{j=1}^p p^s_{w_j\lambda,a}(|\beta_j|)$, where $a>2$.
4. MCP: $P_a(\boldsymbol{\beta})=\sum_{j=1}^p p^m_{w_j\lambda,a}(|\beta_j|)$, 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 value $\tilde{\beta}_j$ is the coefficient of the Ridge estimator with the same value of λ. 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,

$$\frac{1}{n}\sum_{i=1}^{n}\rho_\tau(y_i-\mathbf{x}_i^\top\boldsymbol{\beta})+\sum_{j=1}^{p}\rho_\tau(\lambda w_j\beta_j)+\rho_\tau(-\lambda w_j\beta_j). \tag{3}$$

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 (3) is equivalent to minimizing

$$\frac{1}{n}\sum_{i=1}^{n+2p}\rho_\tau(\tilde{y}_i-\tilde{\mathbf{x}}_i^\top\boldsymbol{\beta}), \tag{4}$$

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 *method="lasso"* within **quantreg**. Below is code for fitting a lasso penalized estimator to the *barro* data set from **rqPen** using the "br" and "fn" algorithms from quantreg. Where the "br" implementation is described in **?** and **?**, while implementation of "fn" is decribed in **?**. Our experience is that "br" performs better and is more likely to provide a sparse solution.

```
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")
R> qfn <- rq.pen(x,y,alg="fn")
```

For large values of n and p the linear programming algorithms become computationally burdensome. To decrease computational complexity, **?** 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 **?** 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

$$h_\gamma^\tau(u) = h_\gamma(u) + (2\tau - 1)u, \tag{5}$$

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

$$\frac{1}{2n}\sum_{i=1}^{n} h_\gamma^\tau(y_i - \mathbf{x}_i^\top \boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} w_j|\beta_j|. \tag{6}$$

This implementation can be used in **rqPen** by the following code.

```
R> qhuber <- rq.pen(x,y,alg="huber")
```

Setting *alg="huber"* calls *hqreg::hqreg()*, and thus **hqreg** is another required package for **rqPen**. The Huber approximation is the default and will be used if the algorithm is not specified. The LASSO penalty provides the backbone for the three non-elastic net algorithms. 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 **?**. 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,

$$\sum_{j=1}^{p} p_{w_j\lambda,a}(|\beta_j|) \approx \sum_{j=1}^{p} p'_{w_j\lambda,a}(|\bar{\beta}_j|)|\beta_j|. \tag{7}$$

Again, the problem becomes a special case of a LASSO estimator with specific weights for each predictor. Thus all the non-elastic net penalties can be solved using the linear programming or the Huber approximation algorithms. **?** 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

$$\frac{1}{n}\sum_{i=1}^{n} \rho_\tau(y_i - \mathbf{x}_i^\top \boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} w_j \left[a|\beta_j| + (1-a)\beta_j^2\right], \tag{8}$$

cannot be framed as a linear programming problem because of the ridge penalty. Thus for $a \neq 1$, the **rqPen** implementation of elastic net uses the Huber approximation approach provided in **hqreg**. While **hqreg** provides a computational backbone, **rqPen** provides additional functions that will be described later. In addition the next section includes group penalties, that are not implemented in **hqreg**. For some group penalties **hqreg** will still provide a computational backbone, but for other penalties it will not be used.

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

$$\frac{1}{n}\sum_{i=1}^{n} \rho_\tau(y_i - \mathbf{x}_i^\top \boldsymbol{\beta}) + \sum_{g=1}^{G} p_{w_g\lambda,a}(||\boldsymbol{\beta}_g||_q). \tag{9}$$

Group penalties are implemented for four penalty functions: (1) LASSO; (2) Adaptive LASSO; (3) SCAD; and (4) MCP. Currently, there is no implementation of a group elastic net or ridge penalty. For the latter there does not exist a group version of the ridge penalty. For the former, that 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.

All four functions use the same form presented in the previous subsection, but the scalar β_j is replaced with the scalar $||\boldsymbol{\beta}_g||_q$. 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 $q = 1$ (9) is

$$\frac{1}{n}\sum_{i=1}^{n}\rho_\tau(y_i - \mathbf{x}_i^\top\boldsymbol{\beta}) + \sum_{g=1}^{G} w_g||\boldsymbol{\beta}_g||_1, \tag{10}$$

while for $q = 2$ it is

$$\frac{1}{n}\sum_{i=1}^{n}\rho_\tau(y_i - \mathbf{x}_i^\top\boldsymbol{\beta}) + \sum_{g=1}^{G} w_g||\boldsymbol{\beta}_g||_2. \tag{11}$$

Note that (10) is a special case of the individual LASSO penalty where the weights within each group are the same, while (11) is the standard group lasso penalty introduced by **?**. The estimator that minimizes (10) is not guaranteed to have variable selection agreement within a group, where this is guaranteed for the estimator that minimizes (11). Due to (10) 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 $q = 2$. These other three penalties are partially motivated by a desire to have the oracle property. However, for $q = 1$ it has been shown that the oracle property only holds at the group level and not at the individual level, although these results do not explore the non-differentiable quantile loss function or the adaptive lasso penalty (**?**). For an excellent review of group penalties, including a discussion of the use of $q = 1$ the reader is referred to **?**. **?** provides a more detailed exploration of group penalties with $q = 1$.

Choosing $q = 1$ for quantile regression estimators has the additional 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. Again let $\bar{\boldsymbol{\beta}}_g$ be an initial group LASSO estimator and the penalized objective function is approximated by

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

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** (**?**). 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 **?** is not for a group penalty. Instead the algorithm of **?** is implemented.

2.3. Estimation of Multiple Quantiles

The previous two subsections considers only one quantile, but, similar to **quantreg**, **rqPen** accommodates estimation of multiple quantiles. Consider B quantiles and let τ_b be the bth quantile of interest. Let $\boldsymbol{\beta}^{\tau_b}$ be the regression coefficients for the bth quantile. For individual penalties the estimator minimizes,

$$\frac{1}{n}\sum_{b=1}^{B}\sum_{i=1}^{n}\rho_\tau\left(y_i - \mathbf{x}_i^\top\boldsymbol{\beta}^{\tau_b}\right) + \sum_{b=1}^{B}\sum_{j=1}^{p} p_{w_j d_b \lambda, a}(|\beta_j^{\tau_b}|). \tag{13}$$

While for group penalties it is

$$\frac{1}{n}\sum_{b=1}^{B}\sum_{i=1}^{n}\rho_\tau\left(y_i - \mathbf{x}_i^\top\boldsymbol{\beta}^{\tau_b}\right) + \sum_{b=1}^{B}\sum_{g=1}^{G} p_{w_g d_b \lambda, a}(||\boldsymbol{\beta}_g||_q). \tag{14}$$

The main difference is the introduction of quantile specific weights for the penalty terms. **?** provide one example of specific weights for each quantiles and use $d_b = \sqrt{\tau_b(1-\tau_b)}/n$. Dividing by n is simply a difference in the scale of λ, but the $\sqrt{\tau_b(1-\tau_b)}$ term provides a quantile dependent weight. The optimization of (13) and (14) consist of B different optimization problems that have been discussed in Sections 2.1 and 2.2.

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). Users can also specify their own value of a or a sequence of potential values of a to be considered. 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 zero otherwise. 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

$$QIC(\tau,\lambda,a) = \log\left[\sum_{i=1}^{n}\rho_\tau(y_i - \mathbf{x}_i^\top\hat{\boldsymbol{\beta}}_{\lambda,a}^\tau)\right] + mk_{\lambda,a}^\tau, \tag{15}$$

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 **?** [$m = \log(n)\log(p)$]. Using the barro data set used previously, below is code for modeling the .25 quantile using elastic net and results for using both AIC and BIC to select the pair of (λ, a).

```
R> rqe <- rq.pen(x,y,tau=.25,penalty="ENet",a=seq(0,1,.1),alg="huber")
R> aicm <- qic.select(rqe, method="AIC")
R> #BIC is the default method
R> bicm <- qic.select(rqe)
R> #following returns the coefficients of the selected model
R> coefficients(bicm)

         tau=0.25
 [1,] -0.040226686
 [2,] -0.023893370
 [3,]  0.006519105
 [4,]  0.003225629
 [5,]  0.000000000
 [6,]  0.000000000
 [7,]  0.059605464
 [8,] -0.001406846
 [9,] -0.215479802
[10,]  0.090625321
[11,] -0.140849188
[12,] -0.025810735
[13,] -0.029461310
[14,]  0.079574971

R> #provides information about the selected model
R> bicm$modelsInfo

    tau modelIndex a   minQIC lambdaIndex      lambda
1: 0.25         11 1 6.388936         100 0.001960444

R> #provides IC values for all possible models
R> bicm$ic[1:5,1:5]

   tau0.25a0 tau0.25a0.1 tau0.25a0.2 tau0.25a0.3 tau0.25a0.4
L1  22.27022    10.87212    9.657005    8.883748    8.441887
L2  22.26040    10.87212    9.657005    8.883748    8.441887
L3  22.25236    10.87212    9.657005    8.883748    8.441887
L4  22.24375    10.87212    9.657005    8.883748    8.441887
L5  22.23495    10.87212    9.657005    8.883748    8.441887
```

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 (15). Define B weights of $(w_1, \ldots, w_B)$. The other approach is to select a single pair of (λ, a) that minimizes,

$$\sum_{b=1}^{B} w_b QIC(\tau_b, \lambda, a). \tag{16}$$

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. Below is code implementing both approaches. Note in this case the two approaches provide identical results, that is optimizing λ uniquely for each τ provides the same λ for all values of τ. However, this is not guranteed to happen.

```
R> #lasso model for 9 quantiles
R> tvals <- seq(.1,.9,.1)
R> rqmt <- rq.pen(x,y,tau=tvals)
R> #default is to find the optimal value of
R> # lambda for each quantile separately
R> rqmt_st <- qic.select(rqmt)
R> # alternative option is to find one value of
R> # lambda best for all tau. Below code
R> # also provides different weights for the models
R> rqmt_g <- qic.select(rqmt,septau = FALSE,weights=sqrt(tvals*(1-tvals)))
R> # first one will have different values of lambda for each quantile
R> # second approach will provide the same value for lambda for all quantiles
R> rqmt_st$modelsInfo
```

```
   tau modelIndex a    minQIC lambdaIndex      lambda
1: 0.1          1 1 11.202792         100 0.001995633
2: 0.2          2 1 13.345651         100 0.001995633
3: 0.3          3 1 14.039835         100 0.001995633
4: 0.4          4 1 14.254535         100 0.001995633
5: 0.5          5 1 14.190048         100 0.001995633
6: 0.6          6 1 14.935143         100 0.001995633
7: 0.7          7 1 14.262631         100 0.001995633
8: 0.8          8 1 12.989257         100 0.001995633
9: 0.9          9 1  9.611465         100 0.001995633
```

```
R> rqmt_g$modelsInfo
```

```
         modelIndex a tau    minQIC lambdaIndex      lambda
tau0.1a1          1 1 0.1 11.202792         100 0.001995633
tau0.2a1          2 1 0.2 13.345651         100 0.001995633
tau0.3a1          3 1 0.3 14.039835         100 0.001995633
tau0.4a1          4 1 0.4 14.254535         100 0.001995633
tau0.5a1          5 1 0.5 14.190048         100 0.001995633
tau0.6a1          6 1 0.6 14.935143         100 0.001995633
tau0.7a1          7 1 0.7 14.262631         100 0.001995633
```

```
tau0.8a1          8 1 0.8 12.989257         100 0.001995633
tau0.9a1          9 1 0.9  9.611465         100 0.001995633

R> #below command gets coefficients for the selected model optimized for each quantile
R> coefficients(rqmt_st)

          tau=0.1      tau=0.2      tau=0.3       tau=0.4
 [1,]  0.046024550 -0.025349510 -0.007628702 -0.0509099459
 [2,] -0.026284237 -0.024599504 -0.024662593 -0.0269258058
 [3,]  0.012328949  0.008081397  0.006480841  0.0064768621
 [4,]  0.002104772  0.005166707  0.002814568  0.0000000000
 [5,] -0.024627664 -0.025607695  0.007792513 -0.0006238768
 [6,]  0.009512346  0.008764757  0.000000000  0.0040800327
 [7,]  0.042289169  0.058181214  0.053186347  0.0687972724
 [8,] -0.001056290 -0.001910715 -0.001721006 -0.0009412972
 [9,] -0.362402003 -0.307690904 -0.094324088 -0.0775372750
[10,]  0.088922202  0.079340696  0.086755769  0.0830892163
[11,] -0.205922592 -0.167878400 -0.144565626 -0.1236879324
[12,] -0.024300727 -0.024496546 -0.029762941 -0.0285998471
[13,] -0.026985871 -0.031168065 -0.031536259 -0.0321213531
[14,]  0.108965090  0.113462718  0.090556363  0.1079694258
          tau=0.5      tau=0.6      tau=0.7      tau=0.8
 [1,] -0.030427445 -0.027667070 -0.036468569 -0.059369918
 [2,] -0.025921439 -0.027154054 -0.027049752 -0.029731119
 [3,]  0.010717433  0.012767446  0.020075468  0.020387034
 [4,]  0.000000000  0.000000000 -0.005597457 -0.007651144
 [5,]  0.000000000  0.004500418  0.000000000 -0.017975998
 [6,]  0.005652769  0.012718075  0.008847529  0.012003063
 [7,]  0.061545816  0.064704621  0.066847390  0.077466640
 [8,] -0.002032809 -0.003109405 -0.003002686 -0.002238763
 [9,] -0.053542693 -0.118189204 -0.097732410  0.099465610
[10,]  0.079649154  0.078695159  0.069455746  0.054449904
[11,] -0.096936333 -0.101068296 -0.098374165 -0.079401138
[12,] -0.026219888 -0.029832149 -0.027298837 -0.029925264
[13,] -0.029603500 -0.023199084 -0.017604570 -0.006516214
[14,]  0.150465797  0.193290866  0.203671322  0.217579350
          tau=0.9
 [1,] -0.031105032
 [2,] -0.032783674
 [3,]  0.019155652
 [4,] -0.005962290
 [5,] -0.001556920
 [6,]  0.004491901
 [7,]  0.078839376
 [8,] -0.002483301
 [9,] -0.047988988
[10,]  0.062508461
```

```
[11,] -0.090451700
[12,] -0.032571747
[13,]  0.003103054
[14,]  0.227296281

R> #code is the same for a group penalty
R> # This is not a great use of a group penalty and only for
R> # example
R> g <- c(rep(1,4),rep(2,3),rep(3,3),rep(4,3))
R> rqgroup <- rq.group.pen(x,y,groups=g,tau=seq(.1,.9,.1))
R> rqgroup_ic <- qic.select(rqgroup)
```

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 $\hat{\boldsymbol{\beta}}^{\tau_b}_{-k,\lambda,a}$ 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*, 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

$$C_k^\tau(\lambda, a) = \frac{1}{n_k} \sum_{i=1}^{n_k} \rho_\tau(y_i^k - \mathbf{x}_i^{k\top} \hat{\boldsymbol{\beta}}^{\tau}_{-k,\lambda,a}). \tag{17}$$

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

$$C^\tau(\lambda, a) = \frac{1}{K} \sum_{k=1}^{K} C_k^\tau(\lambda, a). \tag{18}$$

In addition it provides the λ that provides the sparsest solution that is within one standard error of the value that minimizes (18), with a fixed. The standard error is calculated as

$$\sqrt{\frac{1}{K-1} \sum_{k=1}^{K} \left[C_k^\tau(\lambda, a) - C^\tau(\lambda, a)\right]^2}, \tag{19}$$

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

$$C(\lambda, a) = \sum_{b=1}^{B} w_b C^{\tau_b}(\lambda, a). \tag{20}$$

Thus, again, providing the option to optimize the tuning parameters jointly or separately across quantiles.

By setting *groupError=FALSE* each cross-validation error is weighted equally and (18) and (20) are replaced by

$$\sum_{k=1}^{K} n_k C_k^\tau(\lambda, a), \tag{21}$$

and

$$\sum_{b=1}^{B} w_b \sum_{k=1}^{K} n_k C_k^{\tau}(\lambda, a), \tag{22}$$

respectively.

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 **?**, 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

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

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 (**?**). 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. Our experience has been that the automated methods work well. The cautious user can specify a fixed value of γ for `rq.group.pen`, but **hqreg** does not offer this flexibility and thus only the automatic values of γ are used for `rq.pen`.

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. **?** 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 down in the package **hqreg**, and details can be found in **?**. A similar approach has been used for the group penalties.

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`' 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.

Below are descriptions of `rq.pen` and `rq.pen.cv` and their arguments. These functions are very similar to `rq.group.pen` and `rq.group.pen.cv`. In addition, some of the arguments in `rq.pen.cv` are simliar to those of `qic.select`. The next section includes examples of using all of these functions.

Below is the `rq.pen` function, followed by its arguments.

```
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,...)
```

`x`	Matrix of predictor values.
`y`	Vector of response.
`tau`	Vector of quantiles being modeled, default is to only model the median.
`lambda`	Vector of lambda values. If not specified then a sequence will be automatically generated based on the data and rules described in previous section.
`penalty`	Penalty used, choices are LASSO, Ridge, elastic-net ("ENet"), adaptive LASSO ("aLASSO"), SCAD or MCP.

`a` Additional tuning parameter, not used for lasso or ridge penalties. However, will be set to the elastic net values of 1 and 0 respectively. Defaults are ENet(0), aLASSO(1), SCAD(3.7) and MCP(3).

`nlambda` The number of lambdas generated. This is ignored if lambda is set manually.

`eps` Multiplied by $\lambda_{\max}$ to define $\lambda_{\min}$ when the sequence of lambda values is generated automatically. If $n < p$ it defaults to .05, otherwise it defaults to .01.

`penalty.factor` Weights for the penalty for each predictor, the value w_j from (13). Default is to provide a value of one for each predictor, thus weighting them all equally.

`alg` The algorithm used. Default is "huber", the algorithm described in **?**. Linear programming solutions can be obtained using "br" or "fn" corresponding to the same algorithms available in `quantreg::rq`. Our experience has shown "br" to perform better than "fn" because it tends to provide a sparse solution, but it can be slow. The "QICD" algorithm is available for MCP or SCAD. For elastic net, Ridge or adaptive LASSO the choice must be "huber".

`scalex` If set to TRUE, the default, the x matrix will be centered and scaled to have mean zero and standard deviation of one. Coefficients are returned on the original scale of the data.

`tau.penalty.factor` Weights for the coefficients of each quantile, the value d_b from (13). Default provides equal weight, of one, for each quantile

`coef.cutoff` Some of the linear programming algorithms will provide very small, but not sparse solutions. Estimates below this number will be set to zero. This is ignored if a non-linear programming algorithm is used.

`max.iter` Maximum number of iterations of non-linear programming algorithms.

`converge.eps` Convergence threshold for non-linear programming algorithms.

`lambda.discard` Algorithm may stop for small values of lambda if the coefficient estimates are not changing drastically. One example of this is it is possible for the LLA weights of the non-convex functions, see (7), to all become zero and smaller values of lambda are extremely likely to produce the same zero weights.

Below is the `rq.pen.cv` followed by its arguments.

```
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, ...)
```

`x` Matrix of predictor values.

`y` Vector of response.

`tau` Vector of quantiles being modeled, default is to only model the median.

`lambda` Vector of lambda values. If not specified then a sequence will be automatically generated based on the data and rules described in previous section.

`penalty` Penalty used, choices are LASSO, Ridge, elastic-net ("ENet"), adaptive LASSO ("aLASSO"), SCAD or MCP.

`a` Additional tuning parameter, not used for lasso or ridge penalties. However, will be set to the elastic net values of 1 and 0 respectively. Defaults are ENet(0), aLASSO(1), SCAD(3.7) and MCP(3).

`cvFunc` Loss function for cross-validation. Defaults to quantile loss, but user can specify their own function.

`folds` Number of folds, K, used for K-folds cross validation.

`foldid` A vector of length n of fold assignments. If this is set, then `folds` is ignored.

`nlambda` The number of lambdas generated. This is ignored if lambda is set manually.

`groupError` If true then tuning parameters the by tau results, btr, are found to minimize (**??**).

`...` Additional arguments passed to `rq.pen`.

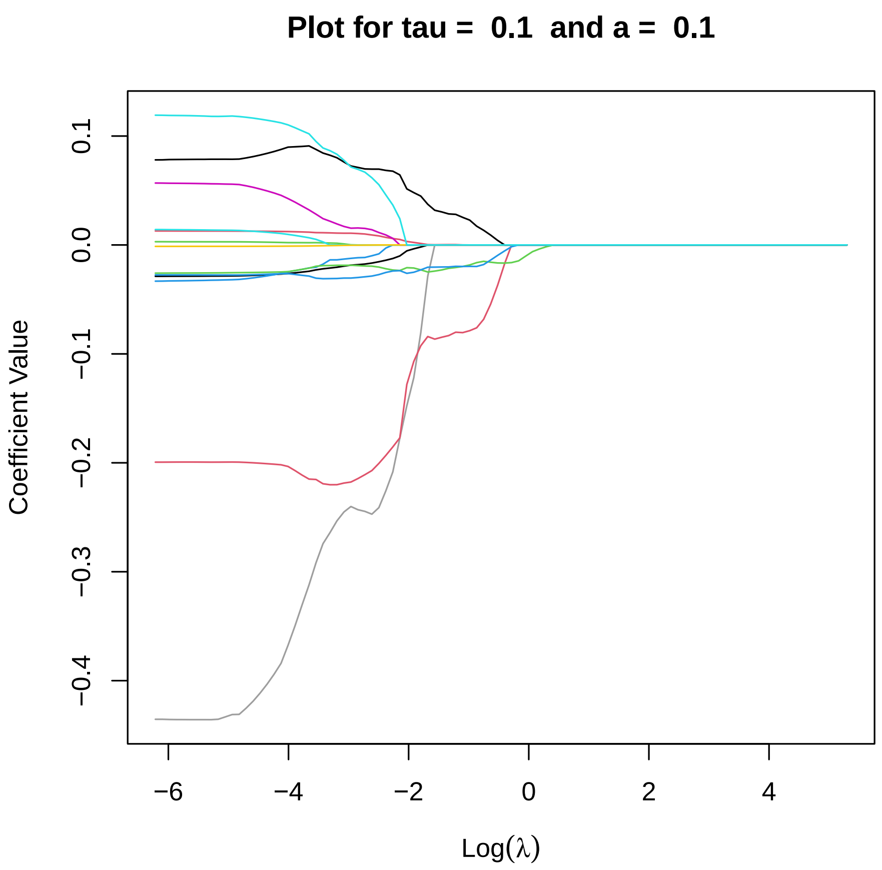
4. Applications

This section details fitting using the aforementioned barro data set and the Ames housing data (**?**).

4.1. barro

The barro data set, available in **quantreg**, contains GPD growth rates and 13 other potential explanatory variables. See **?** for more details. The following will provide code on fitting an elastic net model with potential 11 potential a values from 0 to 1 and equally spaced out by .1. Similarly, we will model 9 quantiles equally spaced out from .1 to .9. The below code fits models for the 9 different models, automatically generates a sequence of λ and considers the 11 potential a values using the Huber approximation algorithm.

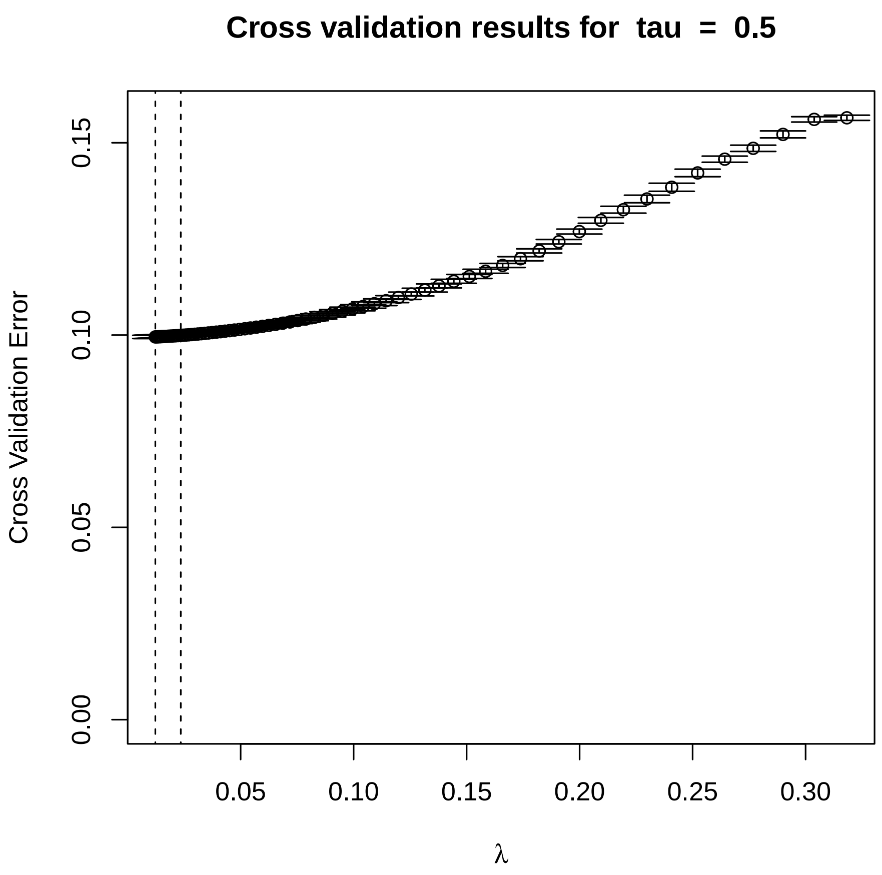
```
R> quants <- seq(.1,.9,.1)
R> r1 <- rq.pen(x,y,a=seq(0,1,.1),tau=quants,penalty="ENet",alg="huber")
R> #Below plots the coefficient values for .1 quantile with a=.1
R> plot(r1,a=.1,tau=.1,logLambda = TRUE)
R> #Below code would provide plots for all possible values of a and tau
R> #plot(r1,logLambda=TRUE)
```



4.2. Group Penalty Example

The Ames Housing data contains many factor variables and is available in **AmesHousing**. Below is an example of cross validation code used to fit a model for the median using the group lasso penalty. The plot demonstrates the cross-validation error as a function of λ.

```
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> r2 <- rq.group.pen.cv(x_g,y_g,groups=g)
R> plot(r2)
```



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 for both individual and group penalties, though for group penalties only the special case of LASSO is available for elastic-net. 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.

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

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