Sankhyā : The Indian Journal of Statistics
Special Issue on Quantile Regression and Related Methods
2005, Volume 67, Part 2, pp 399-417

Computational Issues for Quantile Regression

Colin Chen
SAS Institute Inc. Cary, NC, USA
Ying Wei
Columbia University, NY, USA

Abstract

In this paper, we discuss some practical computational issues for quantile regression. We consider the computation from two aspects: estimation and inference. For estimation, we cover three algorithms: simplex, interior point, and smoothing. We describe and compare these algorithms, then discuss implementation of some computing techniques, which include optimization, parallelization, and sparse computation, with these algorithms in practice. For inference, we focus on confidence intervals. We discuss three methods: sparsity, rank-score, and resampling. Their performances are compared for data sets with a large number of covariates.

AMS (2000) *subject classification.* Primary 62F35; secondary 62J99.
Keywords and phrases. Quantile regression, host optimization, multithreading, sparse computing, smoothing algorithm, simplex, interior point, median regression, linear programming, preprocessing.

1 Introduction

Quantile regression is becoming more and more useful, not only in econometrics, but also in finance, biomedicine, data mining, and environmental studies. To make this technique more generally applicable, reliable, optimal, and efficient computational tools are needed. While the development of advanced algorithms is essential, the state of the art lies in their implementation with modern computing techniques. One purpose of this paper is to evaluate the most advanced algorithms in combination with modern computational techniques.

For any quantile $\tau \in (0,1)$, the regression quantile $\hat{\beta}(\tau)$ is defined as

$$\hat{\beta}(\tau) = \underset{\beta \in \boldsymbol{R}^p}{\operatorname{argmin}} \sum_{i=1}^{n} \rho_\tau(y_i - x_i'\beta), \tag{1.1}$$

where $\{y_1, ..., y_n\}$ is a random sample of the response variable Y, $x_i \in \boldsymbol{R}^p$ is the covariate vector corresponding to the ith observation y_i, and $\rho_\tau(z) = z(\tau - I(z < 0))$, $0 < \tau < 1$, is the check function for the τth quantile. In the special case of $\tau = 1/2$, which corresponds to the median, $\hat{\beta}(1/2)$ is known as the L_1 estimator.

Since the early 1950's it has been recognized that the L_1 minimization can be formulated as a linear programming (LP) problem and solved efficiently with some form of the simplex algorithm. In the next section, we present such a formulation. An efficient version of the simplex algorithm was developed by Barrodale and Roberts (1974). This special version for the L_1 estimator can be naturally extended to compute regression quantiles, even the entire quantile process (Koenker and d'Orey 1993). This algorithm is found slow for data sets with a large number of observations ($n > 100000$). Another algorithm known as the interior point algorithm, which also solves general linear programming problems, was introduced into quantile regression by Portnoy and Koenker (1997). This algorithm shows great advantages in computation efficiency over the simplex algorithm for data sets with a large number of observations.

The algorithms developed for a general LP problem may not fully deploy the properties of the original L_1 or quantile regression and have their own shortcomings. For example, the interior point algorithm uses a full factorization in each iteration, which is computationally expensive with a large number of covariates. Besides, this algorithm can only give the approximate solutions of the original problem and rounding has to been done if one requires the same accuracy as that of the simplex algorithm. For certain problems, this rounding step requires some significant extra computing time. In these cases, some heuristic approaches demonstrate advantages on both speed and accuracy. One of them is the finite smoothing algorithm (Chen 2003).

These algorithms represent the most advanced algorithms for computing regression quantiles. However, to make them reliable, optimal, and efficient, they must be implemented with some modern computing techniques. Optimization is always the first step to improve performance in scientific computing. While some of the optimization procedures are more related to the computing environments, which may not be controlled by a programmer, such as cache, I/O, and system environments, some optimization techniques can simply be implemented with proper programming and significantly improve the performance. When multithreads are available with the computing resource, parallelization should be considered to speed up the intensive computing, which is common in quantile regression. Therefore, scalability of the

three algorithms with multithreads is explored using some empirical results.

Statistical inference on regression quantiles is important in most applications. Confidence intervals of regression quantiles are the most commonly used. Various methods of constructing confidence intervals have been developed in recent years. Using different models, Kocherginsky et al. (2005) did a detailed comparison of these methods based on the coverage probability. However, the models they selected are restricted with a small number of covariates. In many applications, like survey data analysis and semiparametric longitudinal studies, the number of covariates could be considerably large. As the number of covariates increases, the underlying asymptotic results might be questionable, which could result in poor confidence interval estimates. Therefore, it is helpful to assess the performances of these confidence interval methods when a large number of covariates exist. In this paper, we focus on some practical computational issues of these confidence interval methods, including their performance in the existence of a large number of covariates, and their potential improvement of implementing computing techniques of optimization and parallelization.

Section 2 introduces the three algorithms for computing regression quantiles and shows how those modern computing techniques can be implemented in each individual algorithm. Section 3 describes the three recommended methods for computing confidence intervals. Their performances with a large number of covariates are demonstrated. Some other computational issues are discussed in Section 4.

2 Computing Regression Quantiles

AN LP PROBLEM

Let $\mu = [y - X\beta]_+$, $\nu = [X\beta - y]_+$, $\phi = [\beta]_+$, and $\varphi = [-\beta]_+$, where $y = (y_1, ..., y_n)^{'}$, $X = (x_1, ..., x_n)^{'}$, and $[z]_+$ is the nonnegative part of z.

Let $D_{LAR}(\beta) = \sum_{i=1}^{n} |y_i - x_i^{'}\beta|$ and $D_{\rho_\tau}(\beta) = \sum_{i=1}^{n} \rho_\tau(y_i - x_i^{'}\beta)$. The L_1 problem, $\min_\theta D_{LAR}(\theta)$, can be reformulated as

$$\min_{\beta}\{e^{'}\mu + e^{'}\nu | y = X\beta + \mu - \nu, \{\mu, \nu\} \in \boldsymbol{R}_+^n\}, \tag{2.1}$$

where e denotes an n-vector of ones.

Let $B = [X \ -X \ I \ -I]$, $\theta = (\phi^{'} \ \varphi^{'} \ \mu^{'} \ \nu^{'})^{'}$, and $d = (\mathbf{0}^{'} \ \mathbf{0}^{'} \ e^{'} \ e^{'})^{'}$ where

$\mathbf{0}^{'} = (0\ 0\ ...\ 0)_p$. The reformulation presents a standard LP problem:

$$\begin{aligned} (P) \qquad & \min_{\theta} d^{'}\theta \\ \text{subject to} \quad & B\theta = y \\ & \theta \geq 0. \end{aligned}$$

This problem has the dual formulation

$$\begin{aligned} (D) \qquad & \max_{d} y^{'}z \\ \text{subject to} \quad & B^{'}z \leq d, \end{aligned}$$

which can be simplified as

$$\max_{z}\{y^{'}z | X^{'}z = 0, z \in [-1,1]^n\}.$$

By setting $\eta = \frac{1}{2}z + \frac{1}{2}e$, $b = \frac{1}{2}X^{'}e$, it becomes

$$\max_{\eta}\{y^{'}\eta | X^{'}\eta = b, \eta \in [0,1]^n\}. \tag{2.2}$$

For quantile regression, the minimization problem is $\min_{\beta}\sum \rho_{\tau}(y_i - x_i^{'}\beta)$, and a similar set of steps lead to the dual formulation:

$$\max_{z}\{y^{'}z | X^{'}z = (1-\tau)X^{'}e, z \in [0,1]^n\}. \tag{2.3}$$

2.1. Simplex. The special version of the simplex algorithm developed by Barrodale and Roberts (1974) solves the primary LP problem (P) by two stages, which exploit the special structure of the coefficient matrix B. The first stage only picks the columns in X or $-X$ as pivotal columns. The second stage only interchanges the columns in I or $-I$ as basis or nonbasis columns. The algorithm obtains an optimal solution by executing these two stages interactively. Moreover, because of the special structure of B, only the main data matrix X is stored in the current memory.

Although the simplex algorithm is regarded as computationally demanding for large data sets, a careful and proper coding still makes it suitable for the data sets with less than 5000 observations and 50 variables given the present popular hardware (e.g. 2GHz CPU and 512MB of RAM).

2.2. Interior Point. To solve large to huge LP problem, alternative algorithms have been developed. Rather than moving from vertex to vertex

around the outer surface of the constraint set as dictated by the simplex, the interior point approach of Karmarkar (1984) solves a sequence of quadratic problems in which the relevant interior of the constraint set is approximated by an ellipsoid. The worst-case performance of the interior point algorithm has been proved to be better than that of the simplex algorithm.

The excellent paper by Portnoy and Koenker (1997) revives the enthusiasm of applying L_1 regression or quantile regression to large or huge data sets by introducing the faster interior point algorithm into this area.

There are many variations of interior point algorithms. The most popularly used algorithm for L_1 regression or quantile regression is the Primal-Dual with Predictor-Corrector algorithm.

Let $c = y$, $b = (1-\tau)X^{'}e$, and $A = X^{'}$, the dual problem (2.3) with a general upper bound u is

$$\begin{aligned} & \max\{c^{'}z\} \\ \text{subject to} \quad & Az = b \\ & 0 \le z \le u. \end{aligned}$$

To solve this LP problem, $0 \le z \le u$ is split into $z \ge 0$ and $z \le u$. Let v be the primal slack so that $z + v = u$, and associate dual variables w with these constraints. The Interior Point solves the system of equations to satisfy the Karush-Kuhn-Tucker (KKT) conditions for optimality:

$$\begin{aligned} \text{(KKT)} \quad & Az = b \\ & z + v = u \\ & A^{'}t + s - w = c \\ & ZSe = 0 \\ & VWe = 0 \\ & z, s, v, w \ge 0, \end{aligned}$$

where $Z = diag(z)$, (that is, $Z_{i,j} = z_i$ if $i = j$, $Z_{i,j} = 0$ otherwise), $S = diag(s)$, $W = diag(w)$, $V = diag(v)$.

These are the conditions for feasibility, with the *complementarity* conditions $ZSe = 0$ and $VWe = 0$ added. $c^{'}z = b^{'}t - u^{'}w$ must occur at the optimum. Complementarity forces the optimal objectives of the primal and dual to be equal, $c^{'}z_{opt} = b^{'}t_{opt} - u^{'}w_{opt}$.

The interior point algorithm works by iteratively using Newton's method to find a direction $(\Delta z^k, \Delta t^k, \Delta s^k, \Delta v^k, \Delta w^k)$ to move from the current solution $(z^k, t^k, s^k, v^k, w^k)$ toward a better solution.

To do this, two steps are used. The first step is called an *affine* step, which solves a linear system using Newton's method to find a direction (Δz^k_{aff}, Δt^k_{aff}, Δs^k_{aff}, Δv^k_{aff}, Δw^k_{aff}) to reduce the complementarity toward zero. The second step is called a *centering* step, which solves another linear system to determine a centering vector (Δz^k_c, Δt^k_c, Δs^k_c, Δv^k_c, Δw^k_c) to further reduce the complementarity. The centering step may not reduce too much of the complementarity; however, it builds up the *central path* and makes substantial progress toward the optimum in the next iteration. With these two steps, then

$$\begin{aligned}(\Delta z^k, \Delta t^k, \Delta s^k, \Delta v^k, \Delta w^k) &= (\Delta z_{aff}, \Delta t_{aff}, \Delta s_{aff}, \Delta v_{aff}, \Delta w_{aff}) \\ &+ (\Delta z_c, \Delta t_c, \Delta s_c, \Delta v_c, \Delta w_c), \end{aligned} \tag{2.4}$$

$$\begin{aligned}(z^{k+1}, t^{k+1}, s^{k+1}, v^{k+1}, w^{k+1}) &= (z^k, t^k, s^k, v^k, w^k) \\ &+ \alpha(\Delta z^k, \Delta t^k, \Delta s^k, \Delta v^k, \Delta w^k), \end{aligned} \tag{2.5}$$

where, α is the *step length* assigned a value as large as possible but not so large that a z^{k+1}_i, s^{k+1}_i, v^{k+1}_i, or w^{k+1}_i is "too close" to zero.

Although the Predictor-Corrector variant entails solving two linear systems instead of one, fewer iterations are usually required to reach the optimum. In both the affine step and the centering step, factorization of the $(X'[\Theta^k]^{-1}X)$ matrix, where Θ^k is a diagonal matrix computed in kth iteration, takes the majority computing time when solving the linear systems. However, the additional overhead of calculating the second linear system is small, as the factorization of the $(X'[\Theta^k]^{-1}X)$ matrix has already been performed to solve the first linear system. We refer to Wright (1996) for more details about this algorithm.

2.3. Smoothing. The finite smoothing algorithm was used by Clark and Osborne (1986), Madsen and Nielsen (1993) for the L_1 regression. It can be naturally extended to compute regression quantiles. In the following we briefly describe this algorithm. More details can be found in Chen (2003).

The non-differentiable

$$D_{\rho_\tau}(\beta) = \sum_{i=1}^{n} \rho_\tau(y_i - x_i'\beta) \tag{2.6}$$

can be approximated by the smooth function

$$D_{\gamma,\tau}(\beta) = \sum_{i=1}^{n} H_{\gamma,\tau}(r_i(\beta)), \tag{2.7}$$

where $r_i(\beta) = y_i - x_i'\beta$ and

$$H_{\gamma,\tau}(t) = \begin{cases} t(\tau-1) - \frac{1}{2}(\tau-1)^2\gamma & \text{if } t \le (\tau-1)\gamma \\ \frac{t^2}{2}\gamma & \text{if } (\tau-1)\gamma \le t \le \tau\gamma \\ t\tau - \frac{1}{2}\tau^2\gamma & \text{if } t \ge \tau\gamma \end{cases}$$

See Figure 1 for a plot of $H_{\gamma,\tau}$ and ρ_τ.

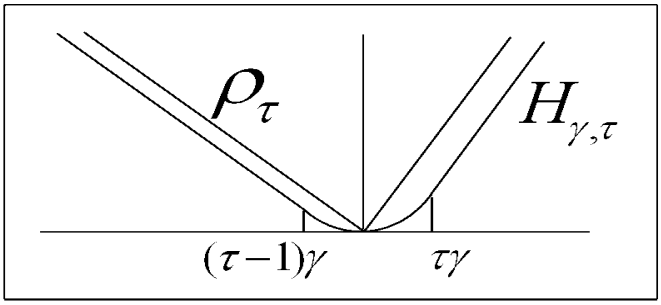

Figure 1. Objective Functions $H_{\gamma,\tau}$ and ρ_τ

The function $H_{\gamma,\tau}$ is determined by whether $r_i(\beta) \le (\tau-1)\gamma$, $r_i(\beta) \ge \tau\gamma$, or $(\tau-1)\gamma \le r_i(\beta) \le \tau\gamma$. These inequalities divide $\boldsymbol{R}^p$ into subregions separated by the parallel hyperplanes $r_i(\beta) = (\tau-1)\gamma$ and $r_i(\beta) = \tau\gamma$. The set of all such hyperplanes is denoted by $B_{\gamma,\tau}$:

$$B_{\gamma,\tau} = \{\beta \in \boldsymbol{R}^p | \exists i : r_i(\beta) = (\tau-1)\gamma \text{ or } r_i(\beta) = \tau\gamma\}. \tag{2.8}$$

Define the sign vector $s_{\gamma,\tau}(\beta) = (s_1(\beta), ..., s_n(\beta))'$ by

$$s_i = s_i(\beta) = \begin{cases} -1 & \text{if } r_i(\beta) \le (\tau-1)\gamma \\ 0 & \text{if } (\tau-1)\gamma \le r_i(\beta) \le \tau\gamma \\ 1 & \text{if } r_i(\beta) \ge \tau\gamma \end{cases}$$

and introduce $w_i = w_i(\beta) = 1 - s_i^2(\beta)$. Thus,

$$D_{\gamma,\tau}(\beta) = \frac{1}{2}\gamma r' W_{\gamma,\tau} r + v'(s) r + c(s), \tag{2.9}$$

where $W_{\gamma,\tau}$ is the diagonal n by n matrix with diagonal elements $w_i(\beta)$, $v'(s) = (s_1((2\tau-1)s_1+1)/2, ..., s_n((2\tau-1)s_n+1)/2)$, $c(s) = \sum[\frac{1}{4}(1-2\tau)\gamma s_i - \frac{1}{4}s_i^2(1-2\tau+2\tau^2)\gamma]$, and $r(\beta) = (r_1(\beta), ..., r_n(\beta))'$.

The gradient of $D_{\gamma,\tau}$ is given by

$$D^{(1)}_{\gamma,\tau}(\beta) = -X' \left[\frac{1}{\gamma} W_{\gamma,\tau}(\beta) r(\beta) + g(s)\right] \tag{2.10}$$

and for $\beta \in \boldsymbol{R}^p \backslash B_{\gamma,\tau}$ the Hessian exists and is given by

$$D^{(2)}_{\gamma,\tau}(\beta) = \frac{1}{\gamma} X' W_{\gamma,\tau}(\beta) X. \tag{2.11}$$

The gradient is a continuous function in $\boldsymbol{R}^p$, whereas the Hessian is piecewise constant.

The smoothing algorithm for minimizing D_{ρ_τ} is based on minimizing $D_{\gamma,\tau}$ for a set of decreasing γ. The essential advantage of the smoothing algorithm is that the solution $\beta_{0,\tau}$ can be detected when $\gamma > 0$ is small enough, i.e., it is not necessary to let γ converge to zero in order to find a minimizer of D_{ρ_τ}. For every new value of γ, information from the previous solution is utilized. The algorithm stops before going through the whole sequence of γ, which is generated by the algorithm itself. The convergence is indicated by no change of the status while γ goes through this sequence. Therefore, the algorithm is usually called the finite smoothing algorithm.

For a given threshold γ, a modified Newton iteration is used. Starting from an initial estimator $\beta_\tau(\gamma)$, the search direction $\mathbf{h}$ is found by solving the equation

$$D^{(2)}_{\gamma,\tau}(\beta_\tau(\gamma))\mathbf{h} \quad = \quad D^{(1)}_{\gamma,\tau}(\beta_\tau(\gamma)). \tag{2.12}$$

If $D^{(2)}_{\gamma,\tau}$ is full rank, $\mathbf{h}$ is the solution to (2.12). Otherwise, if the system (2.12) is consistent, then a consistent basic solution is used. If the system (2.12) is not consistent, a Marquardt perturbation is used to $D^{(2)}_{\gamma,\tau}$. After $\mathbf{h}$ is computed, if $s_\gamma(\beta_\tau(\gamma) + \mathbf{h}) = s_\gamma(\beta_\tau(\gamma))$, then $\beta_\tau(\gamma) + \mathbf{h}$ minimizes $D_{\gamma,\tau}$, otherwise a line search is performed.

Another advantage of the smoothing algorithm is that, when solving (2.12) for a new threshold γ', the factorization only needs a partial update instead of a full update as in the interior point algorithm. This saves time with large p.

Each of the previous three algorithms has its own advantages. None of them can fully dominate the others. Although the simplex algorithm is slow with a large number of observations, it is the most stable of the algorithms. For various kinds of data, especially data with a large portion of outliers and leverage points, the simplex algorithm always find a solution, while the other two algorithms might fail with floating point errors. The interior point algorithm is very fast for *slender* data sets, which have a large number of observations and a small number of covariates. The algorithm has a simple structure and might be easily adopted to other situations, e.g., constrained quantile regression. The finite smoothing algorithm is simple in theory for

quantile regression, and has the advantage in computing speed with a large number of covariates. Based on these facts, an adaptive algorithm by combining these three algorithms was suggested by Chen (2004) for practical purposes. To further improve these algorithms, we explore implementing some modern computing techniques in these algorithms in the following.

2.4. Optimization. Performance comparison of different algorithms is never easy because of the programming optimization, which includes both coding and host optimizations. Even with the same algorithm, optimized programming could show a significant improvement in performance.

Coding optimization eliminates the codes that might keep compiler optimizer from generating optimal codes. The following approaches may be considered:

a. Remove I/O from intensive loops.

b. Optimize or remove subprogram calls in intensive loops.

c. Eliminate complicated conditional operations in loops.

Most operating systems support basic computational routines. The library of these routines can be optimized with different hosts with respect to their special hardware and software environments. The improvement from using these optimized routines can be significant. For example, the interior point and smoothing algorithms use a lot of inner and cross products. Using optimized routines for these two kinds of products significantly improves these algorithms. In our implementation, using the optimized product routines improves the the computing speed by over three times with the interior point algorithm and over twice with the smoothing algorithm.

2.5. Parallelization. For computationally intensive problems, parallelization should always be considered when multiple processors are available. By Ahmdal's law, the speedup ratio depends on the percentage of parallelizable codes in the algorithm. In practice, we should first identify the most computationally intensive tasks in the algorithms, then explore the possibility to parallelize these tasks. While the exact percentage of parallelizable codes in an algorithm is hard to calculate, we give some empirical percentages of time on which the three algorithms spend mostly. These percentages are computed from results of profiling, which is a technique used in performance testing. They are the averages of 10 repeats. The percentage changes with the dimension of the data set. In Table 1, we show two extreme cases: a *fat*

data set with 4000 observations and 100 covariates, and a *slender* data set with 100000 observations and only 4 covariates.

Table 1. Percentage of CPU Time

	$n \times p$	Percentage	Tasks
Simplex	100000×4	> 90	changing rows
	4000×100	> 80	pivoting
Interior Point	100000×4	> 70	cross products
	4000×100	> 90	cross products
Smoothing	100000×4	> 50	line search + partial updates
	4000×100	> 80	partial updates

Parallelizing the general simplex algorithm for dense problems has been studied in the literature. A good survey is provided by Eckstein (1993). However, the special version of the simplex algorithm described here does not fit those parallel formulations. From Table 1, we see that, for *slender* data sets, most of the computing time with the simplex algorithm is spent on changing rows, which is not parallelizable. For *fat* data sets, most of the computing time is spent on pivoting, which is parallelizable. This indicates that parallelizing the pivoting part of the simplex algorithm can improve its performance with multiprocessors for data sets with large p.

For the interior point algorithm, most of the parallel formulations in the literature focus on the parallelization of the Cholesky factorization. However, for the primal-dual with predictor-corrector interior point algorithm implemented here, Table 1 shows that for either *slender* or *fat* data sets, the majority of the computing time is spent on formulating and computing cross products. Therefore, an efficient parallel algorithm for formulating and computing cross products can greatly improve the performance with multiprocessors, especially for data sets with large p.

Figure 2 displays the speedup achieved in our implementation of a parallel version of the interior point algorithm, together with the Ahmdal limit and the linear speedup. The simulation is run on a Sun Solaris 8 configured with 12 processors for data sets with 10000 observations and 200 variable. The Ahmdal limit is based on the parallelizable fraction of 0.9, which is close to the empirical percentage of time spent on cross products. The achieved speedup is close to the Ahmdal limit.

Parallelization the smoothing algorithm has not been studied in the literature. Here we provide some simple insights. Table 1 shows that for *slender* data sets, line search and partial updates cost up to half of the computing time. Both tasks are parallelizable. For *fat* data sets, over 80% of the computing time is spent on partial updates, while line search is not significantly

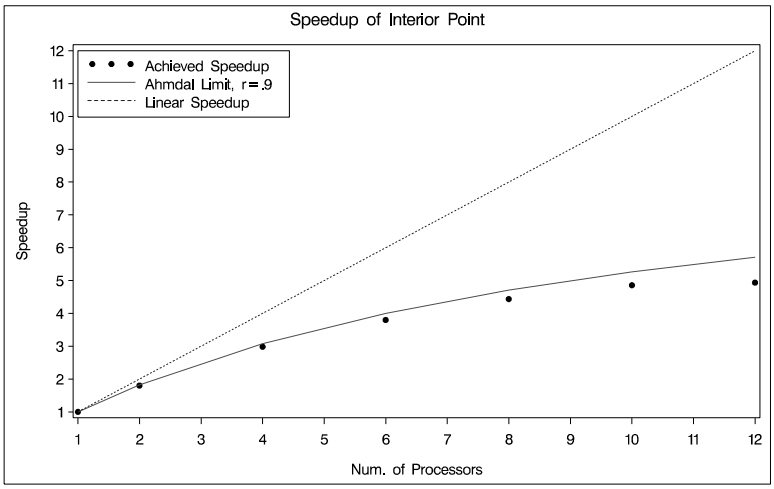

Figure 2. Speedup of the Interior Point Algorithm

time consuming anymore.

The major overheads for parallel algorithms are the communication and computation imbalance. However, the parallelizable tasks we identified in the three algorithms are simple and easy to be distributed without significant overheads.

2.6. Sparse Computing. Sparse computing is another efficient computational technique. In applications of quantile regression to nonparametric functional data analysis and semiparametric longitudinal studies, the design matrix X can have over 90% of zero elements. In sparse computing, these zero elements are neither stored nor involved in the operations. Only nonzero elements are stored and indexed with their positions in the original matrix. This saves both storage and lots of floating point operations involving zeros and can greatly improve the performance. To implement the sparse computing, Koenker and Ng (2003) have developed sparse algebra routines and they have used these routines in the interior point algorithm. Speedups can be substantial for some large problems (personal communication with Pin Ng), especially for problem with large p. Implementing the sparse routines into the simplex and smoothing algorithms is being investigated.

3 Computing Confidence Intervals

Several methods for computing confidence intervals of the regression quantiles have been proposed in the literature. They can be classified into

three categories: the direct method, which computes the confidence intervals based on the asymptotic normality of the estimated regression quantiles; the rank-score method, which computes the confidence intervals based on the inversion of the rank-score test; and the resampling method, which uses the bootstrap technique. We refer to Koenker (1994) and Kocherginsky et al. (2005) for details of these methods. In this section, we focus on three methods: sparsity, rank-score, and MCMB as recommended by the previous researches. After briefly reviewing these methods, we discuss some computational issues, including identifying the computationally expensive tasks, possibilities of optimization and parallelization. Motivated by applications in nonparametric and semiparametric quantile regression, we compare their performance with a large number of covariates.

3.1. Sparsity. Consider the linear model

$$y_i = x_i'\beta + \epsilon_i, \tag{3.1}$$

where $\{\epsilon_i\}$, $i = 1, ..., n$, are i.i.d. with the distribution function F and density $f = F'$. Assume $f(F^{-1}(\tau)) > 0$ in a neighborhood of τ. Under some mild conditions

$$\sqrt{n}(\hat{\beta}(\tau) - \beta(\tau)) \to N(0, \omega^2(\tau, F)\Omega^{-1}), \tag{3.2}$$

where $\omega^2(\tau, F) = \tau(1-\tau)/f^2(F^{-1}(\tau))$ and $\Omega = \lim_{n\to\infty} n^{-1} \sum x_i x_i'$. See Koenker and Bassett (1978).

This asymptotic distribution for the regression quantile $\hat{\beta}(\tau)$ can be used to construct confidence intervals. However, a quantity - the reciprocal of the density function

$$s(\tau) = [f(F^{-1}(\tau))]^{-1}, \tag{3.3}$$

which is called the *sparsity function*, has to be estimated first.

Since

$$s(t) = \frac{d}{dt} F^{-1}(t), \tag{3.4}$$

$s(t)$ can be estimated by the difference quotient of the empirical quantile function, i.e.,

$$\hat{s}_n(t) = [\hat{F}_n^{-1}(t + h_n) - \hat{F}_n^{-1}(t - h_n)]/2h_n, \tag{3.5}$$

where $\hat{F}_n$ is an estimate of F^{-1} and h_n is a bandwidth which tends to zero as $n \to \infty$. There are two options for the bandwidth: Bofinger bandwidth and Hall-Sheather bandwidth.

This estimator of the sparsity function is very sensitive to the i.i.d. assumption. Alternately, Koenker and Machado (1999) considered the non

i.i.d. case. By assuming the local linearity of the conditional quantile function $Q(\tau|x)$ in x, they proposed a local estimator of the density function using difference quotient. A Huber sandwich estimate of the covariance and standard error is computed and used to construct the confidence limits. One difficulty with this method is the selection of the bandwidth when using the difference quotient. With small sample size, both Bofinger and Hall-Sheather bandwidth tend to be too big to assure the local linearity of the conditional quantile function. Some heuristic adjustments should be used in these cases.

For the i.i.d. case, the major computation is for Ω^{-1}. For the non i.i.d. case, besides Ω^{-1}, two sets of regression quantiles at $\tau - h_n$ and $\tau + h_n$ need to be computed. Optimized routines can be used and the sparsity method is the fastest method.

3.2. Rank-Score. The classical theory of rank tests can be extended to the test of the hypothesis H_0: $\beta_2 = \eta$ in the linear regression model $y = X_1\beta_1 + X_2\beta_2 + \epsilon$. See Gutenbrunner and Jurečková (1992). By inverting this test, confidence intervals for the regression quantile estimates of β_2 may be constructed. This method was proposed by Koenker (1994). The rank-score method avoids computing the sparsity function and is very robust to model assumptions.

The rank-score function $\hat{a}_n(t) = (\hat{a}_{n1}(t), ..., \hat{a}_{nn}(t))$ can be solved from the dual

$$\max_a \{(y - X_2\eta)'a | X_1'a = (1-t)X_1'e,\ a \in [0,1]^n\}. \tag{3.6}$$

For a fixed quantile τ, integrating $\hat{a}_{ni}(t)$ with respect to the τ-quantile score function

$$\varphi_\tau(t) = \tau - I(t < \tau) \tag{3.7}$$

yields the τ-quantile scores:

$$\hat{b}_{ni} = -\int_0^1 \varphi_\tau(t) d\hat{a}_{ni}(t) = \hat{a}_{ni}(\tau) - (1-\tau). \tag{3.8}$$

Under the null hypothesis H_0: $\beta_2 = \eta$

$$S_n(\eta) = n^{-1/2} X_2' \hat{b}_n(\eta) \to N(0, \tau(1-\tau)\Omega_n), \tag{3.9}$$

where $\Omega_n = n^{-1} X_2'(I - X_1(X_1'X_1)^{-1}X_1')X_2$.

Let

$$T_n(\eta) = \frac{1}{\sqrt{\tau(1-\tau)}} S_n(\eta) \Omega_n^{-1/2}, \tag{3.10}$$

then $T_n(\hat{\beta}_2(\tau)) = 0$ from the constraint $X'\hat{a} = (1-\tau)X'e$ in the full model. A critical value can be specified for T_n. As explained in Koenker (1994), the dual vector $\hat{a}_n(\eta)$ is a piecewise constant in η and η may be altered without compromising the optimality of $\hat{a}_n(\eta)$ as long as the signs of the residuals in the primal quantile regression problem do not change. When η gets to such a boundary the solution does change, but may be restored by taking one simplex pivot. The process may continue in this way until $T_n(\eta)$, which is monotone in η, exceeds the specified critical value. This procedure uses parametric programming and can be computed using the simplex algorithm of Koenker and d'Orey (1993), which requires a total of $O(np \log n)$ pivots. The computational complexity is exponential in n and p. This makes the rank-score method very slow for middle or large sized data sets, even optimization and parallelization won't speedup too much. One alternative is using a faster algorithm rather than the simplex algorithm for the parametric programming. We are investigating using the smoothing algorithm for the parametric programming.

3.3. Resampling. The bootstrap can be implemented to compute confidence intervals for the regression quantile. As in other regression applications, both the residual bootstrap and the xy-pair bootstrap can be used. The former assumes i.i.d. random errors and resamples from the residual, while the later resamples xy pairs and accommodates some forms of heteroscedasticity. Koenker (1994) considered a more interesting resampling mechanism - resampling directly from the full regression quantile process, which he called the Heqf bootstrap.

Unlike these bootstrap methods, Parzen, Wei, and Ying (1994) observed that

$$S(b) = n^{-1/2} \sum_{i=1}^{n} x_i(\tau - I(y_i \le x_i'b)), \tag{3.11}$$

which is the estimating equation for the τth regression quantile, is a pivotal quantity for the true parameter β_τ, i.e., its distribution may be generated exactly by a random vector U which is a weighted sum of independent, re-centered Bernoulli variables. They further showed that for large n the distribution of $\hat{\beta}(\tau) - \beta_\tau$ can be approximated by the conditional distribution of $\hat{\beta}_U - \hat{\beta}_n(\tau)$, where $\hat{\beta}_U$ solves an augmented quantile regression problem with $n+1$ observation and $x_{n+1} = -n^{-1/2}u/\tau$ and y_{n+1} is sufficiently large for a given realization of u. This approach, by exploiting the asymptotically pivotal role of the quantile regression "gradient condition", also achieves some robustness to certain heteroscedasticity.

Although the bootstrap method by Parzen, Wei, and Ying (1994) is much simpler, it is still too time consuming for relatively large data sets, especially for high-dimensional data sets. He and Hu (2002) developed a new general resampling method, referred to as the Markov chain marginal bootstrap (MCMB). For quantile regression, the MCMB method has the advantage that it solves p one-dimensional equations instead of p-dimensional equations as do the previous bootstrap methods. This greatly improves the feasibility of the resampling method in computing confidence intervals. Since rasampling methods achieve the stability only for relatively large data sets, they are not recommended for small data sets ($n < 5000$ and $p < 20$). We refer to Kocherginsky et al. (2005) for details of the MCMB algorithm. The algorithm has several techniques to save the computing time. The key one is to compute the one-dimensional regression quantile by a weighted sample quantile, which can be efficiently computed by dividing the sample into segments on some pre-estimated scale.

Different from Kocherginsky et al. (2005), our implementation of the MCMB algorithm does not resample from the residuals, but directly from the weighted Bernoulli variables as in Parzen, Wei, and Ying (1994). The resulting MCMB confidence intervals usually do not depend on the resampling methods, since they lead to the same limit distribution. However, resampling from the weighted Bernoulli variables is somehow more robust. Besides, we use a MCMB chain of length 20 instead of 10 for the initial scale estimation to gain a better stability. With our implementation, the majority of the computing time is spent on resampling the weighted Bernoulli variables and sorting the sample in the middle segments. Optimal routines for these tasks could be used. Parallelization of these tasks is also straight-forward.

3.4. Performance with Large P. Based on Monte Carlo simulation, Kocherginsky et al. (2005) elaborate the comparison among the existing methods of constructing confidence intervals in quantile regression. The performances of these methods were well assessed under a wide range of models; from simple linear model with i.i.d. random errors to more demanding ones containing sever heteroscedastic, or highly skewed errors. They give suggestions on suitable methods depending on the sample sizes. Particularly, they recommend rank-score and MCMB due to their robustness or computational efficiency. Their simulations are based on models with small p. However, in applications of quantile regression to survey data analysis and longitudinal studies, the number of covariates could be considerably large. As the number of covariate increases, the underlying asymptotic results might be questionable, which could result in poor confidence interval estimates. Therefore, it

is helpful to assess the performances of the confidence interval methods for large p.

First, we compare performances of the three methods based on the coverage probability. For the sparsity method, we include both the i.i.d. and non i.i.d. cases. Data are generated from a simple linear model

$$y = 1 + \sum_{j=1}^{p} \beta_j x_j + e, \tag{3.12}$$

where x_j, $j = 1, ..., p$, and e are i.i.d. standard normal variables. β_1 and β_2 take value 1 and β_j, $2 < j \leq p$ take value 0. Due to the symmetry in x_1 and x_2, the average results for β_1 and β_2 with quantile 0.5 are reported. This model corresponds to Model 1 of Kocherginsky et al. (2005).

Table 2. Coverage Probability (C) and Length(L) for 90% Confidence Intervals

p	n	Method	C	L
		iid	0.6700	0.1412
20	100	nid	0.8700	0.2154
		rank	0.8950	0.2287
		mcmb	0.9250	0.2429
		iid	0.8650	0.0815
20	500	nid	0.8500	0.0785
		rank	0.9250	0.0936
		mcmb	0.9225	0.0914
		iid	0.6875	0.0575
50	500	nid	0.8175	0.0768
		rank	0.8975	0.0945
		mcmb	0.8875	0.0921
		iid	0.7950	0.0493
50	1000	nid	0.8100	0.0504
		mcmb	0.8925	0.0632
		iid	0.1500	0.0105
100	500	nid	0.9125	0.0937
		mcmb	0.8700	0.1032
		iid	0.5725	0.0322
100	1000	nid	0.8225	0.0543
		mcmb	0.8900	0.0635

Table 2 displays the coverage probabilities and lengths for the 90% confidence intervals of regression quantiles with several n-p combinations. The

Monte-Carlo sample size is 200 for each setting. For the last three n-p combinations, we skipped the runs for rank-score method because they took too much time. Table 2 indicates that for large p, the sparsity methods, especially the one assumes the i.i.d. errors, have poor performances. This is different from the conclusions of Kocherginsky et al. (2005), in which only models with small p were considered. The performance generally becomes better as the sample size increases except the sparsity method with non-i.i.d. errors. The simulated results also indicate that both rank-score and MCMB have good performances with large p. That reinforces the recommendations by Kocherginsky et al. (2005).

Second, we compare the computing speed between MCMB and rank-score. Both methods need to compute the regression quantiles first. We use the simplex algorithm for both methods, since the rank-score method bases on the simplex pivoting. We begin from middle sized p (20) to large p (500) and choose n properly. Data are generated from linear models as in (3.12) with different combinations of n and p. We skip the sparsity method, which is known as computationally efficient and thus not much comparable.

TABLE 3. CPU TIME IN SECONDS

p	n	rank-score	MCMB
20	100	0.26	0.20
	500	1.71	0.79
	1000	6.60	1.43
	5000	113.45	14.96
	10000	310.99	30.23
50	100	1.17	1.37
	500	17.09	1.93
	1000	71.57	3.89
	5000	839.43	40.12
	10000	2245.93	76.39
100	500	112.64	3.92
	1000	378.23	8.07
	5000	3983.73	80.96
	10000	10592.57	160.84
200	500	679.48	13.40
	1000	2112.54	18.40
	5000	$>$ 4 hours	184.04
500	1000	$>$ 4 hours	279.54
	5000	$>$ 6 hours	629.25
	10000	$>$ 6 hours	1314.92

Table 3 displays the CPU time for computing the confidence intervals of the p parameters. The simulations were run on a DELL GX260T with 1.8GHz and 512MB of RAM. We see that for the same n, the ratio between

the CPU time of rank-score and that of MCMB increases exponentially with p. We stopped the last four runs for rank-score since they took too much time. These results further promote the MCMB method for data sets with large p.

4 Discussions

In the previous two sections, we present the most advanced methods for computing regression quantiles and their confidence intervals. To build optimal, reliable, and efficient computational tools with these methods, we give some insights for combining them with some modern computing techniques. While we are trying to include all computational issues, no doubt, there are some good techniques or algorithms we do not cover here. One example is the preprocessing proposed by Pontnoy and Koenker (1997). We have implemented this technique with both the interior point algorithm and the smoothing algorithm and seen evident performance improvements for the *slender* data sets. Some concerns should also be addressed when using this technique. We refer to Chen (2003) for more details.

The methods and algorithms described in this paper are not the final solutions of the computational issues in quantile regression. There is still room for improvements. Kocherginsky et al. (2005) recommended that for very large data sets, the non i.i.d. sparsity method should be used. However, we experienced that this method does not give better performance with larger sample size. This might be related to the bandwidth selection. The rank-score method has very good coverage probabilities. It is stable for small data sets. But it is slow even with middle sized data sets. Optimizing the simplex pivoting improves its speed, but still does not make it comparable to MCMB. Using a faster algorithm rather than the simplex pivoting might be a solution.

Acknowledgment. The authors would like to thank Bob Rodriguez, Xuming He, the editor, and the referee for their helpful comments and discussions.

References

Barrodale, I. and Roberts, F.D.K. (1973). An improved algorithm for discrete l_1 linear approximation. *SIAM J. Numer. Anal.*, **10**, 839-848.

Clark D. I. and Osborne, M.R. (1986). Finite algorithms for Huber's M-estimator. *SIAM J. Sci. Statist. Comput.*, **6**, 72-85.

Chen, C. (2003). A finite smoothing algorithm for quantile regression. Preprint.

CHEN, C. (2004). An adaptive algorithm for quantile regression. In *Theory and Applications of Recent Robust Methods*, M. Hubert, G. Pison, A. Struyf and S. Van Aelst, eds., Birkhauser, Basel, 39-48.

ECKSTEIN, J. (1993). Large-scale parallel computing, optimization, and operations research: A Survey. *ORSA Computer Science Technical Section Newsletter*, **14**, 8-12.

GUTENBRUNNER, C. and JUREČKOVÁ, J. (1992). Regression rank scores and regression quantiles. *Ann. Statist.*, **20**, 305-330.

HE, X. and HU, F. (2002). Markov chain marginal bootstrap. *J. Amer. Statist. Assoc.*, **97**, 783-795.

HUBER, P.J. (1981). *Robust Statistics*. Wiley, New York.

KARMARKAR, N. (1984). A new polynomial-time algorithm for linear programming. *Combinatorica*, **4**, 373-395.

KOCHERGINSKY, M., HE, X. and MU, Y. (2005). Practical confidence intervals for regression quantiles, *J. Comput. Graph. Statist.*, **14**, 41-55.

KOENKER, R. (1994). Confidence intervals for regression quantiles, in *Asymptotic Statistics*, P. Mandl and M. Huskova, eds., Springer-Verlag, New York, 349-359.

KOENKER, R. and BASSETT, G.W. (1978). Regression quantiles. *Econometrica*, **46**, 33-50.

KOENKER, R. and D'OREY, V. (1993). Computing regression quantiles. *J. Roy. Statist. Soc. Ser. C (Appl. Statist.)*, **43**, 410-414.

KOENKER, R. and MACHADO, A.F. (1999). Goodness of fit and related inference processes for quantile regression, *J. Amer. Statist. Assoc.*, **94**, 1296-1310.

KOENKER, R. and NG, P. (2003). SparseM: A sparse linear algebra package for R. *J. Statist. Software*, **8**, 1-9.

MADSEN, K. and NIELSEN, H.B. (1993). A finite smoothing algorithm for linear L_1 estimation. *SIAM J. Optimization*, **3**, 223-235.

PARZEN, M.I., WEI, L.J. and YING, Z. (1994). A resampling method based on pivotal estimating functions, *Biometrika*, **81**, 341-350.

PORTNOY, S. and KOENKER, R. (1997). The Gaussian hare and the Laplacian tortoise: Computation of squared-error vs. absolute-error estimators. *Statist. Science*, **12**, 279-300.

WRIGHT, S.J. (1996). *Primal-Dual Interior Point Algorithms*. SIAM, Philadelphia.

COLIN CHEN
S2024
SAS INSTITUTE INC.
CARY, NC 27513, USA
E-mail: lin.chen@sas.com

YING WEI
DEPARTMENT OF BIOSTATISTICS
COLUMBIA UNIVERSITY
NEW YORK, NY 10032, USA
E-mail: yw2148@columbia.edu

Paper received: August 2004; revised April 2005.