Journal of Statistical Software

January 2017, Volume 76, Issue 7. *doi: 10.18637/jss.v076.i07*

bayesQR: A Bayesian Approach to Quantile Regression

Dries F. Benoit
Ghent University

Dirk Van den Poel
Ghent University

Abstract

After its introduction by Koenker and Basset (1978), quantile regression has become an important and popular tool to investigate the conditional response distribution in regression. The R package **bayesQR** contains a number of routines to estimate quantile regression parameters using a Bayesian approach based on the asymmetric Laplace distribution. The package contains functions for the typical quantile regression with continuous dependent variable, but also supports quantile regression for binary dependent variables. For both types of dependent variables, an approach to variable selection using the adaptive lasso approach is provided. For the binary quantile regression model, the package also contains a routine that calculates the fitted probabilities for each vector of predictors. In addition, functions for summarizing the results, creating traceplots, posterior histograms and drawing quantile plots are included. This paper starts with a brief overview of the theoretical background of the models used in the **bayesQR** package. The main part of this paper discusses the computational problems that arise in the implementation of the procedure and illustrates the usefulness of the package through selected examples.

Keywords: quantile regression, asymmetric Laplace distribution, Bayesian approach, logistic regression, probit, R, Fortran.

1. Introduction

Regression analysis helps us to understand how the typical value of a variable of interest changes when any of the independent variables is varied, while the other independent variables are held fixed. The earliest attempt to estimate regression type of models was performed by Boscovich who investigated the ellipticity of the earth in the 1750's (Stigler 1984). It is interesting to see that his attempt was closely tied to the notion of median (as opposed to mean) regression. However, subsequent developments in regression models mainly used least-squares (OLS), i.e., conditional mean models, as the preferred approach. The computational simplicity together with the optimality when the observed deviation from the regression line is normal, made OLS the workhorse model of regression analysis for many years.

Yet, conditional mean models give only partial information on the effect of the covariate on the dependent variable. As Mosteller and Tukey (1977) point out, just as the mean often gives a rather incomplete picture of a single distribution, so does the regression curve give an incomplete picture for a set of distributions. In their seminal paper, Koenker and Basset (1978) introduced quantile regression to estimate the conditional quantile function of the response. As such, the method makes it possible to get a more complete view of the statistical relations among stochastic variables. It allows the user to examine the relationship between a set of covariates and the different parts of the response distribution.

Next to the motivation in terms of "richer" view, quantile regression can also be motivated from a robustness point of view. Rare but very large deviations from the regression line can strongly influence the fit of OLS. Median estimators and more general quantile estimators are less influenced by outlying observations in the response variable conditional on the covariates. In addition, it is important to recognize that covariates can exert an effect on the dispersion of the response variable as well as its location. When this so called heteroskedasticity is present, quantile regression in contrast to mean regression, offers some expanded flexibility of covariate effects.

Ever since the important overview book about the state-of-the-art in quantile regression of Koenker (2005), the number of research papers on quantile regression increased rapidly. Applications of quantile regression arose in many research areas, ranging from ecology over genetics to economics. One of the more recent research areas of quantile regression investigates Bayesian estimation of model parameters (e.g., Yu and Moyeed 2001).

This paper presents the **bayesQR** package (Benoit, Al-Hamzawi, Yu, and Van den Poel 2017). It is a free extension to the R system for statistical computing (R Core Team 2016). The package contains a set of functions that estimate different types of quantile regression models using a Bayesian approach. This is in contrast to the **quantreg** package (Koenker 2016) that uses the frequentist methodology to estimate quantile regressions. Note that this paper does not intend to compare or value Bayesian versus frequentist methods. The goal of **bayesQR** and this paper is simply to offer a Bayesian flavor to quantile regression to those who have practical or philosophical motivations to do so.

The **bayesQR** package is not the only R package that is focused on Bayesian estimation of quantile regression estimates. The **Brq** package (Al-Hamzawi 2012) contains a function for quantile regression for continuous dependent variables, as well as tobit quantile regression, but is not under active development anymore. Also, there exists one routine, i.e., `MCMCquantreg`, in the package **MCMCpack** (Martin, Quinn, and Park 2011) that has similar functionality as the `bayesQR` function in the **bayesQR** package. The downside of the above packages or functions is that the implementation is written in pure R which slows computations down considerably. This might cause problems when trying to tackle moderate to large problems. In contrast, because the **bayesQR** package has its computational core written in modern Fortran, the implementation is very fast. Moreover, **bayesQR** provides functionalities (e.g., support for binary dependent variables, variable selection, plotting, . . .) that are not contained in other packages. The package **bayesQR** is available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=bayesQR`.

After a brief summary of the general quantile regression theory (Section 2), the Bayesian approach to quantile regression used in the **bayesQR** package will be discussed (Section 3). Section 4 contains a detailed description of the functionalities of **bayesQR**, while its application to real datasets is illustrated in Section 5. The paper ends with final remarks and a discussion about possible future developments in Section 7.

2. Quantile regression

The classical way to derive quantile regression is to start from the well-known regression model. Consider the standard linear model where $x_i \in \Re^k$ and $\beta \in \Re^k$ are column vectors of size k and $y_i \in \Re$ is a scalar variable:

$$y_i = x_i^\top \beta + \epsilon_i. \tag{1}$$

If we assume that $\mathsf{E}(\epsilon|x) = 0$, then we obtain the conditional mean model, while if $\mathrm{Med}(\epsilon|x) = 0$ then we get the conditional median model. In the conditional mean model, we can find the regression coefficients by solving:

$$\hat{\beta} = \underset{\beta \in \Re^k}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - x_i^\top \beta)^2. \tag{2}$$

In median regression, we proceed in exactly the same way, but here, we try to find an estimate of β that minimizes the sum of the absolute deviations:

$$\hat{\beta} = \underset{\beta \in \Re^k}{\operatorname{argmin}} \sum_{i=1}^{n} |y_i - x_i^\top \beta|. \tag{3}$$

Quantile regression proceeds by extending the median case to all other quantiles of interest:

$$\hat{\beta}_\tau = \underset{\beta \in \Re^k}{\operatorname{argmin}} \sum_{i=1}^{n} \rho_\tau(y_i - x_i^\top \beta), \tag{4}$$

where $\rho_\tau(x) = x(\tau - I(x < 0))$ and where $I(\cdot)$ denotes the indicator function. For any $\tau \in (0,1)$, the loss function ρ_τ assigns a weight of τ to positive residuals and a weight of $(1-\tau)$ to negative residuals. The quantity $\hat{\beta}_\tau$ is called the τth regression quantile. Note that the case where τ equals 0.5 corresponds to minimizing the sum of absolute residuals, i.e., median regression.

The method of Barrodale and Roberts (1973) has been widely used to minimize the objective function in Equation 4. For example, in both the **quantreg** R package as well as the SAS `PROC QUANTREG` procedure it is the default method. However, this variant of the simplex algorithm is computationally demanding in large statistical applications. Both the **quantreg** R package and SAS propose an interior point approach for larger problems. The confidence intervals are generally computed by inverting the rank score test using the simplex algorithm. For large problems, the resampling method can be a solution.

3. Bayesian quantile regression

3.1. Continuous dependent variables

Koenker and Machado (1999) were the first to show that likelihood-based inference using independently distributed asymmetric Laplace densities (ALD) is directly related to the minimization problem in Equation 4. Yu and Moyeed (2001) elaborated the ALD approach and provided an approach that initiated the take-off of Bayesian quantile regression. However, it should be emphasized that there exist multiple Bayesian approaches to quantile regression. For example, Kottas and Gelfand (2001) develop Bayesian quantile regression based on Dirichlet process priors, Dunson, Watson, and Taylor (2003) developed an approach based on substitution likelihoods and more recently, Lancaster and Jun (2010) and Yang and He (2012) proposed a method based on the empirical likelihood. These approaches avoid the parametric likelihood of the ALD approach. Sriram, Ramamoorthi, and Ghosh (2013) recently showed that the misspecified likelihood in the ALD approach still leads to consistent results (see Section 3.4). We believe that the ALD approach is a valuable and relatively simple alternative to these other methods that does not need complex choices of prior distributions and prior (hyper-)parameters (Yu and Stander 2007).

Yu and Zhang (2005) propose a three-parameter ALD with a skewness parameter that can be used directly to model the quantile of interest:

$$f(x|\mu,\sigma,\tau) = \frac{\tau(1-\tau)}{\sigma}\exp\left\{-\rho_\tau\left(\frac{x-\mu}{\sigma}\right)\right\}, \tag{5}$$

where again $\rho_\tau(x) = x(\tau - I(x<0))$ and where $I(\cdot)$ denotes the indicator function. Minimizing Equation 4 is equivalent to maximizing a regression likelihood (see Equation 5) using ALD errors with $\mu = x_i^\top\beta$.

Bayesian implementation of quantile regression begins by forming a likelihood comprised of independent asymmetric Laplace densities with $\mu = x_i^\top\beta$. Next, the quantile of interest, τ, has to be specified and priors should be put on the model parameters β and σ. The resulting posterior distribution can be represented as follows:

$$\psi(\beta,\sigma|y,x,\tau) \propto \pi(\beta,\sigma)\prod_{i=1}^{n}\mathrm{ALD}(y_i|x_i^\top\beta,\sigma,\tau), \tag{6}$$

where $\pi(\beta,\sigma)$ is the joint prior on the regression parameters. Inference about model parameters then follows conventional Bayesian procedures (see Section 4 for details).

3.2. Binary dependent variables

Consider the standard binary regression model:

$$y_i = I(y_i^\star \geq 0) = x_i^\top\beta + \epsilon_i, \tag{7}$$

where y_i is the indicator of the ith individual's response determined by the underlying variable $y^\star$, x_i is a $k\times 1$ vector of explanatory variables, β is a $k\times 1$ vector of regression coefficients, ϵ_i is a random error term and $i = 1,\ldots,n$.

Manski's maximum score estimator is considered the first attempt in econometrics to deal with binary quantile regression (Manski 1975, 1985). The original maximum score estimator focused on the median case, but was extended to the more general quantiles later:

$$\hat{\beta}_\tau = \underset{\beta \in \Re}{\operatorname{argmax}}\, n^{-1} \sum_{i=1}^{n} \rho_\tau(2y_i - 1)(2y_i - 1)\operatorname{sgn}(x_i^\top \beta) \tag{8}$$

for any quantile $\tau \in (0,1)$. $\mathsf{sgn}(\cdot)$ is the signum function and again $\rho_\tau(\cdot)$ is the loss function. Scale normalization is necessary because the parameter β is only identified up to a scale (e.g., set $\|\beta\| = 1$).

Benoit and Van den Poel (2012) discuss a number of problems that arise with the maximum score estimator. These difficulties are related to the difficult optimization of the multidimensional step function. As a result, calculating a consistent estimator is challenging. Moreover, because of the complicated asymptotic distribution of the estimator, making inference is daunting.

At the cost of accepting additional distributional assumptions on the error terms, the Bayesian ALD approach sets the problem in a parametric framework in which these problems are avoided (Benoit and Van den Poel 2012). Let

$$\mathsf{P}(y_i = 1 | x_i, \beta) = 1 - F_{y^\star}(-x_i^\top \beta), \tag{9}$$

where $F_{y^\star}(\cdot)$ is the cumulative distribution function of the asymmetric Laplace variable $y^\star$. The joint posterior distribution, $\psi(\cdot)$, of the unobservables β and $y^\star$ given the data x and the quantile of interest τ, is then given by:

$$\psi(\beta, y^\star | y, x, \tau) \propto \pi(\beta) \prod_{i=1}^{n} \{ I(y_i^\star > 0) I(y_i = 1) + I(y_i^\star \leq 0) I(y_i = 0) \} \times \mathrm{ALD}(y_i^\star | x_i^\top \beta, \sigma = 1, \tau) \tag{10}$$

where $\pi(\beta)$ is the prior on the regression coefficients and $I(\cdot)$ is the indicator function. Note that $\sigma = 1$ because of identification issues similar as in Equation 8. Inference about model parameters then follows conventional Bayesian procedures (see Section 4 for details).

3.3. Variable selection using the adaptive lasso

Variable selection plays an important role in the model-building process. Many practical applications have a large number of potential covariates to include in the model. However, it is undesirable to keep irrelevant predictors in the final model since this makes it difficult to interpret the resultant model and may decrease its predictive ability (Wu and Liu 2009). One way to achieve sparseness in the covariate space is to add a penalty term to the objective function. One well-known approach is the l_1 penalty used in the lasso proposed by Tibshirani (1996). Zou (2006) extended the lasso by using adaptive weights for penalizing different coefficients in the penalty and demonstrated its oracle properties.

The first use of regularization in quantile regression is made by Koenker (2004). The author developed an l_1-regularization quantile regression method to shrink individual effects towards a common value. Wang, Li, and Jiang (2007) introduced median regression with the adaptive lasso penalty, while Wu and Liu (2009) extended the approach to general quantile regression. Al-Hamzawi, Yu, and Benoit (2012) provide a Bayesian framework to estimate quantile regression models with the adaptive lasso that is based on the work of Li, Xi, and Lin (2010).

A Bayesian approach to the adaptive lasso for quantile regression models can be achieved by employing a symmetric Laplace prior for every regression parameter β_j. Consider again Equation 6. Instead of placing the usual normal or uniform prior (see Yu and Moyeed 2001) on $\pi(\beta)$, a Laplace prior, with zero mean and scale σ/λ_j, is put on every element j of β. The resulting posterior for β is then given by:

$$\psi(\beta|y, x, \sigma, \lambda, \tau) \propto \exp\left\{-\frac{1}{\sigma}\sum_{i=1}^{n}\rho_\tau(y_i - x_i^\top\beta) - \frac{\lambda_j}{\sigma}\sum_{j=1}^{k}|\beta_j|\right\}. \tag{11}$$

Note how the second term in the exponent acts as a penalty on the usual quantile regression likelihood in the same way as Koenker's lasso penalty (see Li *et al.* 2010). In contrast to the normal lasso procedure where there is one regularization parameter for all regression estimates, the adaptive lasso has such a parameter for every regressor, i.e., λ_j. The λ_j are not treated as known, but estimated from the data. As a result, a prior distribution is put on every element j of λ. Section 4.3 gives more details on the choice of prior distributions and the method of estimating the posterior distribution of the model parameters for the adaptive lasso approach.

3.4. Controversies with the ALD approach

The motivation for quantile regression is usually that there is some departure from the assumption of independent and identically distributed (IID) error terms, e.g., in the case of error heteroskedasticity. In this case, the OLS assumptions do not hold and quantile regression can offer additional flexibility. Yet the Bayesian ALD approach does assume IID errors and this contrasts with the main motivation for quantile regression. The motivation for the approach follows from the fact that the maximum likelihood estimate from the ALD likelihood is equivalent to solving the problem in Equation 4.

Often, the assumption of ALD errors is difficult to maintain and the likelihood becomes misspecified. As such, the ALD approach cannot be considered a truly Bayesian procedure when the errors are not distributed according the ALD. Nevertheless, based on empirical studies, Yu and Moyeed (2001) argued that the results are satisfactory even if the ALD is a misspecification. A more formal justification was given by Sriram *et al.* (2013). They show the posterior consistency based on the misspecified ALD likelihood under fairly general conditions.

However, the posterior consistency results do not imply that the interval estimates constructed from the posterior are automatically valid. This issue is investigated by Sriram, Ramamoorthi, and Ghosh (2012). They argue that it is desirable that, asymptotically, a 95% Bayesian credible interval contains the true parameter value 95% of the time, just like the frequentist confidence interval. These authors find that with increasing sample size, coverage improves and the length of the credible intervals decreases. However, while the credible intervals can be indicative of the true parameter values, they potentially end up over-covering or under-covering the true parameter value.

Recently, both Yang, Wang, and He (2015) and Sriram (2015) proposed a correction to the

MCMC iterations to construct asymptotically valid intervals. Their method builds on the work of Chernozhukov and Hong (2003) and Yang and He (2012). In these papers, it is shown that the MCMC iterations provide a valid piece of the usual sandwich formula (see e.g., Koenker 2005) that can be used to construct a valid estimate of the limiting covariance matrix of the Gaussian approximation. The correction can be accomplished as follows (Yang *et al.* 2015):

$$\sqrt{n}\hat{\Sigma}_{\text{adj}} = \frac{n\tau(1-\tau)}{\sigma^2}\hat{\Sigma}(\sigma)\hat{D}_0\hat{\Sigma}(\sigma), \tag{12}$$

where $\hat{\Sigma}(\sigma)$ is the posterior variance-covariance matrix and $\hat{D}_0 = n^{-1}\sum_{i=1}^{n} x_i x_i^\top$.

4. Computational details of the bayesQR package

We have implemented the methodology described above in an R package, called **bayesQR**. The package contains one main function, i.e., `bayesQR`, that can estimate different types of quantile regression models. That is, both continuous as well as binary dependent variable models can be estimated with or without adaptive lasso variable selection. The core computational part of the estimation procedures is written in Fortran to speed up the calculations as much as possible. The R function `bayesQR` is mainly a wrapper function that does some error handling and passes the arguments to the Fortran core. When the computation is finished, the Fortran core will pass the output to an R S3 object of class '`bayesQR`' for post processing in R. As such, the more applied R user should not notice, nor care, that Fortran was called from R.

The main routine is supplemented with other routines that help to summarize, plot or print the results of the `bayesQR` routine. Two functions are provided that do post processing of the MCMC chains, i.e., the S3 methods `summary` and `plot`. The first function, summarizes the draws of the regression parameters by returning the posterior credible interval and Bayes estimate. The latter function creates quantile plots based on the estimates obtained by estimating a series of quantile regressions. This function also has the option to graphically summarize the MCMC chain by producing the traceplot or posterior histogram of the draws. Finally, the S3 method `predict` will calculate the predicted probabilities of binary quantile regression models.

In this section, we will discuss the computational details of the functions in the **bayesQR** package. The focus will be on the implementation of the setup of the MCMC chains for the different quantile regression models. To some extent, Fortran-specific computational issues will be discussed too.

Function `bayesQR` is the main function for estimating Bayesian quantile regression models. A simple function call is as follows:

```
R> bayesQR(formula = y ~ X, quantile = 0.1, alasso = TRUE)
```

This code will estimate the first decile quantile regression of y on X using adaptive lasso variable selection. Since prior information is omitted, the default vague prior (e.g., a normal and gamma prior with high variance on β and λ respectively) will be used (see Section 4.4) and, in case of a continuous dependent variable, by default the normal approximation of the posterior distribution will be alculated (see Section 3.4). The `bayesQR` function decides which MCMC algorithm to use, based on its input values. The method detects whether the dependent variable is binary or continuous and the logical `alasso` value indicates if adaptive lasso is desired. The logical `normal.approx` indicates whether the MCMC iterations should be used to calculate the normal approximation of the posterior. Below, the computational details of each of the four different MCMC algorithms are discussed.

4.1. MCMC for continuous quantile regression without adaptive lasso

Consider again Equation 6. In this equation the likelihood is well defined. However, to be able to calculate the posterior distribution $\psi(\beta, \sigma | y, x, \tau)$, we have to make a decision about the prior on the model parameters β and σ. A straightforward prior for β is the multivariate normal distribution, Normal(mean $= \beta_0$, varcov $= V_0$). It should be noted that other priors for β are possible too. For example, Yu and Moyeed (2001) show that even a uniform prior on β would result in a proper posterior distribution. For the prior on σ, we propose the inverse gamma distribution invGamma(shape $= n_0$, scale $= s_0$), with density:

$$f(x|n_0, s_0) = \frac{s_0^{n_0}}{\Gamma(n_0)} x^{-n_0-1} \exp\left(-\frac{s_0}{x}\right). \tag{13}$$

As we will show in the next paragraphs, the choice for these prior distributions will make the calculation of the posterior distribution a lot more efficient.

The initial approach to Bayesian quantile regression using the ALD proposed the Metropolis-Hastings algorithm to estimate the model parameters (Yu and Moyeed 2001). This results from the fact that multiplying the priors defined above with the likelihood defined in Equation 6 does not lead to a posterior distribution of known form. All versions before version 2.0 of the **bayesQR** package use this algorithm to estimate the model parameters. However, previous work has shown that exploiting the location-scale mixture of normals representation of the ALD leads to a more efficient Gibbs sampling algorithm to estimate the posterior (see Tsionas 2003). In the **bayesQR** package, however, we implemented the Gibbs sampler proposed by Kozumi and Kobayashi (2011). The advantage of their approach is that β can be updated in one draw, while in Tsionas (2003) each element of β has to be updated separately. From version 2.0 of the **bayesQR** package on, this efficient Gibbs sampler is used to estimate the model parameters.

Consider again the error term ϵ in Equation 1. In this Bayesian approach to quantile regression, the error term is assumed to follow the asymmetric Laplace distribution. It can be shown that the ALD can be represented as a location scale mixture of normal distributions where the mixing distribution follows an exponential distribution (for a proof see Kozumi and Kobayashi 2011). This implies that Equation 1 can equivalently be rewritten as:

$$y_i = x_i^\top \beta + \theta \nu_i + \omega \sqrt{\sigma \nu_i} u_i, \tag{14}$$

where

$$\nu_i = \sigma z_i, \qquad \theta = \frac{1-2\tau}{\tau(1-\tau)}, \qquad \text{and} \qquad \omega^2 = \frac{2}{\tau(1-\tau)},$$

and z_i is a standard exponential and u_i is a standard normal variate. This leads to the following likelihood function:

$$f(y_i | x_i^\top \beta, \nu_i, \sigma, \tau) \propto \exp\left(-\sum_{i=1}^{n} \frac{(y_i - x_i^\top \beta - \theta \nu_i)^2}{2\omega^2 \sigma \nu_i}\right) \prod_{i=1}^{n} \frac{1}{\sqrt{\sigma \nu_i}}. \tag{15}$$

Now, using data augmentation (Tanner and Wong 1987), a Gibbs sampling algorithm can be set up by drawing β, σ and z from their full conditional distributions. The usual Bayesian computations show that the full conditional density of β is given by:

$$\psi(\beta|\sigma,\nu,y,x,\tau) \sim \mathrm{N}(\tilde{\beta},\tilde{V}), \tag{16}$$

where

$$\tilde{V}^{-1} = \sum_{i=1}^{n} \frac{x_i^\top x_i}{\omega^2\sigma\nu_i} + V_0^{-1} \qquad \text{and} \qquad \tilde{\beta} = \tilde{V}\left(\sum_{i=1}^{n} \frac{x_i(y_i-\theta\nu_i)}{\omega^2\sigma\nu_i}\right).$$

The algorithm to sample from the multivariate normal distribution that was implemented in the Fortran code of the **bayesQR** package, is based on the well-known method of Cholesky decomposition of the variance-covariance matrix. The draws from the one dimensional normal distribution are obtained by using the Box-Muller transform (Box and Muller 1958).

From Equation 15 together with a standard exponential density, the full conditional distribution of ν_i is proportional to:

$$\nu_i^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}\left(\hat{\delta}_i^2\nu_i^{-1} + \hat{\gamma}_i^2\nu_i\right)\right\}. \tag{17}$$

This is the kernel of a generalized inverse Gaussian distribution $\mathrm{GIG}(\frac{1}{2},\hat{\delta}_i,\hat{\gamma}_i)$, with $\hat{\delta}_i^2 = \frac{(y_i-x_i^\top\beta)^2}{\sigma\omega^2}$ and $\hat{\gamma}^2 = \frac{2}{\sigma} + \frac{\theta^2}{\sigma\omega^2}$. By recognizing the fact that $\mathrm{GIG}(p,a,b) = \frac{1}{\mathrm{GIG}(-p,b,a)}$ and that $\mathrm{GIG}(-\frac{1}{2},b,a)$ equals the inverse Gaussian distribution, we can sample from the conditional posterior of z_i by drawing from the inverse Gaussian distribution and inverting this draw (Jørgensen 1982). The algorithm to sample from the inverse Gaussian distribution that was implemented in the Fortran code of the **bayesQR** package, is the method proposed by Michael, Schucany, and Haas (1976).

Finally, knowing that $v_i \sim \mathrm{Exponential}(\sigma)$, the full conditional posterior of σ is proportional to:

$$\left(\frac{1}{\sigma}\right)^{n_0+\frac{3}{2}n+1} \exp\left[-\frac{1}{\sigma}\left\{s_0 + \sum_{i=1}^{n}\nu_i + \sum_{i=1}^{n}\frac{(y_i - x_i^\top\beta - \theta\nu_i)^2}{2\omega^2\nu_i}\right\}\right]. \tag{18}$$

This is the kernel of an inverse gamma distribution with parameters shape $= \{2n_0 + 3n\}/2$ and scale $= \{2s_0 + 2\sum_{i=1}^{n}\nu_i + \sum_{i=1}^{n}(y_i - x_i^\top\beta - \theta\nu_i)^2/\omega^2\nu_i\}/2$. Random draws from the inverse gamma distribution, $x \sim$ invGamma(shape, scale), can easily be obtained by sampling from $x \sim 1/\text{gamma(shape, 1/scale)}$. The algorithm to sample from the gamma distribution that was implemented in the Fortran code of the **bayesQR** package, is the method proposed by Marsaglia and Tsang (2000).

This Gibbs sampler is fairly straightforward to implement in Fortran. However, two code sections might need some additional explanation. First, when estimating the variance-covariance matrix for the posterior of β, the tensor product of a vector of length k, i.e., x_i, with itself has to be calculated. We followed the code proposed by Press, Teukolsky, Vetterling, and Flannery (1996). That is:

```
spread(vector1, dim = 2, ncopies = k) * spread(vector1, dim = 1, ncopies = k)
```

Second, simulating from the multivariate normal distribution requires to calculate the Cholesky root of a variance-covariance matrix. The Cholesky decomposition of a positive definite matrix is efficiently implemented in the function `dpotrf` of the Fortran library **LAPACK** (Anderson *et al.* 1999). When installing the **bayesQR** package in R, the source code will automatically be compiled against the **LAPACK** library (that is available through R).

Note that by default, the MCMC iterations are used to construct the normal approximation of the posterior distribution. That is, the `bayesQR` option `normal.approx` is set to `TRUE` by default. Following the suggestion of Yang *et al.* (2015), σ^2 is then set to a fixed value (i.e., $\sigma^2 = 1$) which simplifies the Gibbs sampler.

4.2. MCMC for binary quantile regression without adaptive lasso

The first paper that proposes a Bayesian approach to binary quantile regression (i.e., Benoit and Van den Poel 2012) makes use of the Metropolis-Hastings algorithm to estimate the model parameters. Versions of the **bayesQR** package prior to version 2.0 used this method. However, by exploiting the location-scale mixture of normals representation of the asymmetric Laplace distribution, we can again set up a much more efficient Gibbs sampler to calculate the posterior. In addition, the Gibbs sampler is much more user friendly as it does not require to tweak parameters in the MCMC chain (e.g., step sizes). Therefore, from version 2.0 on, this is the approach that is implemented in the package. Note that the Gibbs sampler for binary quantile regression was also proposed by Ji, Lin, and Zhang (2012).

The MCMC sampler for binary quantile regression is very similar to the Gibbs sampler for quantile regression for continuous dependent variables. In this section, we will only discuss the difference with the calculations of Section 4.1.

The standard Bayesian approach for categorical dependent variable models is to assume an unobserved variable that determines the outcome. This can be seen from Equation 7 and Equation 10. If the latent $y^\star$ would be available to the researcher, the MCMC chain would be identical to the continuous quantile regression setting as discussed in Section 4.1. But since $y^\star$ is unknown, we will estimate this quantity from the data.

Consider again the posterior of the Bayesian binary quantile regression in Equation 10. Splitting up this complicated posterior of unknown form in the conditional posterior of β given $y^\star$, and in the posterior distribution of $y^\star$ conditional on β facilitates sampling the joint posterior. From Equation 10 we find that the fully conditional distribution of $y_i^\star$ is given by:

$$\begin{aligned} \psi(y_i^\star|\beta,\tau,x_i,y_i) &\sim \text{ALD}(x_i^\top\beta,\sigma=1,\tau) \text{ truncated at the left by 0, if } y_i=1, \qquad (19)\\ \psi(y_i^\star|\beta,\tau,x_i,y_i) &\sim \text{ALD}(x_i^\top\beta,\sigma=1,\tau) \text{ truncated at the right by 0, if } y_i=0. \end{aligned}$$

Drawing random samples from the ALD is fairly easy. One could, for example, implement the inverse transformation method (Devroye 1986). However, due to the truncation, for some observations the latent $y^\star$ might be in the extreme tails of the distribution. Consequently, the inverse transformation method can be unstable in some situations. Prior to version 2.0 of the **bayesQR** package, the truncated ALD was sampled by exploiting the fact that each tail of this distribution is an exponential distribution. This algorithm is quite efficient and has not been described elsewhere. The algorithm exploits the fact that both tails of the ALD distribution are exponential distributions with possibly different rate parameters. Thus, drawing from the extreme tails can be done easily by drawing from the suitable exponential distribution. An R implementation is given below:

```
R> truncALD <- function(y, mu, sigma, tau) {
+    trunc <- -mu/sigma
+    if ((y == 0) & (trunc <= 0)) {
+      u <- runif(n = 1)
+      fn_val <- -(abs(trunc) - log(u)) / (1 - tau)
+    } else if ((y == 1) & (trunc >= 0)) {
+      u <- runif(n = 1)
+      fn_val <- trunc - log(u) / p
+    } else {
+      g <- pALD(q = trunc, mu = 0, sigma = 1, tau = tau)
+      if (y == 1) {
+        min <- g
+        max <- 1
+      } else {
+        min = 0
+        max = g
+      }
+      u <- runif(n = 1)
+      u <- min + (max - min) * u
+      fn_val <- qALD(p = u, mu = 0, sigma = 1, tau = tau)
+    }
+    fn_val <- mu + fn_val * sigma
+  }
```

where `pALD` and `qALD` are the distribution function and the cumulative distribution function of the ALD respectively (see Yu and Zhang 2005).

However, from version 2.0 of the **bayesQR** package on, a Gibbs sampling algorithm is used to estimate the model parameters. As shown in Section 4.1, this algorithm relies on the location-scale mixture of normals. For the simulation of the truncated ALD, we also can rely on this representation. That is, conditional on the mixing parameter ν, the problem simplifies to simulating from the truncated univariate normal distribution.

$$\psi(y_i^\star|x_i\beta, y_i, \nu_i, \theta, \omega) = \{I(y_i^\star > 0)I(y_i = 1) + I(y_i^\star \leq 0)I(y_i = 0)\} \\ \times \mathrm{N}(\mathrm{mean} = x_i^\top\beta + \theta\nu_i,\ \mathrm{sd} = \omega\sqrt{\nu_i}) \quad (20)$$

We implemented the algorithm of Geweke (1989) in Fortran to generate draws from the truncated normal distribution.

Once the latent $y^\star$ are sampled, we can condition on these values. Consequently, for the parameters ν and β we can use the same sampler as explained in Section 4.1. Because of identification issues explained in Section 3.2, we set $\sigma = 1$ and thus this parameter is not sampled. Also note that the sampler requires to invert a variance-covariance matrix. We efficiently implemented this in the Fortran code by linking to the **LAPACK** library (Anderson *et al.* 1999). The inverse of a positive definite matrix can be accomplished by first calling `dpotrf`, i.e., calculating the Cholesky root of that matrix. Next, the subroutine `dpotri` calculates the inverse of a positive definite matrix where the Cholesky factorization is given.

4.3. MCMC for adaptive lasso variable selection

Al-Hamzawi *et al.* (2012) describe how particular prior distributions lead to a very efficient Gibbs sampling algorithm for the Bayesian adaptive lasso model. Their choice of prior distributions is again motivated by the location-scale mixture of normals representation of the asymmetric Laplace, both in the likelihood as in the prior. The penalty term in Equation 11, that is the Laplace prior on β, can be rewritten as:

$$\begin{aligned}\psi(\beta|\sigma,\lambda) &= \prod_{j=1}^{k}\frac{\eta}{2}\exp\{-\eta|\beta_j^2|\} \qquad (21)\\ &= \prod_{j=1}^{k}\int_0^\infty \frac{1}{\sqrt{2\pi s_j}}\exp\left\{-\frac{\beta_j^2}{2s_j}\right\}\frac{\eta^2}{2}\exp\left\{-\frac{\eta^2}{2}s_j\right\}\mathrm{d}s_j\end{aligned}$$

with $\eta = \lambda/\sigma$.

If we put an inverse gamma prior on σ and gamma priors on η_j^2, we get the following posterior distributions (see Al-Hamzawi *et al.* 2012 and Kozumi and Kobayashi 2011 for more details on the derivations of the posterior distributions).

The full conditional posterior for β_j is:

$$\psi(\beta_j|\nu,s_j,\sigma,\eta) \propto \exp\left\{-\frac{1}{2}\sum_{i=1}^{n}\frac{(y_i - x_{ij}\beta_j - \theta\nu_i)^2}{\sigma\omega^2\nu_i}\right\}\exp\left\{-\frac{\beta_j^2}{2s_j}\right\}. \qquad (22)$$

Define $\tilde{y}_{ip} = y_i - \theta\nu_i - \sum_{j=1,j\neq p}^{k} x_{ij}\beta_j$ and $\tilde{\sigma}_j^{-2} = (\sigma\omega^2)^{-1}\sum_{i=1}^{n} x_{ij}^2\nu_i^{-1} + s_k^{-1}$ and $\tilde{\mu}_j = \tilde{\sigma}_j^2(\sigma\omega^2)^{-1}\sum_{i=1}^{n}\tilde{y}_{ij}x_{ij}\nu_i^{-1}$. The full conditional posterior for β_j now is the normal distribution $\mathrm{N}(\tilde{\mu}_j, \tilde{\sigma}_j^2)$.

The full conditional posterior for s_j is:

$$\psi(s_j|\beta,\eta) \propto \frac{1}{\sqrt{s_j}}\exp\left\{-\frac{1}{2}(\eta^2 s_j + \beta_j^2 s_j^{-1})\right\}. \qquad (23)$$

This is again a Generalized Inverse Gaussian distribution. In Section 4.1 it was discussed how we can simulate from this distribution.

The full conditional posterior for η^2 is:

$$\psi(\eta^2|s_j) \propto \eta^{2c}\exp\left\{-\left(\frac{\sum_{j=1}^{k}s_j}{2} + d\right)\eta^2\right\}. \qquad (24)$$

This is a gamma distribution with shape parameter $(1+c)$ and rate parameter $\sum_{j=1}^{k} s_j/2 + d$.

Finally, the full conditional posteriors ν_i and σ are identical to the posteriors of ν_i and σ in Section 4.1 and thus need no further explanation.

Note that these conditional distributions are the same in the context of binary quantile regression with adaptive lasso. The only difference is that in this case, we have to work with the simulated latent $y^\star$ and for identification issues σ is set to unity (see Section 4.2). Apart from that, the above conditional posterior distributions are unchanged. Also, implementing this adaptive lasso extension in Fortran causes no difficulties that are not yet described above.

Also note that all MCMC samplers of the quantile regression models that can be invoked with `bayesQR`, call a Fortran subroutine `intprint` that is used to print some information about the current status of the sampler to the screen. The subroutine `intprint` does not come with the **bayesQR** package, but is available trough R. It is the preferred way to print from within a Fortran subroutine to the console. The behavior of this subroutine can vary depending on the operating system and the IDE that is used to run R. Normally, every 500 iterations the iteration number is printed to the screen. However, it is possible that some users only see the information printed to the screen once the MCMC estimation is finished.

4.4. The `prior` function for creating informative priors

The `prior` function creates an S3 object of class '`bayesQR.prior`' that contains all necessary prior information to estimate a Bayesian quantile regression model. If this object is omitted in the `bayesQR` call, then a standard vague prior will be used. However, in some situations the researcher might have relevant prior information that he/she wants to include in the model. The `prior` function makes sure this information is passed in a correct way to the MCMC algorithm.

The function basically creates a list of prior parameters, checking if the specified parameter names make sense for the given problem and making sure that each parameter has the correct dimensions. If a prior parameter is omitted from the function call, then this parameter will get a default value.

For example, the following prior states that the researcher's uncertainty about β in a quantile regression model with continuous dependent variable and without adaptive lasso can be expressed as a normal distribution with $\mu = -5$ and $\sigma^2 = 4$:

```
R> prior(y ~ X, alasso = FALSE, beta0 = rep(-5, ncol(X) + 1),
+    V0 = 4 * diag(ncol(X) + 1))
```

4.5. Functions for post-processing of '`bayesQR`' objects

The core of the **bayesQR** package consists of the four quantile regression routines that can be called through `bayesQR` as discussed above. However, the package also contains some functions that perform recurring tasks in the context of quantile regression, such as estimating a series of quantile regressions, doing some checks on the MCMC output, create quantile plots, etc. All these functions are written in pure R.

Estimating a series of quantile regressions

One of the advantages of quantile regression is that a more complete view of the response distribution can be obtained. To get this more complete view, one has to estimate several quantile regressions with varying quantile of interest. Also, to calculate the predicted probabilities in the context of binary quantile regression the results of a sequence of binary quantile regressions is required. By specifying a vector of quantiles in the `bayesQR` function, an S3 object of class '`bayesQR`' will be created containing all relevant information of all estimated quantiles. The results are then saved in list format and can then be processed further with other functions such as `summary`, `plot` and `print`.

For example, the following code would save the results of three quantile regressions (first quartile, median and third quartile) in the object `out`.

```
R> out <- bayesQR(y ~ X, quantile = c(0.25, 0.5, 0.75), alasso = FALSE)
```

It is important to understand that in this process, no convergence checks are performed. As always, it is the user who should evaluate the convergence of the MCMC chain. Note that, when estimating a series of quantile regressions, the number of MCMC chains that have to be evaluated can grow fast. Nevertheless, it is important to always make sure that enough MCMC iterations were performed. A visual convergence check can be done, for example, with the traceplots available trough the `plot` function.

Summarize the results of `bayesQR` *with* `summary`

`summary` is an S3 method that summarizes the output object of the `bayesQR` function. The output of the function is an S3 object of class '`bayesQR.summary`'. Basically, it is a list containing information about the MCMC algorithm, the quantile(s) that was (were) estimated, the names of the variables in the model and the number of burn-in draws that were discarded. Also, for every estimated parameter, the Bayes estimate (i.e., the posterior mean) and the 95% posterior credible interval is calculated. When `normal.approx = TRUE`, the `summary` function will also provide the adjusted 95% credible interval based on the normal approximation of the posterior (see Section 4.1). The user has the option to discard a number of burn-in draws or to change the width of the credible interval. When the input object contains information about more than one quantile, the user also can choose which quantiles to summarize. By default all quantiles will be summarized. The S3 method `print` can than be used to output a nicely formatted output of the '`bayesQR.summary`' object. A simple function call is as follows:

```
R> summary(bayesQR.object)
```

The R code underneath the function contains no real difficulties. The code starts with some checks on the input arguments. Then, the code loops through the input object and calculates the summary statistics. The results are then passed to the output object.

Produce traceplots, quantile plots or posterior histograms with `plot`

`plot` is an S3 method that produces traceplots, quantile plots or posterior histograms based on the estimates obtained by the `bayesQR` function. A traceplot shows the evolution of the MCMC draws over the iterations and is often used as a first visual check of the convergence of the MCMC chains. A posterior histogram gives a graphical representation of the marginal posterior distribution of the model parameter based on the MCMC draws. When the normal approximation was requested, the posterior histogram will be overlaid with the density of the normal approximation of the posterior. A quantile plot shows how the value of the regression parameter changes over a range of quantiles. In quantile plots, also the associated credible interval is plotted for every quantile. Again, when the normal approximation was requested, the quantile plot will use the adjusted 95% credible interval for visualizing the uncertainty. Obviously, the more quantiles are estimated, the more precise the resulting plot will be.

The function leaves the option to change some default parameters. For example, the number of burn-in draws that have to be discarded can be adjusted, as well as the width of the credible intervals. That is, the main title, labels of the axes and limits of the horizontal axis can be adjusted. A simple function call for a traceplot and quantile plot respectively, is as follows:

```
R> plot(bayesQR.object, plottype = "trace", burnin = 1000)
R> plot(bayesQR.object, plottype = "quantile", burnin = 1000)
```

The R code underneath the function works as follows. As always, the code starts with some checks on the input arguments. The main input argument is an output object of the `bayesQR` function. Then, the code loops through the input object, that is an S3 object of class '`bayesQR`'. For every estimated quantile and for the desired element of β, the Bayes estimate and the corresponding credible interval is calculated and saved in a temporary matrix. Next, these numbers are plotted and linked by straight lines. Finally, some aesthetic operations are performed, such as representing the credible intervals with a light gray band. The latter is done by considering the credible intervals together with the x-axis as the edges of a polygon. The R function `polygon` is then used to fill the surface of the polygon.

Calculate predicted probabilities with `predict`

To obtain the predicted probabilities of the binary quantile regression model the approach of Kordas (2006) is used. This approach also was applied in Miguéis, Benoit, and Van den Poel (2013) in the context of credit scoring. From the definition of the τth quantile model we have that $\mathsf{P}(y_i^\star > x_i^\top \beta) = 1 - \tau$. The following equation then gives the predicted probability:

$$\mathsf{P}(y_i = 1 | x_i, \beta, \tau) = \int_0^1 I(x_i^\top \beta_\tau \geq 0)\, \mathrm{d}\tau. \tag{25}$$

Kordas (2006) proposes to approximate this conditional probability of a grid of quantile estimates, e.g., $\theta = \{0.05, 0.1, \ldots, 0.95\}$. Now, the element l of Θ has to be found where $\Theta_l < 0$ and $\Theta_{l+1} > 0$:

$$\hat{\tau}_i = \operatorname*{argmin}_{\tau \in \Theta} \{\tau : x_i^\top \beta_\tau \geq 0\}. \tag{26}$$

Then, an interval estimate of the conditional probability of success is given by:

$$\mathsf{P}(y_i = 1 | x_i, \beta, \hat{\tau}_i) = [1 - \hat{\tau}_i, 1 - \hat{\tau}_{i,-1}), \tag{27}$$

where $\hat{\tau}_{i,-1}$ denotes the element in Θ immediately preceding $\hat{\tau}_i$.

For example, if the following grid of quantiles is estimated, $\theta = \{0.05, 0.1, \ldots, 0.95\}$, one should calculate the value of $x_i^\top \beta$ for every quantile in Θ. Then, if this value for observation i is negative for quantiles less than or equal to 0.6 and positive for quantiles 0.65 and above, then $\hat{\tau}_i = 0.65$ and prediction interval is $[0.35, 0.70)$. If the value $x_i^\top \beta$ is negative for all estimated quantiles, then the predicted interval is $(0, 0.05)$. Similarly, the predicted interval is $[0.95, 1)$ if the value $x_i^\top \beta$ is positive for all estimated quantiles.

The S3 method `predict` implements this methodology. The function takes the output argument of the `bayesQR` function as input. Again, the code starts with some checks on the input arguments. Obviously, `predict` will only calculate predicted probabilities for binary quantile regression models. Note that the more estimated quantiles are present in the '`bayesQR`' object, the smaller the predicted interval in Equation 27 will be. However, the `predict` function will not return the interval as such, but the middle value of the interval. It is again possible to discard a number of burn-in draws in the input object. Finally, a matrix of predictor variables has to be provided. It is clear that this matrix should contain the same variables as the matrix used to estimate the parameters. The output is a vector containing the predicted probabilities.

5. Data illustrations

In this section, it will be shown how to apply the methods and functions contained in the **bayesQR** package to real life data. Two datasets are included in the package, that is one dataset with a continuous dependent variable (i.e., prostate cancer data) and one dataset with a binary dependent variable (i.e., customer churn data). This section is organized as follows: First, the quantile regression models for continuous dependent variables are demonstrated on the prostate cancer dataset. Next, the quantile regression models for binary dependent variables are applied to the customer churn dataset. In both examples, it will be shown how to use the "support functions", for example, summarizing and plotting the results or calculating the predicted probabilities.

5.1. Example 1: The prostate cancer dataset

The prostate cancer dataset was first reported by Stamey *et al.* (1989) and was analyzed in many subsequent studies, for example Tibshirani (1996), Al-Hamzawi *et al.* (2012), etc. This dataset consists of the medical records of 97 male patients who were to receive a radical prostatectomy. The response variable is the level of prostate antigen (`lpsa`) and the dataset contains eight predictor variables. These predictor variables include: log of cancer volume (`lcavol`), log of prostate weight (`lweight`), `age`, log of the amount of benign prostatic hyperplasia (`lbph`), seminal vesicle invasion (`svi`), log of capsular penetration (`lcp`), Gleason score and percentage of Gleason scores 4 or 5 (`pgg45`).

We start by analyzing this dataset with the `bayesQR` function without adaptive lasso. For numerical stability it is better to first standardize the predictors as well as the dependent variable. An intercept is desired, and will automatically be included using the `formula` notation. In the current example, the median is modeled. Note that the "median" is the default quantile and that by default "adaptive lasso" is not used, so specifying those parameters is not necessary. Also, since the "normal approximation" of the posterior is the default, this is automatically calculated without explicitly requesting it. The prior is not specified here and this will result in the default dispersed prior on the model parameters. The number of MCMC iterations is set to 5000. Posterior convergence checks will show whether this is enough to find convergence in the MCMC chain. If not, this value has to be increased. Finally, all arguments have to be passed to the `bayesQR` function.

```
R> library("bayesQR")
R> data("Prostate", package = "bayesQR")
R> str(Prostate)
```

```
'data.frame':   97 obs. of  9 variables:
 $ lcavol : num  -0.58 -0.994 -0.511 -1.204 0.751 ...
 $ lweight: num  2.77 3.32 2.69 3.28 3.43 ...
```

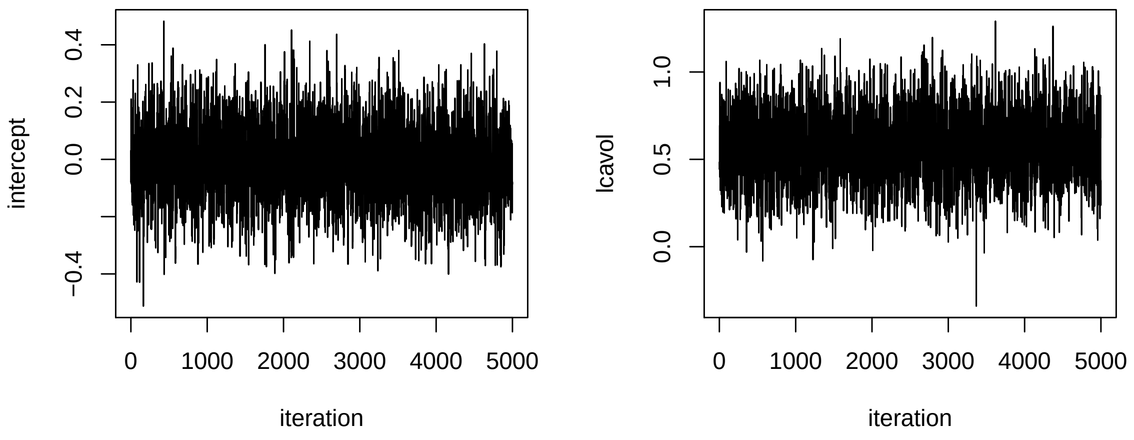

Figure 1: Traceplots of the MCMC chains for intercept (left panel) and log of the cancer volume (`lcavol`, right panel) for quantile regression without adaptive lasso.

```
 $ age    : num  50 58 74 58 62 50 64 58 47 63 ...
 $ lbph   : num  -1.39 -1.39 -1.39 -1.39 -1.39 ...
 $ svi    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ lcp    : num  -1.39 -1.39 -1.39 -1.39 -1.39 ...
 $ gleason: num  6 6 7 6 6 6 6 6 6 6 ...
 $ pgg45  : num  0 0 20 0 0 0 0 0 0 0 ...
 $ lpsa   : num  -0.431 -0.163 -0.163 -0.163 0.372 ...
```

```
R> y <- Prostate$lpsa
R> y <- scale(y)
R> X <- Prostate[, 1:8]
R> X <- scale(X)
R> out <- bayesQR(formula = y ~ X, quantile = 0.5, alasso = FALSE,
+    ndraw = 5000)
```

On a recent computer, this code executes in less than a second. During execution, information is plotted to the console about the status of the current iteration. This can be useful in large scale applications that take a long time to estimate. Note again that on some operating systems or IDE's this information will only be displayed when execution has finished. In that case, this feature is otiose.

A visual check about the MCMC convergence can be done by inspecting the traceplots. The `plot` routine provides this functionality with a single command. To see the next traceplot the user has to type "`y`" and press return. Typing "`n`" will bring the user back to the console.

```
R> plot(out, plottype = "trace")
```

Figure 1 gives the traceplots of the intercept and the log of the cancer volume. The traceplots of the other variables are very similar, so we do not show the other six traceplots here. The plots show that the MCMC sampler quickly moves to the stationary distribution and mixes very well. Many formal checks are developed to formally check the convergence of MCMC

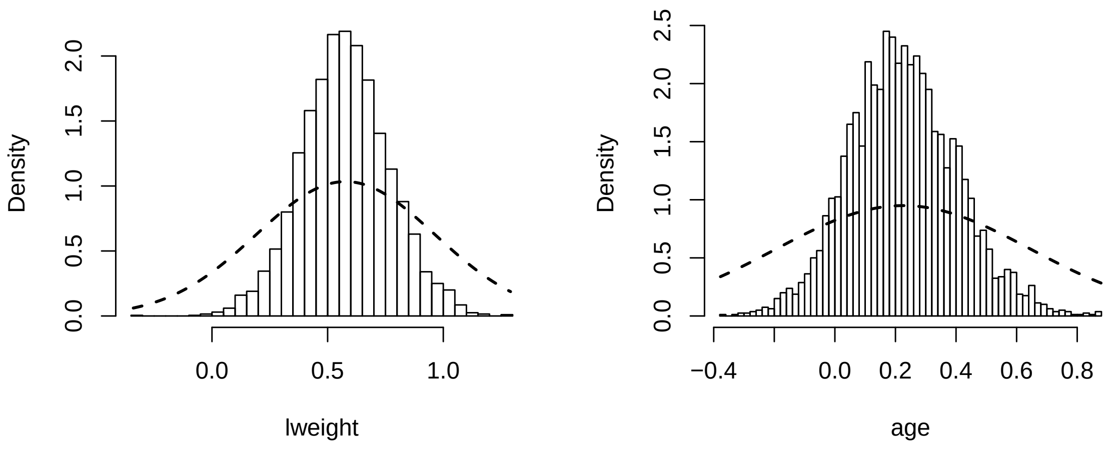

Figure 2: Posterior histograms for log of the weight (`lweight`, left panel) and age (`age`, right panel) for quantile regression without adaptive lasso. The histograms are overlaid with the density of the normal approximation of the posterior distribution.

chains. However, these tools are outside the range of the **bayesQR** package. The well-known R package **coda** (Plummer, Best, Cowles, Vines, Sarkar, and Almond 2016; Plummer, Best, Cowles, and Vines 2006) serves this goal very well.

The marginal posterior distributions can be visualized by plotting the histograms of the simulated draws. The `plot` routine provides this functionality with a single command. As was the case for the traceplots, to see the next histogram the user has to type "`y`" and press return. Typing "`n`" will bring the user back to the console.

```
R> plot(out, plottype = "hist", burnin = 1000)
```

As an example, Figure 2 gives the posterior histograms of the log of the weight and age. To make sure only draws from the stationary posterior distribution are plotted, one thousand draws were excluded as burn-in draws.

Often, the posterior distribution has to be summarized in a limited number of key statistics. The following command gives us the Bayes estimate of the posterior and the posterior 80% credible intervals. Nevertheless the traceplots showed that the effect of the initial values of the MCMC chains wears off very fast, we decide to reject the first 1000 draws as burn-in draws.

```
R> results <- summary(out, burnin = 1000, credint = c(0.1, 0.9))
R> results
```

The above function call prints the following information to the screen:

```
Type of dependent variable: continuous
Lasso variable selection: no
Normal approximation of posterior: yes
Estimated quantile:  0.5
```

```
Lower credible bound:  0.1
Upper credible bound:  0.8
Number of burnin draws:  1000
Number of retained draws:  4000


Summary of the estimated beta:

            Bayes Estimate    lower   upper adj.lower adj.upper
(Intercept)       -0.00798 -0.16890  0.0993    -0.354     0.220
Xlcavol            0.57760  0.34529  0.7333     0.083     0.902
Xlweight           0.22568  0.00718  0.3756    -0.313     0.579
Xage              -0.14612 -0.33120 -0.0236    -0.541     0.114
Xlbph              0.16455 -0.03626  0.2978    -0.289     0.462
Xsvi               0.28937  0.04305  0.4470    -0.264     0.653
Xlcp              -0.17041 -0.47730  0.0318    -0.824     0.259
Xgleason           0.03414 -0.23443  0.2013    -0.524     0.401
Xpgg45             0.14994 -0.13566  0.3417    -0.467     0.555
```

Printing a '`bayesQR.summary`' object first gives some general information about the model that was estimated. Then, the estimated parameters are summarized. The first column summarizes the posterior distribution in one point estimate, i.e., the Bayes estimate or posterior mean. The second and third column give the 80% credible interval and can be interpreted as "the probability is 80% that the true value of the parameter is in this interval". When the "normal approximation" is requested (which is the default option), then the last two columns give the adjusted credible intervals based on the normal approximation of the posterior distribution. The user can change the width of this credible interval, for example, by default the function would return the conventional 95% credible interval.

Next, we will estimate a quantile regression model with adaptive lasso penalty on the same dataset. The only thing that has to be changed is set the logical indicator for the adaptive lasso to `TRUE`. Note that the quantile should not be specified because we are interested in the (default) median regression. Again no prior object is passed to the `bayesQR` function, and this implies that the researcher agrees on the default vague prior. As always, the first thing to do after invoking one of the quantile regression routines is to check convergence.

```
R> out2 <- bayesQR(formula = y ~ X, alasso = TRUE, ndraw = 5000)
R> plot(out2, plottype = "trace")
```

Again only the first two traceplots are shown here because these are representative for the rest of the plots (see Figure 3). Also the MCMC output for the median regression with adaptive lasso variable selection function shows good convergence and mixing. The initial values wear off very fast and quickly the chain stabilizes on the stationary distribution.

The real strength of quantile regression is that it makes it possible to investigate the entire response distribution and not only the expected mean (as OLS does). The entire response distribution can be analyzed by estimating a whole family of quantile regressions. This can easily be done by supplying a vector of quantiles to the `bayesQR` function. In the current example, the following sequence of quantiles will be estimated: $\tau = \{0.05, 0.1, \ldots, 0.95\}$. Also, a prior object is again not specified and thus the standard vague prior will be used.

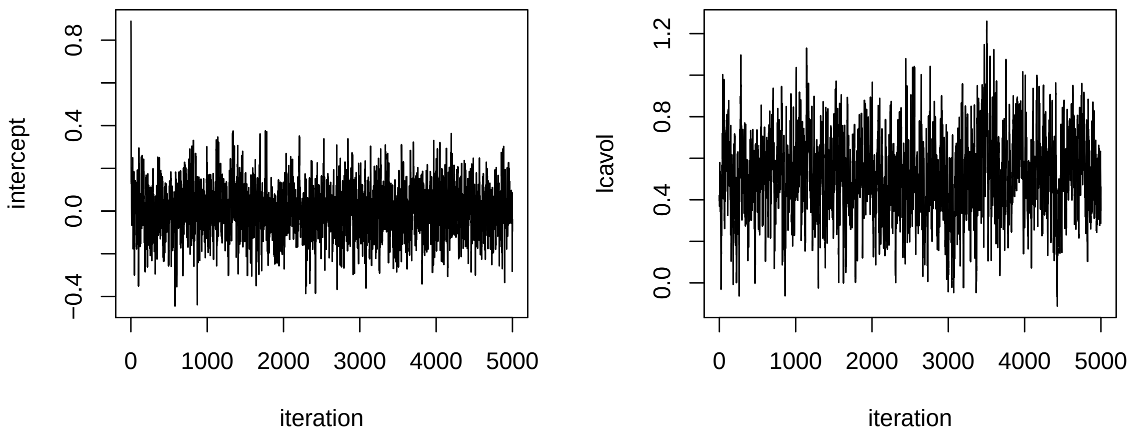

Figure 3: Traceplots of the MCMC chains for intercept (left panel) and log of the cancer volume (`lcavol`, right panel) for median regression with adaptive lasso.

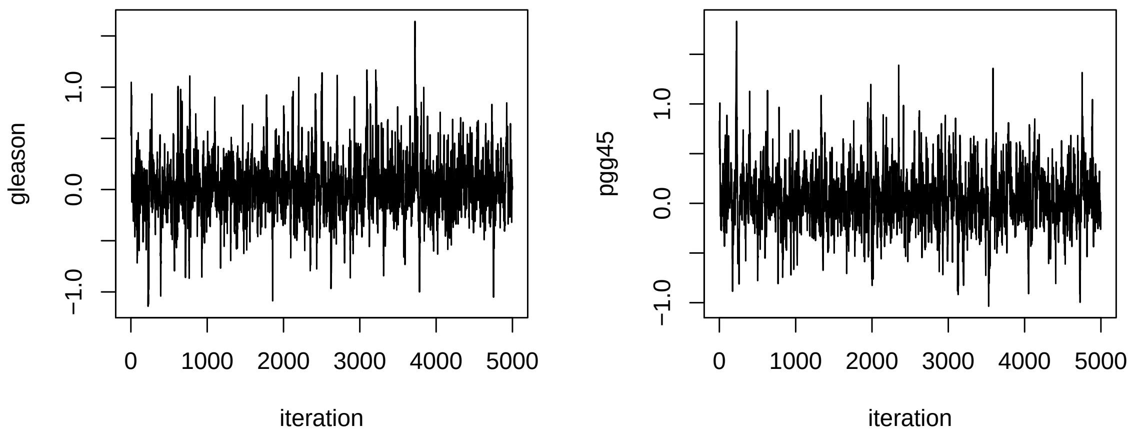

Figure 4: Traceplots of the MCMC chains for Gleason score (left panel) and percentage Gleason scores 4 or 5 (`pgg45`, right panel) for quantile $\tau = 0.05$ using adaptive lasso.

```
R> quantvec <- seq(0.05, 0.95, 0.05)
R> out3 <- bayesQR(formula = y ~ X, quantile = quantvec, alasso = TRUE,
+    ndraw = 5000)
```

Executing this code will estimate 19 quantile regressions. Again, information is printed to the console about the current status of the procedure. We have now estimated 19 times 9 β coefficients. It might be tempting to skip the investigation of the MCMC output, but it is very important not to do so.

```
R> plot(out3, plottype = "trace")
```

In the more extreme quantiles, that is the very low or very high quantiles, it is possible that the MCMC chain converges slower than in the middle quantiles. This can be seen from Figure 4.

Clearly, the MCMC chains in Figure 4 are less thick or dense compared to the ones in Figure 1 or Figure 3. This means that the chains suffer more from autocorrelation and that more draws are needed to fully cover the posterior distribution. In this case the situation does not seem very problematic, but the user should always assess if there is a problem or not. Usually, increasing the number of MCMC draws resolves the issue. Again, note that the **coda** R package contains more formal checks for MCMC output (Plummer *et al.* 2016).

Now that we have assessed the quality of our MCMC chains, we can use the output object of the `bayesQR` procedure to create the quantile plots. The `plot` procedure makes this relatively straightforward. It simply takes the output object of the `bayesQR` procedure as input and requires the index of β of interest to be specified. Additional parameters that can be passed to the routine are the number of burn-in draws to discard, the labels of the horizontal of vertical axis or the width of the credible interval. The following code generates the plot for the intercept (the first element of β):

```
R> plot(out3, var = 1, plottype = "quantile", burnin = 1000,
+    ylab = "intercept")
```

Often, it is interesting to compare the quantile regression results with the results of OLS. The following code calculates and prints the OLS estimate of the prostate cancer dataset.

```
R> out4 <- lm(y ~ X)
R> coef(out4)
```

Once the quantile graph is plotted, some simple statements allow to add additional information to the plot. For example, here we will add a full line to indicate zero and a dotted line to indicate the OLS estimate. The following code does this for the first quantile plot.

```
R> abline(h = 0, lty = 1)
R> abline(h = coef(out4)[1], lty = 3)
```

Figure 5 presents the quantile plots of four variables in the model. To save space, not all nine plots are printed here. The following code generates the quantile plots for four selected variables:

```
R> oldpar <- par(mfrow = c(2, 2))
R> for (i in c(1, 3, 4, 6)) {
+    plot(out3, var = i, plottype = "quantile", burnin = 1000)
+    abline(h = 0, lty = 1)
+    abline(h = coef(out4)[i], lty = 3)
+ }
R> par(oldpar)
```

The intercept in Figure 5 takes on very different values for the different quantiles. For the lower quantiles the intercept is negative, while for the higher quantiles it is positive. As a result, both the OLS estimate as well as the median estimate are close to zero. This means that for average values of the predictors, the expected value of the log of prostate-specific antigen for the lower quantiles is clearly below average (and vice versa). This is very typical behavior of the intercept in quantile regression. According to the OLS estimate, the predictor

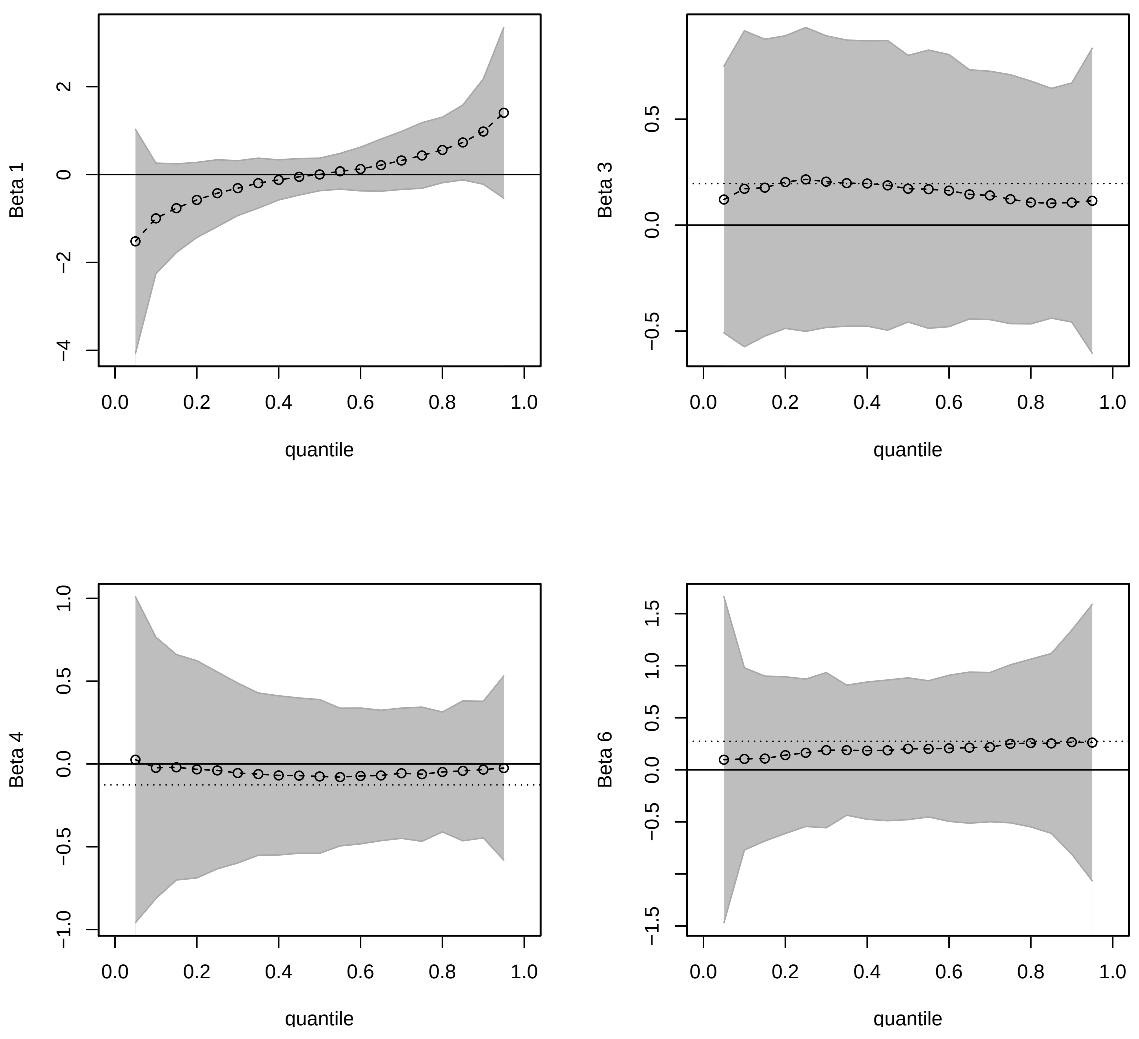

Figure 5: Quantile plots for the intercept (upper left panel), age (upper right panel), log of the weight (lower left panel) and seminal vesicle invasion (lower right panel) for quantile regression with adaptive lasso. Dotted lines represent the OLS estimate. The shaded area gives the adjusted credible intervals as the parameters were estimated with the default `normal.approx = TRUE` option.

age has a negative effect on the dependent variable. This is also true for the median regression estimate. However, the full quantile plot indicates that this variable is not distinguishable from zero for almost the entire quantile range. This is also true for the other two variables in the plot. Clearly, the added value of using quantile regression for these predictors is rather limited.

5.2. Example 2: The customer churn dataset

For this example, the dataset `Churn` contained in the **bayesQR** package is used. This dataset is a stratified random sample from all active customers, at the end of June 2006, of a European financial services company. The dataset is a small subset of 400 observations of the dataset analyzed in Benoit and Van den Poel (2013). The dependent variable in this dataset is the `churn` behavior of the customers in the period from 2006-07-01 until 2006-12-31. Here a churned customer is defined as someone who closed all his/her bank accounts with the company.

The dataset contains a small set of four predictor variables. That is, gender with male coded as one (`gender`), social class score where a higher score means a higher social class (`Social_Class_Score`), length of relationship denoting the number of days since first purchase (`lor`) and recency defined as the number of days since last purchase (`recency`). Note that all predictor variables are standardized both for numerical stability as well as confidentiality considerations.

The following code loads the customer churn dataset into memory and displays its contents. The code for estimating a binary quantile regression model is very similar as in the previous example. The `bayesQR` function automatically detects whether the dependent variable is binary or continuous and chooses the correct algorithm. Again, we will not specify any prior values so that the default vague priors will be used. Note that the dot in the formula means "all variables except the specified dependent variable" and the "`0 +`" means that the model should not have an intercept.

```
R> data("Churn", package = "bayesQR")
R> str(Churn)

'data.frame':        400 obs. of  5 variables:
 $ churn             : int  0 0 0 0 0 0 0 0 0 0 ...
 $ gender            : int  1 0 1 1 1 0 1 1 0 1 ...
 $ Social_Class_Score: num  -0.116 -0.313 -1.325 1.967 1.455 ...
 $ lor               : num  -1.0892 1.183 -0.8462 0.0869 -1.1666 ...
 $ recency           : num  -0.721 3.634 -0.428 -0.536 -0.673 ...

R> out5 <- bayesQR(churn ~ 0 + ., data = Churn, quantile = 0.05,
+    ndraw = 5000)
R> plot(out5, plottype = "trace")
```

As with the other quantile regression functions, during execution of the algorithm, information is printed to the console about the progress of the routine. As always, it is important to check the convergence of the MCMC chains. The left panel of Figure 6 gives the traceplot of the variable gender using 5000 iterations and estimating the fifth percentile. Although, the MCMC chain finds the stationary distribution very fast, the MCMC chain is rather thin. This indicates that the chain suffers from autocorrelation, as often happens in latent variable models and even more when estimating extreme quantiles. Again, the **coda** R package contains some more formal checks for MCMC chains (Plummer *et al.* 2016). This can be solved by increasing the number of MCMC iterations. The right panel of Figure 6 gives the traceplot of the same variable, but now using 50000 iterations and keeping every tenth draw. This shows how increasing the number of MCMC draws increases the quality of the chain.

```
R> out5 <- bayesQR(churn ~ 0 + ., data = Churn, quantile = 0.05,
+    ndraw = 50000, keep = 10)
R> plot(out5, plottype = "trace")
```

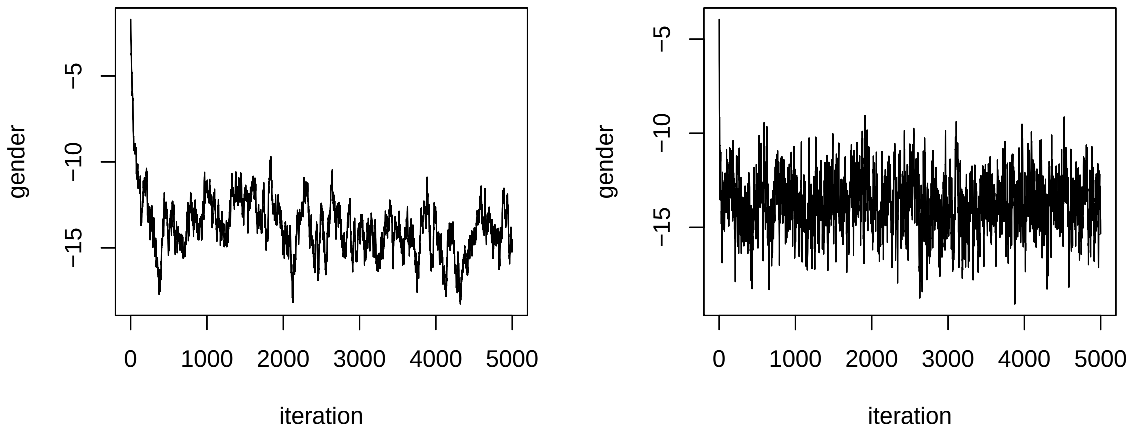

Figure 6: Traceplots of MCMC chain for gender using 5000 iterations (left panel) and gender using 50000 iterations (keeping every 10th draw, right panel) for binary quantile regression with $\tau = 0.05$ and no adaptive lasso.

	Default vague prior			Informative prior		
	Bayes est.	Lower	Upper	Bayes est.	Lower	Upper
`gender`	−0.1734	−0.507	0.170	7.29	6.75	7.82
`Social_Class_Score`	0.0184	−0.204	0.238	5.63	5.15	6.17
`lor`	−0.7961	−1.165	−0.452	4.03	3.51	4.54
`recency`	0.6525	0.375	0.962	6.70	6.12	7.32

Table 1: Output of `summary`: Bayes estimate and 95% credible interval of binary median regression with a vague versus an informative prior on the `Churn` dataset.

The following code estimates median binary regressions on the churn dataset. Two output objects are created, one with the default vague prior on β and one with an (unrealistic) informative prior on β. For the unrealistic prior, we will assume that the researcher's uncertainty about β can be represented as a normal distribution with $\mu = 10$ and variance $\sigma^2 = 0.1$.

```
R> myprior <- prior(churn ~ 0 + ., data = Churn, alasso = FALSE,
+    beta0 = rep(10, 4), V0 = diag(4) * 0.1)
R> out6 <- bayesQR(churn ~ 0 + ., data = Churn, ndraw = 5000)
R> out7 <- bayesQR(churn ~ 0 + ., data = Churn, prior = myprior,
+    ndraw = 5000)
```

The `plot` function can be used to first check the MCMC chains and the `summary` function to print the Bayes estimates and the posterior credible intervals.

```
R> plot(out6, plottype = "trace")
R> plot(out7, plottype = "trace")
R> summary(out6)
R> summary(out7)
```

Table 1 gives the output of printing the summary objects of both models. It reveals large differences between both summary results. For the default vague prior, only lor and recency have most of their posterior mass away from zero. In contrast, for the unrealistic informative prior the results show that the Bayes estimates are pulled strongly to the prior expectations. This results from the fact that the prior has most of its mass around the value of 10. It is clear that the researcher should always think about the prior that is used in the analysis to avoid undesired effects. Also it often is a good practice to check the robustness of the results over different (realistic) prior values.

According to the Bayesian paradigm, everything one can know about the parameters of interest is contained in the posterior distribution. This means that if a researcher is interested in certain hypotheses about the parameters, he/she has to investigate the posterior distribution. If, for example, one is interested in the probability that the regression parameter for Social Class Score is greater than zero, $\mathsf{P}(\beta_{\text{Social\_Class\_Score}} > 0)$, then this probability can be approximated by calculating the relative number of posterior draws for $\beta_{\text{Social\_Class\_Score}}$ that are larger than zero.

```
R> mean(out6[[1]]$betadraw[, 2] > 0)

[1] 0.5704
```

In a similar way the probability that one parameter is larger than another can be calculated. For example, if one is interested in the probability that the regression parameter for gender is greater than the regression parameter for Social Class Score, $\mathsf{P}(\beta_{\text{gender}} > \beta_{\text{Social\_Class\_Score}})$, then this probability can be approximated by calculating the relative number of posterior draws for β_{gender} that are larger than the posterior draws for $\beta_{\text{Social\_Class\_Score}}$.

```
R> mean(out6[[1]]$betadraw[, 1] > out6[[1]]$betadraw[, 2])

[1] 0.1678
```

Following the same reasoning, the posterior distributions of the parameters for different quantiles can be compared. That is, one can approximate the probability that the regression parameter for length of relationship estimated for the first quartile is larger than that for the third quartile by calculating the relative number of posterior draws for $\beta_{\text{lor}}(\tau = 0.25)$ that are larger than the posterior draws for $\beta_{\text{lor}}(\tau = 0.75)$.

```
R> out8 <- bayesQR(formula = y ~ X, quantile = c(0.25, 0.75), ndraw = 5000)
R> mean(out8[[1]]$betadraw[, 3] > out8[[2]]$betadraw[, 3])

[1] 0.61
```

It is important to note, however, that due to the specific nature of the Bayesian ALD approach these calculations should be interpreted with caution. The discussion in Section 6 treats this issue in more detail.

The full advantage of quantile regression becomes clear when estimating a whole family of quantile regressions. Again, this can simply be done by specifying a vector of quantiles in the `bayesQR` function. Again, the quantiles are estimated over the range $\tau = \{0.05, 0.1, \ldots, 0.95\}$. As always, the first thing to do after running the `bayesQR` procedure is to check the quality of the MCMC output. If the chains converged as expected, we can have a closer look at the quantile plots.

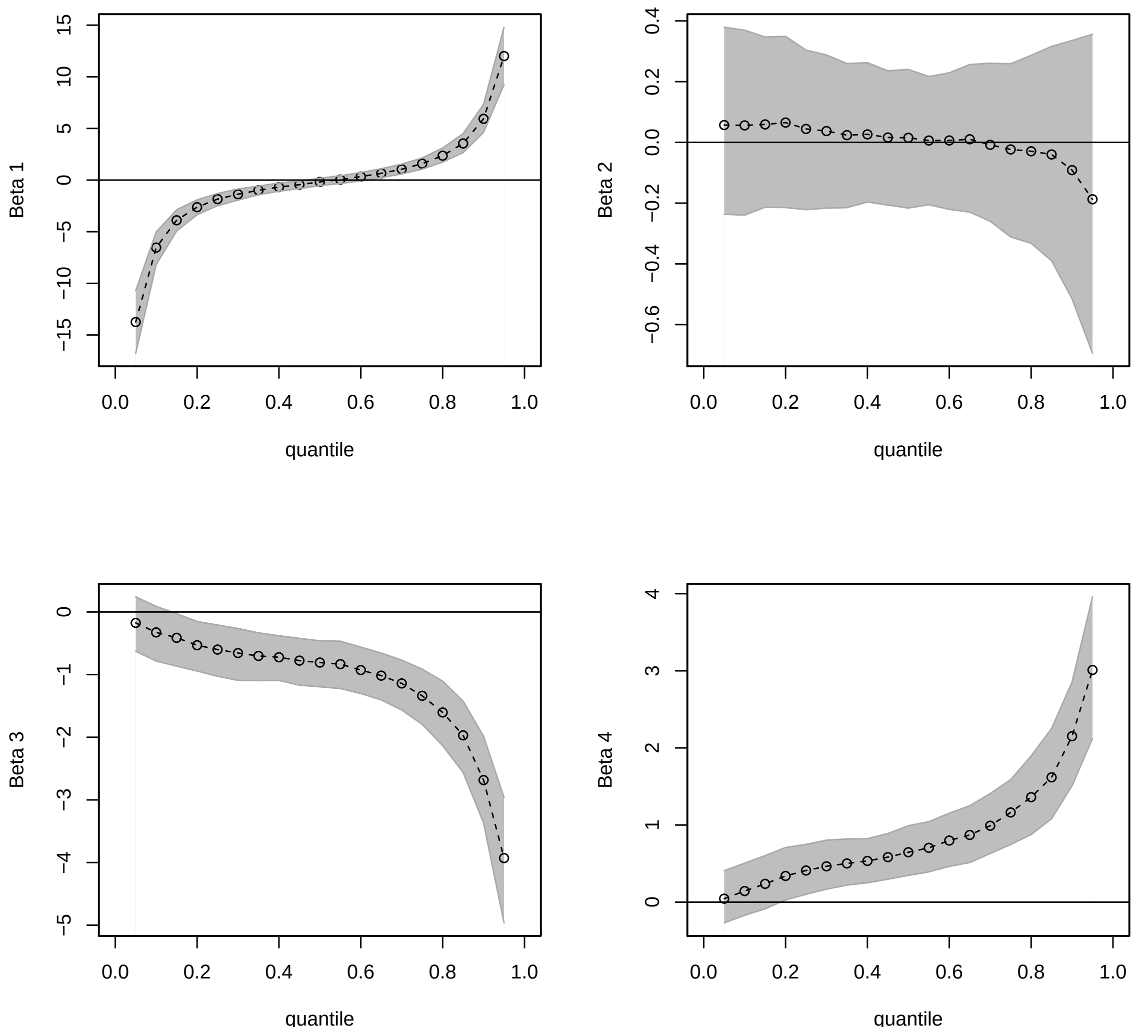

Figure 7: Quantile plots for age (upper left panel), length of relationship (upper right panel), social class score (lower left panel) and recency (lower right panel) for binary quantile regression without adaptive lasso.

```
R> out9 <- bayesQR(churn ~ 0 + ., data = Churn, quantile = quantvec,
+    ndraw = 5000)
R> plot(out9, plottype = "trace")
R> oldpar <- par(mfrow = c(2, 2))
R> for (i in 1:4) {
+    plot(out9, var = i, plottype = "quantile")
+  }
R> par(oldpar)
```

Figure 7 shows the quantile plots for the variables in the `Churn` dataset. The variable age is insignificant for the expected median response, but the quantile plots show that is has a negative effect for the lower quantiles and a positive effect for the higher quantiles. This shows that age is a variable that should not be neglected, a conclusion that is totally missed when

	Predicted	Binary QR False	Binary QR True	Logit False	Logit True
Actual	False	102	98	102	98
	True	49	151	48	152

Table 2: Confusion matrices for binary quantile regression without adaptive lasso versus logistic regression on the `Churn` dataset.

focusing on the expected value of the response. The social class score of the customer turns out not to be different from zero for the entire quantile range. The length of relationship has a negative effect on the entire quantile range, however, the effect becomes more pronounced for the higher quantiles. The same is true for recency, but with opposite sign. This variable has a positive effect on churn and becomes more and more important for higher quantiles.

Finally, the **bayesQR** package also contains a function to calculate the predicted probabilities for binary quantile regression methods, i.e., the S3 method `predict`. This function takes the output object of the `bayesQR` function as input, that is an S3 object of class '`bayesQR`', together with a matrix of predictor variables. Note that for a meaningful interpretation of the predicted probabilities, the results of at least 9 quantiles should be used (see Section 4.5). Also, it is clear that the interpretation of the predictor matrix should be the same as the matrix used to estimate the model parameters. Finally, it is possible to discard a number of burn-in draws when calculating predicted probabilities.

The following code calculates the predicted probabilities on the customer churn dataset using the series of quantile regressions that were estimated above. In addition, a simple logit regression is estimated and the predicted probabilities are also saved in a vector. The predictive performance of both models is then compared using the confusion matrices.

```
R> pred <- predict(out9, X = Churn[, 2:5], burnin = 1000)
R> table(Churn$churn, as.numeric(pred > 0.5))


      0   1
  0 102  98
  1  48 152


R> mylogit <- glm(churn ~ 0 + ., data = Churn, family = binomial(logit))
R> table(Churn$churn, as.numeric(mylogit$fit > 0.5))


      0   1
  0 102  98
  1  48 152
```

Table 2 compares the confusion matrices of both approaches. The performance of both methods is very similar on this dataset. The logit model and the binary quantile regression method lead to exactly the same confusion matrix. It should be noted that it is not necessarily the case that the predictive performance of both methods is similar in other applications (see Miguéis *et al.* 2013).

6. Discussion

As the literature study showed, Bayesian quantile regression based on the ALD has been investigated intensively. Nevertheless, a number of shortcomings are worthwhile discussing. Most notably is the discussion of the deliberate misspecification of the likelihood on the implications for posterior consistency and the coverage probabilities of the resulting posterior distributions. This issue has been discussed in detail in Section 3.4.

Another drawback of the described approaches is that, as quantiles are fitted separately, the conditional quantile curves are not smooth and the fitted regression lines may cross, which violates the basic probabilistic rule and causes problems for inference in practice. The usual approach to deal with this problem is to simultaneously fit several quantiles (e.g., Taddy and Kottas 2010). These approaches are not implemented in the package as many of these methods give raise to new problems in terms of computation or data requirements (Sriram *et al.* 2012). However, the applied researcher will typically be interested in well separated quantiles and then quantile crossing is less likely to occur.

Finally, it should be noted that in making inferences about the relation between $\beta(\tau_1)$ and $\beta(\tau_2)$ (e.g., $\mathsf{P}(\beta(\tau_1) > \beta(\tau_2))$) the methods presented in this manuscript assume independence of $\beta(\tau_1)$ and $\beta(\tau_2)$. Whenever this assumption is hard to justify, the proposed methods are not well suited to conduct inference about the relation between $\beta(\tau_1)$ and $\beta(\tau_2)$. However, it remains unclear how problematic a possibly incorrect assumption of independence is. An interesting path for further research is to investigate to what extent this leads to biased inferences of the type $\mathsf{P}(\beta(\tau_1) > \beta(\tau_2))$.

The goal of the **bayesQR** package is to provide a fast and high-quality implementation of the Bayesian ALD approach to quantile regression. Still, it is important that researchers using the **bayesQR** package are aware of these limitations when using this software and drawing conclusions from it.

7. Conclusions

This paper presents the R package **bayesQR** for Bayesian estimation of quantile regression. The package contains methods for both continuous as well as binary dependent variables. For both types of dependent variables, the user can choose between normal estimation and estimation with adaptive lasso penalty. The MCMC estimation makes use of an efficient Gibbs sampling algorithm. In addition, the MCMC part of the procedures is programmed in Fortran to speed up the computational time. The latter can be regarded as one of the main advantages of this implementation.

Next to the core algorithms of the package, also some support functions are provided. That is, functions to analyze the MCMC chains, summarize the results, estimate a whole family of quantile regressions, plot the quantile processes or plot the MCMC trace or marginal posterior histograms. For the binary dependent variables, also a function is provided that calculates the predicted probabilities.

In data examples it was shown that the **bayesQR** package is a convenient way to estimate quantile regressions using the Bayesian framework. Future versions of the package will focus on computational efficiency and possibly other types of data will be supported too (e.g., Poisson, etc.).

References

Al-Hamzawi R (2012). ***Brq**: Bayesian Estimation and Variable Selection for Quantile Regression Models.* R package version 1.0, URL `https://CRAN.R-project.org/src/contrib/Archive/Brq/`.

Al-Hamzawi R, Yu K, Benoit DF (2012). "Bayesian Adaptive Lasso Quantile Regression." *Statistical Modelling*, **12**(3), 279–297. `doi:10.1177/1471082x1101200304`.

Anderson E, Bai Z, Bischof C, Blackford S, Demmel J, Dongarra J, Croz JD, Greenbaum A, Hammerling S, McKenney A, Sorensen D (1999). ***LAPACK** Users' Guide.* Society for Industrial and Applied Mathematics, Philadelphia, 3rd edition.

Barrodale I, Roberts FDK (1973). "An Improved Algorithm for Discrete l_1 Linear Approximations." *SIAM Journal of Numerical Analysis*, **10**(5), 839–848. `doi:10.1137/0710069`.

Benoit DF, Al-Hamzawi R, Yu K, Van den Poel D (2017). ***bayesQR**: Bayesian Quantile Regression.* R package version 2.3, URL `https://CRAN.R-project.org/package=bayesQR`.

Benoit DF, Van den Poel D (2012). "Binary Quantile Regression: A Bayesian Approach Based on the Asymmetric Laplace Distribution." *Journal of Applied Econometrics*, **27**(7), 1174–1188. `doi:10.1002/jae.1216`.

Benoit DF, Van den Poel D (2013). "Quantile Regression for Database Marketing: Methods and Applications." In K Coussement, KW De Bock, SA Neslin (eds.), *Advanced Database Marketing: Innovative Methodologies and Applications for Managing Customer Relationships*. Gower Publishing, London.

Box GEP, Muller ME (1958). "A Note on the Generation of Random Normal Deviates." *The Annals of Mathematical Statistics*, **29**(2), 610–611. `doi:10.1214/aoms/1177706645`.

Chernozhukov V, Hong H (2003). "An MCMC Approach to Classical Estimation." *Journal of Econometrics*, **115**(2), 293–346. `doi:10.1016/s0304-4076(03)00100-3`.

Devroye L (1986). *Non-Uniform Random Variate Generation.* Springer-Verlag, New York. `doi:10.1007/978-1-4613-8643-8`.

Dunson DB, Watson M, Taylor JA (2003). "Bayesian Latent Variable Models for Median Regression on Multiple Outcomes." *Biometrics*, **59**(2), 296–304. `doi:10.1111/1541-0420.00036`.

Geweke J (1989). "Efficient Simulation from the Multivariate Normal and Student-t Distributions Subject to Linear Constraints." In EM Keramidas (ed.), *Computing Science Statistics: Proceedings of the 23rd Symposium on the Interface*, pp. 571–578. Interface Foundation of North America, Fairfax.

Ji Y, Lin N, Zhang B (2012). "Model Selection in Binary and Tobit Quantile Regression Using the Gibbs Sampler." *Computational Statistics & Data Analysis*, **56**(4), 827–839. `doi:10.1016/j.csda.2011.10.003`.

Jørgensen B (1982). *Statistical Properties of the Generalized Inverse Gaussian Distribution.* Lecture Notes in Statistics. Springer-Verlag, New York. doi:10.1007/978-1-4612-5698-4.

Koenker R (2004). "Quantile Regression for Longitudinal Data." *Journal of Multivariate Analysis*, **91**(1), 74–89. doi:10.1016/j.jmva.2004.05.006.

Koenker R (2005). *Quantile Regression.* Cambridge University Press, Cambridge.

Koenker R (2016). ***quantreg**: Quantile Regression and Related Methods.* R package version 5.29, URL https://CRAN.R-project.org/package=quantreg.

Koenker R, Basset G (1978). "Regression Quantiles." *Econometrica*, **46**(1), 33–50. doi:10.2307/1913643.

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.

Kordas G (2006). "Smoothed Binary Regression Quantiles." *Journal of Applied Econometrics*, **21**(3), 387–407. doi:10.1002/jae.843.

Kottas A, Gelfand AE (2001). "Bayesian Semiparametric Median Regression Modeling." *Journal of the American Statistical Association*, **96**(456), 1458–1468. doi:10.1198/016214501753382363.

Kozumi H, Kobayashi G (2011). "Gibbs Sampling Methods for Bayesian Quantile Regression." *Journal of Statistical Computation and Simulation*, **81**(11), 1565–1578. doi:10.1080/00949655.2010.496117.

Lancaster T, Jun SJ (2010). "Bayesian Quantile Regression Methods." *Journal of Applied Econometrics*, **25**(2), 287–307. doi:10.1002/jae.1069.

Li Q, Xi R, Lin N (2010). "Bayesian Regularized Quantile Regression." *Bayesian Analysis*, **5**(3), 533–556. doi:10.1214/10-ba521.

Manski CF (1975). "Maximum Score Estimation of the Stochastic Utility Model of Choice." *Journal of Econometrics*, **3**(3), 2–16. doi:10.1016/0304-4076(75)90032-9.

Manski CF (1985). "Semiparametric Analysis of Discrete Response: Asymptotic Properties of the Maximum Score Estimator." *Journal of Econometrics*, **27**(3), 313–333. doi:10.1016/0304-4076(85)90009-0.

Marsaglia G, Tsang WW (2000). "A Simple Method for Generating Gamma Variables." *ACM Transactions on Mathematical Software*, **26**(3), 363–372. doi:10.1145/358407.358414.

Martin AD, Quinn KM, Park JH (2011). "**MCMCpack**: Markov Chain Monte Carlo in R." *Journal of Statistical Software*, **42**(9), 1–22. doi:10.18637/jss.v042.i09.

Michael J, Schucany W, Haas R (1976). "Generating Random Variables Using Transformations with Multiple Roots." *The American Statistician*, **30**(2), 88–90. doi:10.2307/2683801.

Miguéis V, Benoit DF, Van den Poel D (2013). "Enhanced Decision Support in Credit Scoring Using Bayesian Binary Quantile Regression." *Journal of the Operational Research Society*, **64**(9), 1374–1383. doi:10.1057/jors.2012.116.

Mosteller F, Tukey J (1977). *Data Analysis and Regression: A Second Course in Statistics*. Addison-Wesley, Reading.

Plummer M, Best N, Cowles K, Vines K (2006). "**coda**: Convergence Diagnosis and Output Analysis for MCMC." *R News*, **6**(1), 7–11. URL https://www.R-project.org/doc/Rnews/.

Plummer M, Best N, Cowles K, Vines K, Sarkar D, Almond R (2016). ***coda**: Output Analysis and Diagnostics for MCMC*. R package version 0.19-1, URL https://CRAN.R-project.org/package=coda.

Press WH, Teukolsky SA, Vetterling WT, Flannery BP (1996). *Numerical Recipes in Fortran 90*, volume 2. 2nd edition. Cambridge University Press, Cambridge.

R Core Team (2016). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Sriram K (2015). "A Sandwich Likelihood Correction for Bayesian Quantile Regression Based on the Misspecified Asymmetric Laplace Density." *Statistics & Probability Letters*, **107**, 18–26. doi:10.1016/j.spl.2015.07.035.

Sriram K, Ramamoorthi RV, Ghosh P (2012). "Simultaneous Bayesian Estimation of Multiple Quantiles with an Extension to Hierarchical Models." *IIM Bangalore Research Paper*, **359**. doi:10.2139/ssrn.2117722.

Sriram K, Ramamoorthi RV, Ghosh P (2013). "Posterior Consistency of Bayesian Quantile Regression Based on the Misspecified Asymmetric Laplace Density." *Bayesian Analysis*, **8**(2), 269–504. doi:10.1214/13-ba817.

Stamey TA, Kabalin JN, McNeal JE, Johnstone IM, Freiha F, Redwine EA, Yang N (1989). "Prostate Specific Antigen in the Diagnosis and Treatement of Adenocarcinoma of the Prostate: II. Radical Prostatectomy Treated Patients." *Journal of Urology*, **141**(5), 1076–1083.

Stigler SM (1984). "Studies in the History of Probability and Statistics XL Boscovich, Simpson and a 1760 Manuscript Note on Fitting a Linear Relation." *Biometrika*, **71**(3), 615–620. doi:10.1093/biomet/71.3.615.

Taddy MA, Kottas A (2010). "A Bayesian Nonparametric Approach to Inference for Quantile Regression." *Journal of Business and Economic Statistics*, **28**(3), 357–369. doi:10.1198/jbes.2009.07331.

Tanner MA, Wong WH (1987). "The Calculation of the Posterior Distribution by Data Augmentation." *Journal of the American Statistical Association*, **82**(398), 528–540. doi:10.1080/01621459.1987.10478458.

Tibshirani RJ (1996). "Regression Shrinkage and Selection via the Lasso." *Journal of the Royal Statistical Society B*, **58**(1), 267–288.

Tsionas E (2003). "Bayesian Quantile Inference." *Journal of Statistical Computation and Simulation*, **73**(9), 659–674. doi:10.1080/0094965031000064463.

Wang H, Li G, Jiang G (2007). "Robust Regression Shrinkage and Consistent Variable Selection through the LAD-Lasso." *Journal of Business & Economic Statistics*, **25**(3), 347–355. doi:10.1198/073500106000000251.

Wu Y, Liu Y (2009). "Variable Selection in Quantile Regression." *Statistica Sinica*, **19**(2), 801–817. doi:10.5705/ss.2011.100.

Yang Y, He X (2012). "Bayesian Empirical Likelihood for Quantile Regression." *The Annals of Statistics*, **40**(2), 1102–1131. doi:10.1214/12-aos1005.

Yang Y, Wang HJ, He X (2015). "Posterior Inference in Bayesian Quantile Regression with Asymmetric Laplace Likelihood." *International Statistical Review*, **84**(3), 327–344. doi:10.1111/insr.12114.

Yu K, Moyeed R (2001). "Bayesian Quantile Regression." *Statistics & Probability Letters*, **54**(4), 437–447. doi:10.1016/s0167-7152(01)00124-9.

Yu K, Stander J (2007). "Bayesian Analysis of a Tobit Quantile Regression Model." *Journal of Econometrics*, **137**(1), 260–276. doi:10.1016/j.jeconom.2005.10.002.

Yu K, Zhang J (2005). "A Three-Parameter Asymmetric Laplace Distribution and Its Extension." *Communiations in Statistics – Theory and Methods*, **34**(9–10), 1867–1879. doi:10.1080/03610920500199018.

Zou H (2006). "The Adaptive Lasso and Its Oracle Properties." *Journal of the American Statistical Association*, **101**(476), 1418–1429. doi:10.1198/016214506000000735.

Affiliation:

Dries F. Benoit
Faculty of Economics and Business Administration
Ghent University
9000 Ghent, Belgium
E-mail: Dries.Benoit@UGent.be

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
January 2017, Volume 76, Issue 7 *Submitted:* 2013-01-04
doi:10.18637/jss.v076.i07 *Accepted:* 2015-11-06