Fitting a Quantile Regression Model for Residual Life with the R Package qris

Kyu Hyun Kim, Sangwook Kang, and Sy Han Chiou

Abstract In survival analysis, regression modeling has traditionally focused on assessing covariate effects on survival times, which is defined as the elapsed time between a baseline and event time. Nevertheless, focusing on residual life can provide a more dynamic assessment of covariate effects, as it offers more updated information at specific time points between the baseline and event occurrence. Statistical methods for fitting quantile regression models have recently been proposed, providing favorable alternatives to modeling the mean of residual lifetimes. Despite these progresses, the lack of computer software that implements these methods remains an obstacle for researchers analyzing data in practice. In this paper, we introduce an R package **qris** (Kim et al., 2022), which implements methods for fitting semiparametric quantile regression models on residual life subject to right censoring. We demonstrate the effectiveness and versatility of this package through comprehensive simulation studies and a real-world data example, showcasing its valuable contributions to survival analysis research.

1 Introduction

In the analysis of time-to-event data, standard statistical inference procedures often focus on quantities based on failure time and its relationship with covariates measured at baseline. However, throughout the follow-up process, inference procedures based on residual life become increasingly intuitive for assessing the survival of subjects and can offer insights into the effectiveness of treatments in prolonging the remaining lifetime. As covariates can substantially change over time and models based solely on baseline covariates have limited potential for long-term prognosis, there is a growing interest in modeling the remaining lifetime of a surviving subject with updated patient information. Many efforts have been made to model the mean residual life including proportional mean residual life models (Maguluri and Zhang, 1994; Oakes and Dasu, 1990, 2003; Chen et al., 2005), additive mean residual life models (Chen and Cheng, 2006; Chen, 2007; Zhang et al., 2010), and proportional scaled mean residual life models (Liu and Ghosh, 2008). Given that failure times are usually right-skewed and heavy-tailed, the mean of the residual life might not be identifiable if the follow-up time is not sufficiently long. For this reason, quantiles, which are robust under skewed distribution, have traditionally been used more frequently as alternative summary measures. For example, the approach on the semiparametric quantile regression model for continuous responses (Koenker and Bassett Jr, 1978) has been extended to uncensored failure time data (Jung, 1996; Portnoy and Koenker, 1997; Wei et al., 2006) and censored failure times data (Ying et al., 1995; Portnoy, 2003; Peng and Huang, 2008; Huang, 2010).

When the outcome variable is the residual life, semiparametric quantile models that apply the inverse probability of censoring weighting (IPCW) principle to address right-censored observations have been explored (Jung et al., 2009; Kim et al., 2012; Li et al., 2016). These approaches are based on non-smooth estimating functions with respect to regression parameters, and the estimates of the regression parameters are obtained either through zero-crossing of non-smooth estimating functions using grid search techniques (Jung et al., 2009) or by optimizing non-smooth objective functions with L_1-minimization algorithms (Kim et al., 2012; Li et al., 2016). While these methods are relatively straightforward to implement, an additional challenge lies in standard error estimation, which necessitates the computationally intensive use of a multiplier bootstrap method (Li et al., 2016). Alternatively, Jung et al. (2009) and Kim et al. (2012) utilized the minimum dispersion statistic and the empirical likelihood method, respectively, to bypass the need to directly estimate the variance of the regression parameter estimator for hypothesis testing and constructing confidence intervals. The non-smooth nature of the estimating functions in these approaches precludes the estimation of variance using the robust sandwich-type variance estimator typically employed in equation-based estimation methods. To lessen the associated computational burden, an induced smoothing was proposed (Brown and Wang, 2005), which modifies the non-smooth estimating equations into smooth ones. Leveraging the asymptotic normality of the non-smooth estimator, the smooth estimating functions are constructed by averaging out the random perturbations inherent in the non-smooth estimating functions. The resulting estimating functions become smooth with respect to the regression parameters, allowing for the straightforward application of standard numerical algorithms, such as the Newton-Raphson method. Furthermore, these smoothed estimating functions facilitate the straightforward computation of variances using the robust sandwich-type estimator. The induced smoothing approach has been employed in fitting semiparametric accelerated failure

time (AFT) models via the rank-based approach (Johnson and Strawderman, 2009; Chiou et al., 2021, 2015b; Kang, 2017). Regarding quantile regression, Choi et al. (2018) considered the induced smoothing approach under a competing-risks setting. All of these methods are based on modeling event times. Recently, Kim et al. (2023) proposed an induced smoothing estimator for fitting a semiparametric quantile regression model for residual life.

The availability of published R packages for fitting quantile regression models is somewhat limited. The `rq()`, `nlrq()`, `rqss()`, and `crq()` functions in the package **quantreg** (Koenker, 2022) are predominantly used and provide various features for fitting linear, nonlinear, non-parametric, and censored quantile regression models, respectively. The `rq()` function minimizes non-smooth objective functions to obtain point estimates of regression coefficients and can accommodate right-censored survival times by incorporating weights. By redefining survival times as the remaining lifetime at time t_0, one can also obtain a non-smoothed estimator for quantile regression models for residual life (Kim et al., 2012). On the other hand, the `nlrq()` function is designed to fit a nonlinear quantile regression model, while the `rqss()` function fits additive quantile regression models with nonparametric terms, including univariate components and bivariate components, using smoothing splines and total variation regularization techniques (Koenker et al., 1994; Koenker and Mizera, 2004). Furthermore, the `crq()` function fits a quantile regression model for censored data on the τ-th conditional quantile function of the response variable. Overall, the **quantreg** implements three methods for handling right-censored survival times: Powell (1986)'s estimator, Portnoy (2003)'s estimator and Peng and Huang (2008)'s estimator. However, none of the implemented methods in the `nlrq()`, `rqss()`, or `crq()` functions are applicable for handling censored residual life using the induced smoothing methods. The only function that implements the induced smoothing method is the `aftsrr()` function in the package **aftgee** (Chiou et al., 2021), but it is specifically designed for fitting semiparametric AFT models, which are not directly applicable to fitting quantile regression models.

Other R packages that can be used to fit quantile regression models for survival data include the package **ctqr** (Frumento, 2021), package **Brq** (Alhamzawi, 2020), package **brms** (Bürkner, 2018), and package **cmprskQR** (Dlugosz et al., 2019). The `ctqr()` function in the package **ctqr** implements the methods proposed in Frumento (2021) for right or interval-censored failure times with left-truncation. The `Bqr()` function in the package **Brq** implements Bayesian methods based on the asymmetric Laplace distribution. In the package **brms**, the `brm()` function with the `family=asym_laplace()` option enables the implementation of full Bayesian inference. The `crrQR()` function in the package **cmprskQR** allows fitting quantile regression models with competing risks. All of these R packages are designed for fitting quantile regression models for failure times defined from a baseline and are not applicable to the residual life setting.

The recently developed R package **qris** (Kim et al., 2022) provides an efficient tool for fitting semiparametric quantile regression models for residual life subject to right censoring. The **qris** package offers three methods for estimating the regression parameters: L_1-minimization of non-smooth objective functions, induced smoothing with a non-iterative approach, and an iterative procedure. For standard error estimation, the **qris** package provides two resampling-based approaches: the partial multiplier bootstrap and the full multiplier bootstrap methods. The partial multiplier bootstrap method utilizes the robust sandwich-type estimator by incorporating the sample variance of perturbed estimating functions, while the full multiplier bootstrap method is obtained by considering the sample variance from the solutions of perturbed estimating functions. To enhance the interpretability of results, the **qris** package incorporates graphical visualizations of covariate effects at different quantiles and base times, utilizing the plotting environment similar to that in the **ggplot2** package (Wickham et al., 2022), thereby allowing for extensive flexibility and customization. The ultimate goal of creating the **qris** package is to facilitate the easy incorporation of quantile regression for residual life into daily routines. The package **qris** is available on the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=qris`.

The rest of the article is organized as follows: Section Semiparametric quantile regression for residual life introduces a semiparametric regression model for quantiles of residual life and the estimation methods implemented in the package. Section Package implementation provides details about computing algorithms. Illustrations of the package using a simulated dataset and the real data from the North Central Cancer Treatment Group are presented in Section Illustration. Finally, in Section Conclusion, concluding remarks are provided along with some discussions.

2 Semiparametric quantile regression for residual life

Define T as the potential failure time that is subject to right censoring by C and $\mathbf{X}$ as a $p \times 1$ vector of covariates, where p is the number of covariates, including an intercept. The observed data consists of n independent copies of $(Z, \delta, \mathbf{X})$, where $Z = \min(T, C)$, $\delta = I(T \leq C)$, and $I(\cdot)$ is an indicator

function. We also assume T and C are marginally independent. Define the τ-th quantile of the residual life at $t_0 > 0$ as $\theta_\tau(t_0)$ that satisfies $P(T_i - t_0 \geq \theta_\tau(t_0) \mid T_i > t_0) = 1 - \tau$. We consider the semiparametric quantile regression model for the residual life (Kim et al., 2012, 2023). Given $T_i > t_0$,

$$\log(T_i - t_0) = \mathbf{X}_i^\top \boldsymbol{\beta}_0(\tau, t_0) + \epsilon_i, i = 1, \ldots, n, \tag{1}$$

where $\boldsymbol{\beta}_0(\tau, t_0)$ is a $p \times 1$ vector of regression coefficients, and ϵ_i is a random error having zero τ-th quantile. The quantile regression model for a continuous response (Koenker and Bassett Jr, 1978) is a special case of Equation (1) when $t_0 = 0$. For ease of notation, we omit τ and t_0 in $\boldsymbol{\beta}_0(\tau, t_0)$ and $\theta_\tau(t_0)$ and write $\boldsymbol{\beta}_0$ and θ. We present different estimation procedures to estimate $\boldsymbol{\beta}_0$ given τ and t_0 in the following.

2.1 Estimation using non-smooth functions

When there is no censoring, an estimator for β_0 in Equation (1) can be obtained by solving the estimating equation (Kim et al., 2012), where

$$\frac{1}{n}\sum_{i=0}^{n} I[T_i \geq t_0]\mathbf{X}_i \left\{ I\left[\log(T_i - t_0) \leq \mathbf{X}_i^\top \boldsymbol{\beta}\right] - \tau \right\} = 0. \tag{2}$$

However, Equation (2) cannot be directly used when $T_i - t_0$ is subject to right censoring. The IPCW technique can be incorporated into Equation (2) to account for the right censoring (Li et al., 2016). Specifically, in the presence of right censoring, the estimator for $\boldsymbol{\beta}_0$ in Equation (1) can be obtained as the root of the following weighted estimating equations:

$$U_{t_0}(\boldsymbol{\beta}, \tau) = \frac{1}{n}\sum_{i=1}^{n} I[Z_i \geq t_0]\mathbf{X}_i \left\{ I\left[\log(Z_i - t_0) \leq \mathbf{X}_i^\top \boldsymbol{\beta}\right] \frac{\delta_i}{\widehat{G}(Z_i)/\widehat{G}(t_0)} - \tau \right\}, \tag{3}$$

where $\widehat{G}(\cdot)$ is the Kaplan-Meier estimate of the survival function $G(\cdot)$ of the censoring time C and $\widehat{G}(t) = \prod_{i:t_i \leq t}(1 - \sum_{j=1}^{n}(1-\delta_j)I(Z_j \leq t_i)/\sum_{j=1}^{n} I(Z_j \geq t_i))$. A computational challenge arises because the exact solution to Equation (3) might not exist due to the non-smoothness in β caused by the involvement of indicator functions. When the exact solutions do not exist, the root of Equation (3) can be approximated by minimizing the L_1-objective function $L_{t_0}(\boldsymbol{\beta}, \tau)$ (Li et al., 2016),

$$\begin{aligned} L_{t_0}(\boldsymbol{\beta}, \tau) =& \frac{1}{n}\sum_{i=1}^{n} \frac{\delta_i I[Z_i > t_0]}{\widehat{G}(Z_i)/\widehat{G}(t_0)} \left|\log(Z_i - t_0) - \mathbf{X}_i^\top \beta\right| + \\ & \left| M - \boldsymbol{\beta}^\top \sum_{l=1}^{n} -\mathbf{X}_l \frac{\delta_l I[Z_l > t_0]}{\widehat{G}(Z_l)/\widehat{G}(t_0)} \right| + \left| M - \boldsymbol{\beta}^\top \sum_{l=1}^{n} 2\tau \mathbf{X}_l I[Z_l > t_0] \right|, \end{aligned}$$

where $M > 0$ bounds $\left|\boldsymbol{\beta}^\top \sum_{i=1}^{n} -\mathbf{X}_i \frac{\delta_i I[Z_i > t_0]}{\widehat{G}(Z_i)/\widehat{G}(t_0)}\right|$ and $\left|\boldsymbol{\beta}^\top \sum_{i=1}^{n} 2\tau \mathbf{X}_i I[Z_i > t_0]\right|$ from above. Numerically, the limit M is set to be an extremely large number, and the `qris()` function uses $M = 10^6$. Denote the resulting estimator to be $\widehat{\boldsymbol{\beta}}_{\text{NS}}$. It has been shown that $\widehat{\boldsymbol{\beta}}_{\text{NS}}$ is consistent for $\boldsymbol{\beta}_0$ and asymptotically normally distributed (Li et al., 2016).

Despite the well-established asymptotic properties, directly estimating the variance of $\widehat{\boldsymbol{\beta}}_{\text{NS}}$ is impractical because it involves the derivative of non-smooth functions. A multiplier bootstrap method has typically been employed (Li et al., 2016) to address this difficulty. The multiplier bootstrap method considers the perturbed version of $U_{t_0}(\beta, \tau)$, defined as

$$U^*_{t_0}(\beta, \tau) = \frac{1}{n}\sum_{i=1}^{n} \eta_i I[Z_i \geq t_0]\mathbf{X}_i \left\{ I\left[\log(Z_i - t_0) \leq \mathbf{X}_i^\top \boldsymbol{\beta}\right] \frac{\delta_i}{\widehat{G}^*(Z_i)/\widehat{G}^*(t_0)} - \tau \right\},$$

where $\eta_i, i = 1, \ldots, n$, are independently and identically (iid) generated from a positive random variable with unity mean and variance, and $\widehat{G}^*(\cdot)$ is a perturbed version of $\widehat{G}(\cdot)$, constructed as $\widehat{G}^*(t) = \prod_{i:t_i \leq t}(1 - \sum_{j=1}^{n}\eta_j(1-\delta_j)I(Z_j \leq t_i)/\sum_{j=1}^{n}\eta_j I(Z_j \geq t_i))$ for a given realization of η_i. On the other hand, a perturbed L_1-objective function, denoted as $L^*_{t_0}(\boldsymbol{\beta}, \tau)$, can be similarly

constructed, where

$$L^*_{t_0}(\boldsymbol{\beta},\tau) = \frac{1}{n}\sum_{i=1}^{n} \frac{\delta_i I[Z_i > t_0]}{\widehat{G}^*(Z_i)/\widehat{G}^*(t_0)} \left|\log(Z_i - t_0) - \mathbf{X}_i^\top \boldsymbol{\beta}\right| + \left|M - \boldsymbol{\beta}^\top \sum_{l=1}^{n} -\mathbf{X}_l \frac{\delta_l I[Z_l > t_0]}{\widehat{G}^*(Z_l)/\widehat{G}^*(t_0)}\right| + \left|M - \beta^\top \sum_{l=1}^{n} 2\tau \mathbf{X}_l \eta_l I[Z_l > t_0]\right|.$$

Solving for $U^*_{t_0}(\boldsymbol{\beta},\tau) = 0$, or equivalently, minimizing $L^*_{t_0}(\boldsymbol{\beta},\tau)$, yields one realization of $\widehat{\boldsymbol{\beta}}_{\text{NS}}$. The multiplier bootstrap variance is computed as the sample variance of a large number of realizations of $\widehat{\boldsymbol{\beta}}_{\text{NS}}$.

2.2 Estimation using induced smoothed functions

The regression coefficient in Equation (1) can be more efficiently obtained through the induced smoothed version of Equation (3). The induced smoothed estimating functions are constructed by taking the expectation with respect to a mean-zero random noise added to the regression parameters in Equation (3). Specifically,

$$\begin{aligned}\widetilde{U}_{t_0}(\boldsymbol{\beta},\tau,H) &= E_w\{U_{t_0}(\boldsymbol{\beta} + \mathbf{H}^{1/2}\mathbf{W},\tau)\} \\ &= \frac{1}{n}\sum_{i=1}^{n} I[Z_i > t_0]\mathbf{X}_i \left\{\Phi\left(\frac{\mathbf{X}_i^\top\boldsymbol{\beta} - \log(Z_i - t_0)}{\sqrt{\mathbf{X}_i^\top \mathbf{H}\mathbf{X}_i}}\right) \frac{\delta_i}{\widehat{G}(Z_i)/\widehat{G}(t_0)} - \tau\right\}, \end{aligned} \tag{4}$$

where $\mathbf{H} = O(n^{-1})$, $\mathbf{W} \sim N(0, \mathbf{I}_p)$ is a standard normal random vector, $\mathbf{I}_p$ is the $p \times p$ identity matrix, and $\Phi(\cdot)$ is the cumulative distribution function of a standard normal random variable. A typical choice for $\mathbf{H}$ is to fix it at $n^{-1}\mathbf{I}_p$, while some alternative choices are explored in Chiou et al. (2015a). Let $\widehat{\boldsymbol{\beta}}_{\text{IS}}$ be the solution to $\widetilde{U}_{t_0}(\boldsymbol{\beta},\tau,\mathbf{H}) = 0$. Since Equation (4) is a smooth function in $\boldsymbol{\beta}$, the estimator can be obtained using standard numerical algorithms such as the Newton-Raphson method. Moreover, the induced smoothed estimator for $\boldsymbol{\beta}_0$ has been shown to be asymptotically equivalent to its non-smooth counterpart (Kim et al., 2023).

Following the idea in Section Estimation using non-smooth functions, the multiplier bootstrap procedure can be similarly employed to estimate the variance of $\widehat{\boldsymbol{\beta}}_{\text{IS}}$. The perturbed version of Equation (4) takes the form of

$$\widetilde{U}^*_{t_0}(\boldsymbol{\beta},\tau,\mathbf{H}) = \frac{1}{n}\sum_{i=1}^{n} \eta_i I[Z_i > t_0]\mathbf{X}_i \left\{\Phi\left(\frac{\mathbf{X}_i^\top\boldsymbol{\beta} - \log(Z_i - t_0)}{\sqrt{\mathbf{X}_i^\top \mathbf{H}\mathbf{X}_i}}\right) \frac{\widehat{G}^*(t_0)\delta_i}{\widehat{G}^*(Z_i)} - \tau\right\}. \tag{5}$$

The multiplier bootstrap procedure estimates the variance of $\widehat{\boldsymbol{\beta}}_{\text{IS}}$ by calculating the sample variance of a large number of realizations of $\widehat{\boldsymbol{\beta}}_{\text{IS}}$ obtained by repeatedly solving Equation (5).

It has been shown that the asymptotic variance $\text{Var}(\boldsymbol{\beta},\tau)$ can be decomposed into $\mathbf{A}(\boldsymbol{\beta})^\top\mathbf{V}(\boldsymbol{\beta})\mathbf{A}(\boldsymbol{\beta})$ (Kim et al., 2023), where the two components, $\mathbf{A}(\boldsymbol{\beta})$ and $\mathbf{V}(\boldsymbol{\beta})$, can be estimated separately. Since Equation (4) is a smooth function in $\boldsymbol{\beta}$, the slope matrix, $\mathbf{A}(\boldsymbol{\beta})$, can be conveniently estimated by differentiating $\widetilde{U}_{t_0}(\boldsymbol{\beta},\tau,\mathbf{H})$ with respect to $\boldsymbol{\beta}$. The explicit form of $\mathbf{A}(\boldsymbol{\beta})$ is as follows:

$$\begin{aligned}\mathbf{A}(\boldsymbol{\beta}) &= \frac{\partial \widetilde{U}_{t_0}(\boldsymbol{\beta},\tau,\mathbf{H})}{\partial\boldsymbol{\beta}} \\ &= \frac{1}{n}\sum_{i=1}^{n} I[Z_i > t_0]\mathbf{X}_i \frac{G(t_0)\delta_i}{G(Z_i)} \phi\left(\frac{\mathbf{X}_i{}^\top\boldsymbol{\beta} - \log(Z_i - t_0)}{\sqrt{\mathbf{X}_i{}^\top \mathbf{H}\mathbf{X}_i}}\right)\left(\frac{-\mathbf{X}_i}{\sqrt{\mathbf{X}_i{}^\top \mathbf{H}\mathbf{X}_i}}\right), \end{aligned} \tag{6}$$

where $\phi(\cdot)$ is the density function of the standard normal random variable.

The slope matrix, $\widehat{\mathbf{A}}(\widehat{\boldsymbol{\beta}}_{\text{IS}})$, can be evaluated directly by plugging in $\widehat{\boldsymbol{\beta}}_{\text{IS}}$ and $\widehat{G}(\cdot)$. On the other hand, the variance of the estimating function, $\widehat{\mathbf{V}}(\boldsymbol{\beta})$, can be obtained by a computationally efficient resampling method motivated by the multiplier bootstrap procedure in Section Estimation using non-smooth functions. Specifically, we propose estimating $\widehat{\mathbf{V}}(\widehat{\boldsymbol{\beta}}_{\text{IS}})$ as the simple variance of a large set of realizations of the perturbed version of $\widetilde{U}_{t_0}(\widehat{\boldsymbol{\beta}}_{\text{IS}},\tau,\mathbf{H})$ presented in Equation (5). We refer to this procedure as the partial multiplier bootstrapping approach because it utilizes the perturbed estimating function, similar to the full multiplier bootstrapping approach, but the computation of $\widehat{\mathbf{A}}(\widehat{\boldsymbol{\beta}}_{\text{IS}})$ and $\widehat{\mathbf{V}}(\widehat{\boldsymbol{\beta}}_{\text{IS}})$ does not involve the repeated solving of the perturbed estimating equations. Thus, the partial multiplier bootstrapping approach is expected to be computationally more efficient

than the multiplier bootstrap method. A similar procedure and its performance have been studied in modeling failure times with semiparametric AFT models (Chiou et al., 2014, 2021).

2.3 Iterative procedure in induced smoothing estimation

The induced estimator $\widehat{\boldsymbol{\beta}}_{\text{IS}}$ is obtained with a fixed $\mathbf{H}$, as described in Section Estimation using induced smoothed functions, and its variance is estimated separately. This estimation procedure can be viewed as a special case of the following iterative procedure, which updates $\mathbf{H}$ and $\widehat{\boldsymbol{\beta}}_{\text{IS}}$ iteratively. Specifically, the iterative algorithm utilizes the Newton-Raphson method while sequentially updating $\widehat{\boldsymbol{\beta}}_{\text{IS}}$ and $\widehat{\text{Var}}(\widehat{\boldsymbol{\beta}}_{\text{IS}})$ until convergence. Similar iterative algorithms have also been considered previously in the induced smoothing approach for semiparametric AFT models (Johnson and Strawderman, 2009; Chiou et al., 2014, 2015b; Choi et al., 2018). The iterative procedure is summarized as follows:

Step 1: Set the initial values $\widehat{\boldsymbol{\beta}}^{(0)}$, $\widehat{\boldsymbol{\Sigma}}^{(0)} = \mathbf{I}_p$, and $\mathbf{H}^{(0)} = n^{-1}\widehat{\boldsymbol{\Sigma}}^{(0)}$.

Step 2: Given $\widehat{\boldsymbol{\beta}}^{(k)}$ and $\mathbf{H}^{(k)}$ at the k-th step, update $\widehat{\boldsymbol{\beta}}^{(k)}$ by

$$\widehat{\boldsymbol{\beta}}^{(k+1)} = \widehat{\boldsymbol{\beta}}^{(k)} - \widehat{\mathbf{A}}(\widehat{\boldsymbol{\beta}}^{(k)})^{-1}\widetilde{U}_{t_0}(\widehat{\boldsymbol{\beta}}^{(k)}, \tau, \mathbf{H}^{(k)}).$$

Step 3: Given $\widehat{\boldsymbol{\beta}}^{(k+1)}$ and $\widehat{\boldsymbol{\Sigma}}^{(k)}$, update $\widehat{\boldsymbol{\Sigma}}^{(k)}$ by

$$\widehat{\boldsymbol{\Sigma}}^{(k+1)} = \widehat{\mathbf{A}}(\widehat{\boldsymbol{\beta}}^{(k+1)})^{-1}\widehat{\mathbf{V}}(\widehat{\boldsymbol{\beta}}^{(k+1)}, \tau)\widehat{\mathbf{A}}(\widehat{\boldsymbol{\beta}}^{(k+1)})^{-1}.$$

Step 4: Set $\mathbf{H}^{(k+1)} = n^{-1}\widehat{\boldsymbol{\Sigma}}^{(k+1)}$. Repeat Steps 2, 3 and 4 until $\widehat{\boldsymbol{\beta}}^{(k)}$ and $\widehat{\boldsymbol{\Sigma}}^{(k)}$ converge.

The initial value, $\widehat{\boldsymbol{\beta}}^{(0)}$, could be chosen as $\widehat{\boldsymbol{\beta}}_{\text{NS}}$. We define $\widehat{\boldsymbol{\beta}}_{\text{IT}}$ and $\widehat{\boldsymbol{\Sigma}}_{\text{IT}}$ as the values of $\widehat{\boldsymbol{\beta}}^{(k)}$ and $\widehat{\boldsymbol{\Sigma}}^{(k)}$ at convergence, and $\widehat{\text{Var}}(\widehat{\boldsymbol{\beta}}_{\text{IT}}) = n^{-1}\widehat{\boldsymbol{\Sigma}}_{\text{IT}}$. In Step 3, $\widehat{\mathbf{V}}(\widehat{\boldsymbol{\beta}}^{(k+1)}, \tau)$ is obtained using the partial multiplier bootstrap approach. However, the full multiplier bootstrap approach can also be employed but would require longer computation times.

3 Package implementation

The main function in the **qris** package for estimating the regression parameters in the quantile regression model for residual life is the `qris()` function. The `qris()` function is written in C++ and incorporated into R using the **Rcpp** (Eddelbuettel et al., 2022a) and **RcppArmadillo** (Eddelbuettel et al., 2022b) packages. The synopsis of `qris` is:

```
> args(qris)
function (formula, data, t0 = 0, Q = 0.5, nB = 100, method = c("smooth",
"iterative", "nonsmooth"), se = c("fmb",
"pmb"), init = c("rq", "noeffect"), verbose = FALSE,
control = qris.control())
```

The required argument is `formula`, which specifies the quantile regression model to be fitted using the variables in `data`. The `formula` assumes that the response variable is a '`Surv`' object created by the `Surv()` function in the **survival** package (Therneau, 2021). This formula structure is commonly adopted for handling survival data in R, as seen in functions like `survreg()` and `coxph()` in the **survival** package. The argument `t0` specifies the base time used in defining residual life. The default value of `t0` is set to zero, in which case residual life reduces to a failure time. The `Q` argument is used to specify the target quantile of residual life to estimate, with the default value being set to 0.5 (median). The `nB` argument specifies the bootstrapping size used in standard error estimation, with the default value set to 100. The `method` argument specifies one of the three estimation methods: `"nonsmooth"`, `"smooth"`, and `"iterative"`, corresponding to the estimating procedures outlined in Sections Estimation using non-smooth functions, Estimation using induced smoothed functions, and Iterative procedure in induced smoothing estimation, respectively. Given the point estimates of the regression parameters, their standard errors can be estimated using one of two implemented methods: `se = "fmb"` and `se = "pmb"`. The `se = "fmb"` method employs a full-multiplier bootstrapping approach to estimate the variance by the sample variance of large realizations of $\widehat{\beta}$. The `se = "pmb"` method estimates the variance using a robust sandwich variance estimator and employs the computationally efficient partial multiplier bootstrapping approach described in Section Estimation using induced smoothed functions. The `"fmb"` option is available for all three point estimation methods, whereas the `"pmb"` option is not available for the `"nonsmooth"` point estimation method due to the lack of a closed-form sandwich variance estimator. The `init`

argument allows users to specify the initial value for estimating regression parameters by either a p-dimensional numerical vector or a character string. In the latter case, the options `init = "rq"` and `init = "noeffect"` correspond to the point estimate obtained from the `rq()` function in the **quantreg** package and a p-dimensional vector of zeros, respectively. The default value for `init` is `init = "rq"`. Among the three methods implemented for point estimation, `method = "smooth"` and `method = "nonsmooth"` are non-iterative, in the sense that point estimation is performed separately from the estimation of standard errors. On the other hand, `method = "iterative"` calculates point estimates and the corresponding standard error estimates simultaneously through iterative updates. When `method = "iterative"`, users can define specific convergence criteria using `qris.control()`. The available options in `qris.control()` are as follows.

```
> args(qris.control)
function (maxiter = 10, tol = 0.001, trace = FALSE)
```

The `maxiter` argument specifies the maximum number of iterations. The default value for `maxiter` is ten, as the proposed algorithm typically converges within ten steps based on our exploration. The convergence tolerance is controlled using the `tol` argument, which has a default value of `1e-3`. The `trace` argument takes a logical value and is used to determine whether to print the result for each iteration. The default setting is `trace = FALSE`. The '`qris`' object is fully compatible with many of R's generic functions, including `coef()`, `confint()`, `plot()`, `predict()`, `print()`, `residuals()`, `summary()`, and `vcov()`.

Among the available `S3` methods, a unique feature of the **qris** package's `S3 plot` method, when applied to a '`qris`' object, is its ability to automatically update the original object by extending the range of τ or t_0 values. This extension enables the generation of a covariate effect plot over the newly specified values of τ or t_0, providing a comprehensive visualization of the covariate effects across the extended range. The `S3` method for plotting a '`qris`' object is shown below.

```
> argsAnywhere(plot.qris)
function (x, t0s = NULL, Qs = NULL, nB = NULL, vari = NULL, byQs = FALSE,
    ggextra = NULL, ...)
NULL
```

The argument `x` is a '`qris`' object created using the `qris()` function. The `t0s` and `Qs` arguments are numeric vectors that enable users to specify the values of t_0 or τ for plotting the covariate effect. If `t0s` and `Qs` are not specified, the covariate effects are plotted against $\tau = 0.1, 0.2, \ldots, 0.9$ at the base time (t_0) inherited from the '`qris`' object specified in `x`. The `nB` argument is a numerical variable that controls the sample size for bootstrapping, used to compute standard error estimations based on the variance estimation specified in the original '`qris`' object. When `nB` is specified, the function calculates standard errors for all combinations of t_0 and τ specified in `t0s` and `Qs`, computes 95% confidence intervals accordingly, and includes them in the covariate effect plot. The `vari` argument is a character string that allows users to specify the names of the covariates they want to display in the effect plots. When the `vari` argument is not specified, all covariates will be included in the plots by default. The coefficient event plot can be plotted against the specified quantiles by setting `byQs = TRUE` or against the specified base times by setting `byQs = FALSE`. Finally, the `ggextra` argument allows users to pass additional graphical parameters to the **ggplot2** package, offering further customization options for the plots. When the `plot()` function is called, it internally invokes the `qris.extend()` function to compute the covariate effects at additional values. The syntax for the `qris.extend()` function is provided below:

```
> args(qris.extend)
function (x, t0s = NULL, Qs = NULL, nB = NULL, vari = NULL)
NULL
```

The arguments in `qris.extend()` are inherited from the arguments specified in the `plot()` function. To reduce runtime when repeatedly calling the `plot()`, one can calculate the desired covariate effects by applying `qris.extend()` outside of `plot()` first and then supply the results to `plot()`. This approach allows for pre-computation of the covariate effects, making it more efficient when generating multiple plots. Overall, the unique plotting feature in **qris** provides users with a seamless and effortless approach to conducting a comprehensive assessment of the covariate effects across different quantiles or base times.

4 Illustration

4.1 Simulated data

In this subsection, we present a simple simulation example to validate the implementations in the proposed qris package. The simulation involves five covariates, denoted as $X_1, \ldots, X_5$. Among these covariates, X_1 and X_4 follow a standard uniform distribution, X_2 follows a binomial distribution with a success probability of 0.5, X_3 follows a standard normal distribution, and X_5 follows a standard exponential distribution. We assume that X_2, X_3, X_4, and X_5 do not impact the residual life, meaning their corresponding coefficient values β_2, β_3, β_4, and β_5 are zero. The survival time T is generated from a Weibull distribution with the survival function $S(t) = \exp\{-(\rho t)^\kappa\}$ for $t > 0$, where $\kappa = 2$, and ρ is obtained by solving

$$\rho^{-1}\{(\rho t_0)^\kappa - \log(1-\tau)\}^{(1/\kappa)} - t_0 = \exp\{\beta_0 + \beta_1 X_1\}, \tag{7}$$

for a specified t_0 and τ. We set the intercept $\beta_0 = \log(5)$ and $\beta_1 = \log(2)$ at $t_0 = 0$. Given ρ, τ, and X_1, the true values of β_0 and β_1 can be obtained sequentially from Equation 7 for different $t_0 > 0$. In our case, the corresponding true values of β_0 are approximately 1.411 and 1.219 for $t_0 = 1$ and 2, respectively. Similarly, the true values of β_1 are approximately 0.797 and 0.907 for $t_0 = 1$ and 2, respectively. The closed-form expression for generating T is then $\{-\log(1-u)\}^{1/\kappa}/\rho$, where u is a uniform random variable over $(0, 1)$. Given these specifications, we have implemented the `data.gen()` function to generate simulation data. The `data.gen()` function takes four arguments: `n`, `t0`, `cen`, and `Q`, representing the sample size, t_0, censoring proportion, and τ, respectively. We generate censoring times C from an independent uniform distribution over $(0, c)$, where c is chosen to achieve the desired censoring proportions of 10% and 30%. Using the generated dataset, we fit the model using three different estimation methods: induced smoothing, non-smooth, and iterative-induced smoothing. All analyses were conducted on a 4.2 GHz Intel(R) quad Core(TM) i7-7700K central processing unit (CPU) using R 4.3.0 (R Core Team, 2021). The following code demonstrates the implementation of `data.gen()` to generate a simulation dataset.

```
> data.gen <- function(n, t0, cen = .3, Q = .5) {
+   if (!(t0 %in%  0:2))
+     stop("T0 is limited to three specific values: 0, 1, or 2.")
+   if (!(cen %in%  c(0, .1, .3)))
+     stop("Censoring is limited to three specific values: 0%, 10%, or 30%.")
+   if (!(Q %in% c(.25, .5)))
+     stop("Q is limited to two specific values: 0.25, or 0.50.")
+   censoring <- Inf
+   if (t0 == 0) {
+     if (cen == .1) censoring <- runif(n, 0, 125.1)
+     if (cen == .3) censoring <- runif(n, 0, 25.49)
+     beta0 <- log(5); beta1 <- log(2)
+   }
+   if (t0 == 1) {
+     if (cen == .1) censoring <- runif(n, 0, 120.8)
+     if (cen == .3) censoring <- runif(n, 0, 23.41)
+     beta0 <- 1.410748; beta1 <- 0.7974189
+   }
+   if (t0 == 2) {
+     if (cen == .1) censoring <- runif(n, 0, 120.6)
+     if (cen == .3) censoring <- runif(n, 0, 26.20)
+     beta0 <- 1.219403; beta1 <- 0.9070615
+   }
+   dat <- data.frame(censoring,
+                     Time0 = sqrt(-log(1 - runif(n))),
+                     X1 = runif(n),
+                     X2 = rbinom(n, 1, .5),
+                     X3 = rnorm(n),
+                     X4 = runif(n),
+                     X5 = rexp(n, 1))
+   rho <- (-log(1 - Q))^0.5 * (((exp(beta0 + beta1 * dat$X1) + t0)^2 - t0^2)^-0.5)
+   dat$Time0 <- dat$Time0 / rho
+   dat$Time <- pmin(dat$Time0, dat$censoring)
+   dat$status <- 1 * (dat$Time0 < dat$censoring)
```

```
+   subset(dat, select = c(Time, status, X1, X2, X3, X4, X5))
+ }
> set.seed(3)
> head(data.gen(200, 0))

Time status         X1 X2         X3         X4        X5
1  4.283379      0 0.09137221  0  2.1638425 0.33833437 0.8751895
2 14.797025      1 0.81196535  1  0.8803785 0.82101134 0.3648634
3  5.934559      1 0.60923418  1  0.5051163 0.56536790 0.3997803
4  7.223266      1 0.54550179  1  0.1105902 0.32417202 1.2169470
5 15.128553      1 0.86115736  0 -0.2928586 0.05825095 0.1835962
6  5.135852      1 0.28915525  0  0.7723200 0.94126325 0.3809120
```

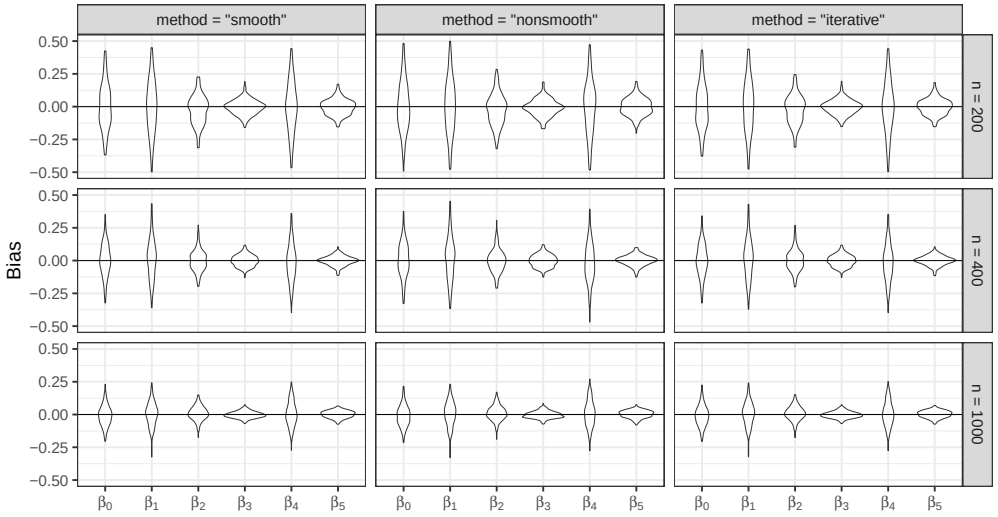
The `data.gen()` function generates a `data.frame` containing seven variables. The `Time` variable represents the observed survival time, while the `status` variable serves as the event indicator, taking the value 1 for observed events and 0 for censored observations. The variables `X1`, ..., `X5` are the covariates. The implementation in the `data.gen()` function generates the Weibull survival times using the inverse probability integral transform technique. Alternatively, users can use the `rweibull()` function with the parameters `shape = 2` and `scale = 1 / rho` to generate these Weibull survival times directly.

We assess the performance of the proposed implementation across various scenarios, including three sample sizes ($n = 200, 400, 1000$), three levels of t_0 $(0, 1, 2)$, two censoring proportions (10% and 30%), and two values of τ (0.25 and 0.50). For a given dataset, we apply the full-multiplier bootstrapping approach with 200 bootstrap samples to all three available estimating procedures: `method = "nonsmooth"`, `method = "smooth"`, and `method = "iterative"`. To facilitate the evaluation process, we create the `do_fmb()` function to record the coefficient estimates, standard errors, and computing times for fitting a single simulated dataset generated from `data.gen()`. The following is the implementation of the `do_fmb()` function and the corresponding code to run the simulation with 200 replications. We present the code and result of the simulation experiments conducted at three different sample sizes, with t_0 values set to 0 and 1, while holding the censoring proportion at 30% and τ value at 0.5. The results for other simulation scenarios are provided in the Supplementary Materials.

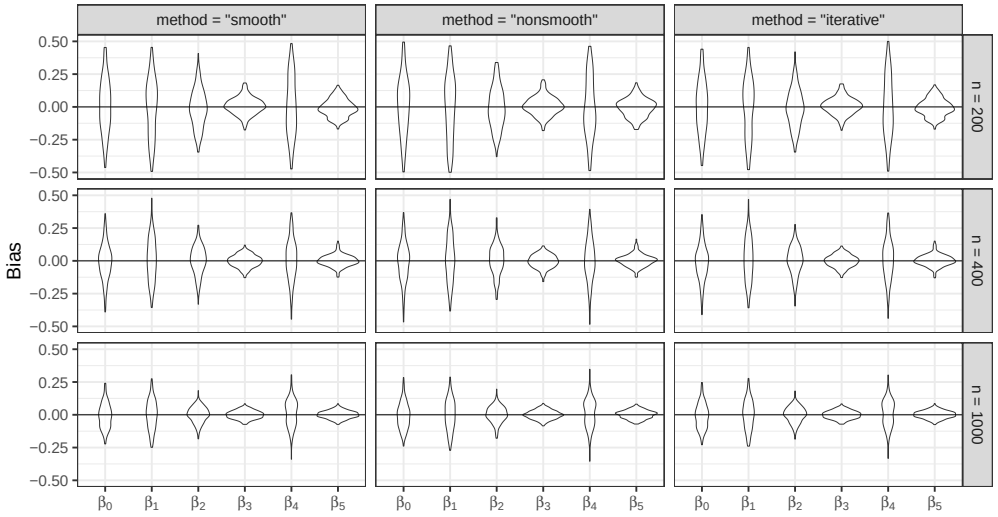
```
> do_fmb <- function(n, t0, cen, Q, nB) {
+   dat <- data.gen(n, t0, cen, Q)
+   fm <- Surv(Time, status) ~ X1 + X2 + X3 + X4 + X5
+   stamp <- NULL
+   stamp[1] <- Sys.time()
+   f1 <- qris(fm, data = dat, t0 = t0, Q = Q, nB = nB, method = "smooth", se = "fmb")
+   stamp[2] <- Sys.time()
+   f2 <- qris(fm, data = dat, t0 = t0, Q = Q, nB = nB, method = "nonsmooth", se = "fmb")
+   stamp[3] <- Sys.time()
+   f3 <- qris(fm, data = dat, t0 = t0, Q = Q, nB = nB, method = "iterative", se = "fmb")
+   stamp[4] <- Sys.time()
+   list(smooth = c(f1$coef, f1$std),
+     nonsmooth = c(f2$coef, f2$std),
+     iter = c(f3$coef, f3$std),
+     times = diff(stamp))
+ }
> B <- 200
> set.seed(2)
> sims0_fmb <- mapply(function(n, t0)
  +     replicate(B, do_fmb(n, t0 = t0, cen = .3, Q = .5, nB = 200)),
  +     n = c(200, 400, 1000), t0 = c(0, 0, 0), SIMPLIFY = F)
> sim1_fmb <- mapply(function(n, t0)
  +     replicate(B, do_fmb(n, t0 = t0, cen = .3, Q = .5, nB = 200)),
  +     n = c(200, 400, 1000), t0 = c(1, 1, 1), SIMPLIFY = F)
```

Figure 1 displays violin plots that provide visualizations of the empirical distribution of the coefficient estimates. As expected, all three estimators exhibit small biases, which are calculated as the difference between the point estimates (PE) and the true regression coefficients. Furthermore, the empirical distributions of the PEs demonstrate a normal-like shape, aligning with the asymptotic properties of the proposed method (Li et al., 2016; Kim et al., 2023). When the sample size is smaller (e.g., $n = 200$ and 400), the `nonsmooth` approach appears to yield slightly larger empirical standard errors (ESE) compared to the `smooth` or `iterative` approaches. However, when $n = 1000$, the ESEs

are similar across all approaches. On the other hand, the comprehensive simulation results presented in Table 1 of the Supplementary Materials confirm that all coefficient estimates closely approximate the true regression coefficients. On the other hand, the ESEs and the averaged estimated standard errors (ASE) are in close agreement for all scenarios, indicating the validity of the variance estimation. Furthermore, the computation times, which are presented separately in the upper panel of Table 1, indicate that when employing the full multiplier bootstrapping approach, the `nonsmooth` approach demonstrates a slight advantage in terms of computational efficiency over the `smooth` approach, while the `iterative` approach takes 5.1 to 9.5 times longer than the `smooth` approach. In summary, the timing results show that the proposed method can yield valid inference results within seconds, even with large datasets of up to 1000 observations or when using the computationally demanding full multiplier bootstrapping approach for variance estimation.



(a) $t_0 = 0$



(b) $t_0 = 1$

Figure 1: Comparison of the `smooth`, `nonsmooth` and `iterative` estimators with `se = "fmb"` under 30% censoring and $\tau = 0.5$.

When $t_0 = 0$, the targeted semiparametric quantile regression model for residual life simplifies to the standard quantile regression model for survival time. In such cases, existing functions like `crq()` from the **quantreg** package (Koenker, 2022) can be employed. A comparison between the performance of `crq()` and our proposed implementation when $t_0 = 0$ is presented in the Supplementary Materials, where the standard errors of the `crq()` are obtained from the bootstrap method with 200 bootstrap samples. Overall, the performance of `crq()` is comparable to the proposed methods in terms of bias

Table 1: Runtimes (in seconds) when `se = fmb` and `se = pmb`.

		$t_0 = 0$			$t_0 = 1$		
se	method	200	400	1000	200	400	1000
`fmb`	Smooth	0.103	0.174	0.471	0.106	0.178	0.480
	Nonsmooth	0.080	0.142	0.472	0.080	0.141	0.468
	Iterative	0.981	1.500	2.410	0.985	1.567	2.882
`pmb`	Smooth	0.022	0.052	0.223	0.022	0.053	0.224
	Iterative	0.296	0.580	1.407	0.296	0.581	1.435

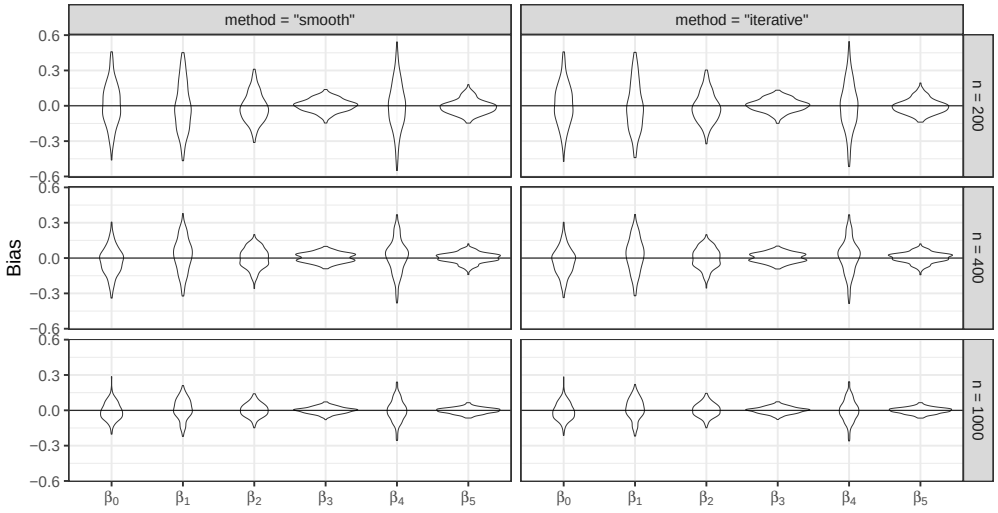
and standard errors. However, we have occasionally encountered situations where the `crq()` function fails to converge, particularly when the sample size is large, as in the case of $n = 1000$. In the other extended simulation scenarios outlined in the Supplementary Materials, which encompass various levels of t_0, censoring proportions, and τ, the proposed methods consistently exhibit satisfactory performance across all settings.

The true potential of the proposed smooth approach lies in its capability for efficient variance estimation through the implementation of the partial multiplier bootstrapping approach. This approach eliminates the need for repetitive solving of estimating equations, resulting in improved computational efficiency in variance estimation. To demonstrate its usefulness, we conducted a simulation using both the smooth approach and the iterative approach with the partial multiplier bootstrapping approach (`se = "pmb"`). This simulation was conducted under the settings of $\tau = 0.5$, $t_0 = 0$ and 1, and a 30% censoring rate. The `do_pmb()` function was accordingly modified as follows.

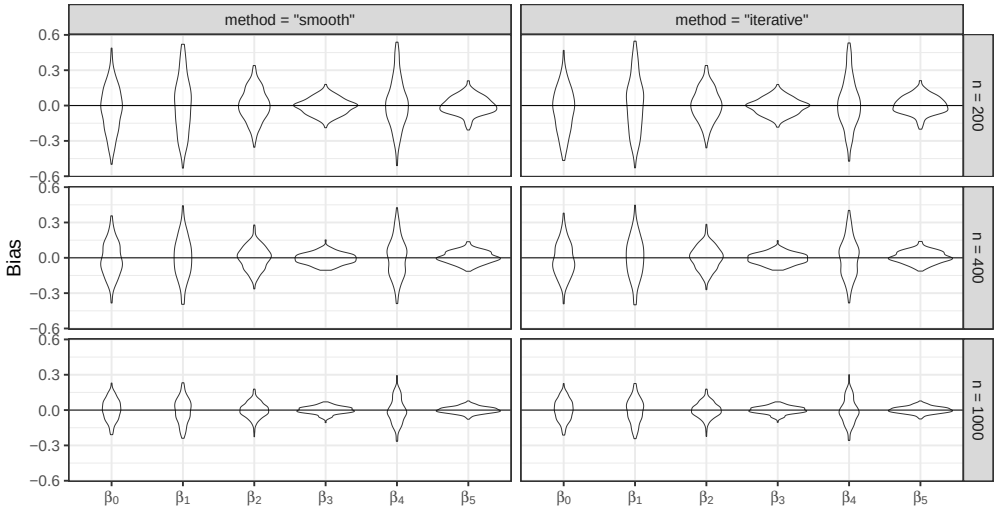
```
> do_pmb <- function(n, t0, cen, Q, nB) {
+   dat <- data.gen(n, t0, cen, Q)
+   fm <- Surv(Time, status) ~ X1 + X2 + X3 + X4 + X5
+   stamp <- NULL
+   stamp[1] <- Sys.time()
+   f1 <- qris(fm, data = dat, t0 = t0, Q = Q, nB = nB, method = "smooth", se = "pmb")
+   stamp[2] <- Sys.time()
+   f2 <- qris(fm, data = dat, t0 = t0, Q = Q, nB = nB, method = "iterative", se = "pmb")
+   stamp[3] <- Sys.time()
+   list(smooth = c(f1$coef, f1$std),
+     iter = c(f2$coef, f2$std),
+     times = diff(stamp))
+ }
> B <- 200
> set.seed(2)
> sims0_pmb <- mapply(function(n, t0)
+   replicate(B, do_pmb(n, t0 = t0, cen = .3, Q = .5, nB = 200)),
+   n = c(200, 400, 1000), t0 = c(0, 0, 0), SIMPLIFY = F)
> sims1_pmb <- mapply(function(n, t0)
+   replicate(B, do_pmb(n, t0 = t0, cen = .3, Q = .5, nB = 200)),
+   n = c(200, 400, 1000), t0 = c(1, 1, 1), SIMPLIFY = F)
```

The simulation results obtained using the partial multiplier bootstrapping approach are presented in Figure 2 and Tables 7 – 12 in the Supplementary Materials, while the computing times are displayed in the lower panel of Table 1. Overall, the estimation results obtained using `se = "pmb"` in Figure 2 closely resemble those in Figure 1 with `se = "fmb"`. As seen in Tables 7 and 8, the ESEs from the non-iterative and iterative methods are comparable, while the ASEs slightly overestimate the ESEs when the sample size is small. The gaps are slightly smaller for the iterative method, as shown in some cases (Johnson and Strawderman, 2009; Kim et al., 2021). The magnitudes of the differences are not large, and they also become smaller when the sample size reaches $n = 1000$. More importantly, the computing times with `se = "pmb"` show significant speed improvements compared to when `se = "fmb"` is used in every case; we observed up to 79% timing improvements.

After confirming the satisfactory performance of the proposed methodologies, we now proceed to illustrate the application of the `init` argument. This argument controls the initial values assigned to the root-finding algorithm's estimates and the plotting capacity of the **qris** package. For this illustrative example, we consider a simpler simulation scenario that involves a single binary covariate. This simplified simulation can be generated using the revised version of the `data.gen()` function provided below.



(a) $t_0 = 0$



(b) $t_0 = 1$

Figure 2: Comparison of the `smooth` and `iterative` estimators with `se = "pmb"` under 30% censoring and $\tau = 0.5$.

```
> ## Global parameters
+   rho0 <- .2 * sqrt(log(2))
+   rho1 <- .1 * sqrt(log(2))
> data.gen <- function(n) {
+   dat <- data.frame(censoring = runif(n, 0, 23.41),
+                     Time0 = sqrt(-log(1 - runif(n))),
+                     X = rbinom(n, 1, .5))
+   dat$Time0 <- ifelse(dat$X > 0, dat$Time0 / rho1, dat$Time0 / rho0)
+   dat$Time <- pmin(dat$Time0, dat$censoring)
+   dat$status <- 1 * (dat$Time0 < dat$censoring)
+   subset(dat, select = c(Time, status, X))
+ }
> set.seed(10)
> head(dat <- data.gen(200))
       Time status X
1  6.034713      1 1
2  7.181451      0 1
3  9.993908      0 1
```

```
4 16.225520      0 1
5  1.993033      0 1
6  5.277471      0 0
```

The updated `data.gen()` function returns a `data.frame` comprising three variables: `Time`, `status`, and `X`, representing the observed survival time, event indicator, and binary covariate, respectively. We will first illustrate the usage of the argument `init` by considering three different initial values: `init = "rq"`, `init = c(1,1)`, and a random vector `init = rnorm(2)`, all used in conjunction with the smooth estimator `method = "smooth"`. The following codes provide an example with different initial values.

```
> (random <- rnorm(2))
[1] 1.5025446 0.5904095
> f1 <- qris(Surv(Time, status) ~ X, data = dat, t0 = 1, init = "rq", nB = 0)
> f2 <- update(f1, init = c(1, 1))
> f3 <- update(f1, init = random)
> all.equal(f1$coef, f2$coef)
[1] TRUE
> all.equal(f2$coef, f3$coef)
[1] TRUE
```

The '`qris`' object, with its `call` component, is compatible with the `update()` function, a built-in function commonly used for updating the attributes of an existing object without requiring redundant and repetitive code. In the example above, we used the `update()` function to modify the initial value specification in `f1`. We observed that different initial values yield identical point estimates, thereby affirming the robustness of the proposed method against fluctuations in initial values.

The covariate effects, along with their associated 95% point-wise confidence intervals across various quantiles or base times, can be visually assessed by applying the generic function `plot()` to a '`qris`' object. We demonstrate this feature using the following `qris` fit, where the standard errors are obtained using `se = "pmb"`, $t_0 = 1$, and all other parameters are set to their default values. We update the `qris` fit with extended quantiles over $0.4, 0.5, 0.6, 0.7$ and plot the covariate effects against these quantiles using the `plot()` function.

```
> fit <- qris(Surv(Time, status) ~ X, data = dat, t0 = 1, se = "pmb")
> fit2 <- qris.extend(fit, Qs = 4:7 / 10)
```

The extended '`qris`' fit generated by the `qris.extend()` function inherits all the attributes from the original '`qris`' object and includes additional `ggdat` components. The following code compares the components of the returned values from the extended '`qris`' fit and the original '`qris`' fit.

```
> class(fit2)
[1] "qris"
> names(fit)
[1] "call"        "coefficient" "data"        "formula"     "para"
[6] "stderr"      "varNames"    "vcov"
> setdiff(names(fit2), names(fit))
[1] "ggdat"
```

Specifically, the extended '`qris`' fit inherits `call`, `coefficient`, `para`, `stderr`, `varNames`, and `vcov` from the original '`qris`' object. The `call` component is the function call from the original `qris()` fit, while `coefficient`, `stderr`, and `vcov` are used to store the point estimates, standard error estimates, and covariance matrix, respectively. The `para` component is a list containing the parameters specified during the fitting of the quantile regression model, and `varNames` is a character string representing the variable names in the function call. The newly added values are `ggdat` and `gg`. The `ggdat` is a data frame containing covariate information generated under the different quantiles and base times specified in the `qris.extend()`. Finally, the corresponding covariate effect plot can be generated by plotting the extended '`qris`' fit as follows.

```
> plot(fit2)
```

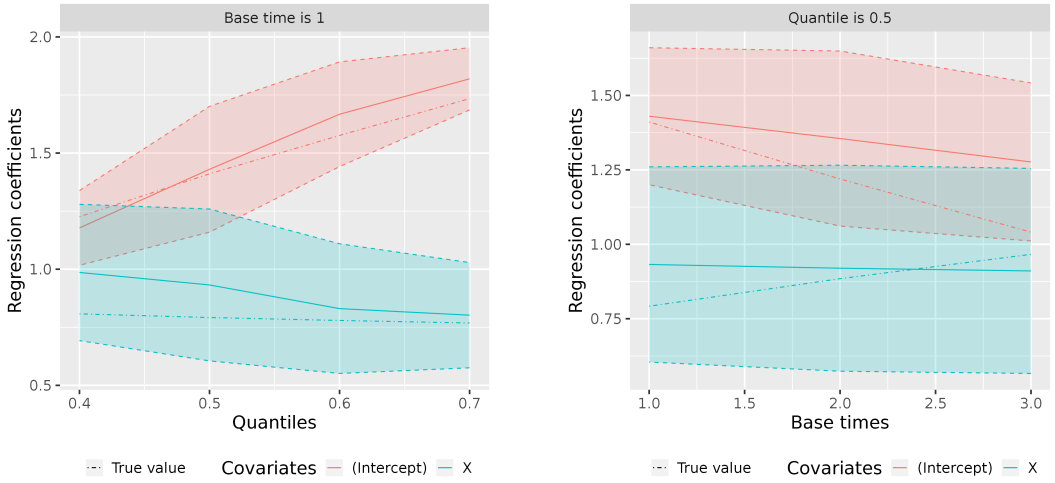
The true values of β's at different quantiles and base times, computed from Equation (7), can be implemented in the following commands.

```
> ## Global parameters
> r <- 2:1 * sqrt(log(2)) / 10
> k <- 2
> ## Function to calculate true beta
> trueB <- function(t0, tau) {
```

```
+     b <- log(1 / r * ((r * t0) ^ k - log(1 - tau))^(1 / k) - t0)
+     c(b[1], b[2] - b[1])
+ }
> ## True beta calculation
> true_Q <- c(t(sapply(4:7 / 10, trueB, t0 = 1)))
> true_t0 <- c(t(sapply(1:3, trueB, tau = .5)))
```

The following code extends the 'ggplot' objects generated by `plot.qris()` by adding additional layers of true value curves and incorporating various `ggplot` options. The resulting figures, Figure 3a and Figure 3b, present the output based on whether the covariate effects are plotted against quantiles or base times, respectively. This observed trend aligns with the specifications described in Equation (7), where increasing τ corresponds to an increasing β_0 while keeping ρ and X fixed. On the other hand, the covariate effect does not change with quantiles but slightly increases with base times, echoing the model specification where β_0 is inversely related to t_0 and β_1 increases as t_0 increases.

```
> library(ggplot2)
> plot(fit2) + theme(legend.position = "bottom") +
+    geom_line(aes(x = Qs, y = true_Q, col = variable, linetype = "True value")) +
+    scale_linetype_manual(name = "", values = c("True value" = "dotdash"))
> b <- plot(fit2, t0s = 1:3, byQs = F)
> b + theme(legend.position = "bottom") +
+    geom_line(aes(x = t0s, y = true_t0, col = variable,
+                  linetype = "True value")) +
+    scale_linetype_manual(name = "", values = c("True value" = "dotdash"))
```



(a) Plot for $Q \in \{0.4, \ldots, 0.7\}$ at $t_0 = 1$

(b) Plot for $t_0 \in \{1, \ldots, 3\}$ at $Q = 0.5$

Figure 3: (a) Estimated effects of covariate with the associated 95% pointwise confidence intervals for quantiles ranging from 0.4 to 0.7 at $t_0 = 1$. Red and blue solid lines are the point estimates of regression parameters for intercept and covariate X, respectively. Similarly, red and blue dotted lines are the upper and lower bounds of 95% pointwise confidence intervals for intercept and covariate X, respectively. (b) Estimated effects of covariate with the associated 95% pointwise confidence intervals for base times ranging from 1 to 3 at $\tau = 0.5$. Red and blue solid lines are the point estimates of regression parameters for intercept and covariate X, respectively. Similarly, red and blue dotted lines are the upper and lower bounds of 95% pointwise confidence intervals for intercept and covariate X, respectively.

4.2 North Central Cancer Treatment Group Lung Cancer Data

The North Central Cancer Treatment Group Lung Cancer Data records the survival of patients with advanced lung cancer, along with assessments of the patients' performance status measured by both physicians and the patients themselves (Loprinzi et al., 1994). The original objective of the study was to ascertain whether descriptive information from a patient-completed questionnaire could offer prognostic insights. The original objective of the study was to determine whether descriptive information from a patient-completed questionnaire could provide prognostic information. However,

for this illustration, we focus on how gender and weight loss affect the quantiles of residual life for patients diagnosed with advanced lung cancer at different time points. The lung cancer data are publicly available from the **survival** package (Therneau, 2021) as `lung`. The following code displays the structure of the `lung` dataset with variables of interest.

```
> data(cancer, package = "survival")
> str(subset(lung, select = c(time, status, sex, wt.loss)))
'data.frame':        228 obs. of  4 variables:
$ time   : num  306 455 1010 210 883 ...
$ status : num  2 2 1 2 2 1 2 2 2 2 ...
$ sex    : num  1 1 1 1 1 1 2 2 1 1 ...
$ wt.loss: num  NA 15 15 11 0 0 10 1 16 34 ...
```

The `lung` data contains 228 patients whose observed survival times in days and censoring status (1 = censored, 2 = dead) are recorded in the `time` and the `status` columns, respectively. Although the censoring status in this dataset is not recorded in the typical 0-1 fashion, the `Surv()` function is still applicable to create the corresponding "`Surv`" object. The `lung` data yields a censoring rate of 27.6% with a median survival time of 310 days. The covariates of interest are gender (`sex = 1` if male, `sex = 2` if female) and weight loss (`wt.loss`). In the following, we use the proposed semiparametric quantile regression models to assess the gender and standardized weight loss effects on different quantiles of residual life at different base times.

We first model the median residual life (`Q = 0.5`) when the base time is one month (`t0 = 30`). Since the estimated median survival times for combined lung cancers are typically less than one year, with a range of 8 to 13 months (Siegel et al., 2021), setting the base time at one month provides insight into how gender and weight loss impact the residual time in early follow-up. In the following, we obtain the regression coefficient estimates using the induced smoothing functions and the corresponding variance estimate with the partial multiplier bootstrap approach.

```
> lung$male <- factor(lung$sex, 1:2, c("Male", "Female"))
> lung$std.wt.loss <- scale(lung$wt.loss)
> fit1 <- qris(Surv(time, status) ~ male + std.wt.loss,
+              data = lung, t0 = 30, Q = .5, nB = 100,
+              method = "smooth", se = "pmb")
> summary(fit1)
Call:
qris(formula = Surv(time, status) ~ male + std.wt.loss,
data = lung, t0 = 30, Q = 0.5, nB = 100, method = "smooth",
se = "pmb")

qris Estimator
              estimate std.Error z.value p.value
(Intercept)     5.5611    0.0950  58.550  <2e-16 ***
maleFemale      0.4804    0.1805   2.661  0.0078 **
std.wt.loss    -0.0731    0.0837  -0.874  0.3824
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Subjects with missing values (in any of the variables relevant for the modeling task) are automatically removed when `qris()` is called. The estimated intercept implies that the median residual life for patients who have survived up to 30 days is $\exp(5.5611) = 260.1$ days for a male with an average weight loss. More interestingly, the summary shows that the gender effect is statistically significant at the 0.05 significance level, indicating that a female patient is expected to have a median residual life at 30 days that is $\exp(0.4804) = 1.617$ times that of a male patient with the same weight loss. The effect of the weight loss is not statistically significant at the 0.05 level. In addition to `summary()`, important statistics such as the coefficient and variance estimates can be extracted by `S3` methods `coef()` and `vcov()`, respectively.

```
> coef(fit1)
(Intercept)    maleFemale    std.wt.loss
5.56111984     0.48044228    -0.07307635
> vcov(fit1)
              (Intercept)   maleFemale    std.wt.loss
(Intercept)    0.009021459 -0.010944549   -0.003074041
maleFemale    -0.010944549  0.032594288    0.002847148
std.wt.loss   -0.003074041  0.002847148    0.006998314
```

Moreover, the corresponding 95% Wald-type confidence interval can be printed by applying the `confint()` function to the 'qris' object.

```
> confint(fit1)
                  2.5 %     97.5 %
(Intercept)   5.3749598 5.74727989
maleFemale    0.1265926 0.83429199
std.wt.loss  -0.2370390 0.09088626
```

The `update()` function can be conveniently applied to update existing 'qris' objects. The following examples update the `method` and `se` arguments from `fit1`. The updated results yield similar coefficient estimates, but the non-smooth procedure (`method = "nonsmooth"`) yields slightly greater standard error estimates.

```
> summary(fit2 <- update(fit1, method = "nonsmooth", se = "fmb"))
Call:
qris(formula = Surv(time, status) ~ male + std.wt.loss,
data = lung, t0 = 30, Q = 0.5, nB = 100, method = "nonsmooth",
se = "fmb")

qris Estimator
             estimate std.Error z.value p.value
(Intercept)    5.5585    0.1132  49.106  <2e-16 ***
maleFemale     0.4695    0.2015   2.331  0.0198 *
std.wt.loss   -0.0668    0.1029  -0.650  0.5159
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> summary(update(fit1, method = "iterative"))
Call:
qris(formula = Surv(time, status) ~ male + std.wt.loss,
data = lung, t0 = 30, Q = 0.5, nB = 100, method = "iterative",
se = "pmb")

qris Estimator
             estimate std.Error z.value p.value
(Intercept)    5.5605    0.1016  54.712  <2e-16 ***
maleFemale     0.4807    0.1626   2.957  0.0031 **
std.wt.loss   -0.0720    0.0903  -0.797  0.4252
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

At a lower (`Q = 0.25`) and a higher (`Q = 0.75`) quantiles, the gender effect remains significant at the 0.05 significance level indicating female patients are associated with longer lower-quantile and higher-quantile residual life than male patients with the same weight loss. Among these models, we observed that female patients tend to have higher coefficient estimates when fitting higher-quantile residual life. While the sign of the estimated regression coefficient for weight loss changes to a negative value when considering the lower quantile, the effects remain statistically insignificant for both the lower and higher quantiles.

```
> summary(update(fit1, Q = 0.25))
Call:
qris(formula = Surv(time, status) ~ male + std.wt.loss,
data = lung, t0 = 30, Q = 0.25, nB = 100, method = "smooth",
se = "pmb")

qris Estimator
            estimate std.Error z.value p.value
(Intercept)   4.9111    0.1034  47.480  <2e-16 ***
maleFemale    0.4651    0.2041   2.279  0.0227 *
std.wt.loss   0.0543    0.0584   0.930  0.3525
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> summary(update(fit1, Q = 0.75))
Call:
```

```
qris(formula = Surv(time, status) ~ male + std.wt.loss,
data = lung, t0 = 30, Q = 0.75, nB = 100, method = "smooth",
se = "pmb")

qris Estimator
             estimate std.Error z.value p.value
(Intercept)    6.0748    0.1063  57.126  <2e-16 ***
maleFemale     0.5237    0.1487   3.522  0.0004 ***
std.wt.loss   -0.0171    0.1166  -0.147  0.8835
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

We also consider the base time at six months `t0 = 180`, which enables us to assess gender and weight loss effects in median residual time at a moderate length of follow-up. The estimated effect for the gender and weight loss increases as t_0 increases from 30 days to 180 days and becomes significant at the 0.05 significant level. Additionally, the effect of the weight loss seems to be associated with a shorter survival time after 180 days, with a p-value of 0.0008.

```
> summary(update(fit1, t0 = 180))
Call:
qris(formula = Surv(time, status) ~ male + std.wt.loss,
data = lung, t0 = 180, Q = 0.5, nB = 100, method = "smooth",
se = "pmb")

qris Estimator
             estimate std.Error z.value p.value
(Intercept)    5.2243    0.0912  57.255  <2e-16 ***
maleFemale     0.5821    0.1867   3.117  0.0018 **
std.wt.loss   -0.2515    0.0754  -3.337  0.0008 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

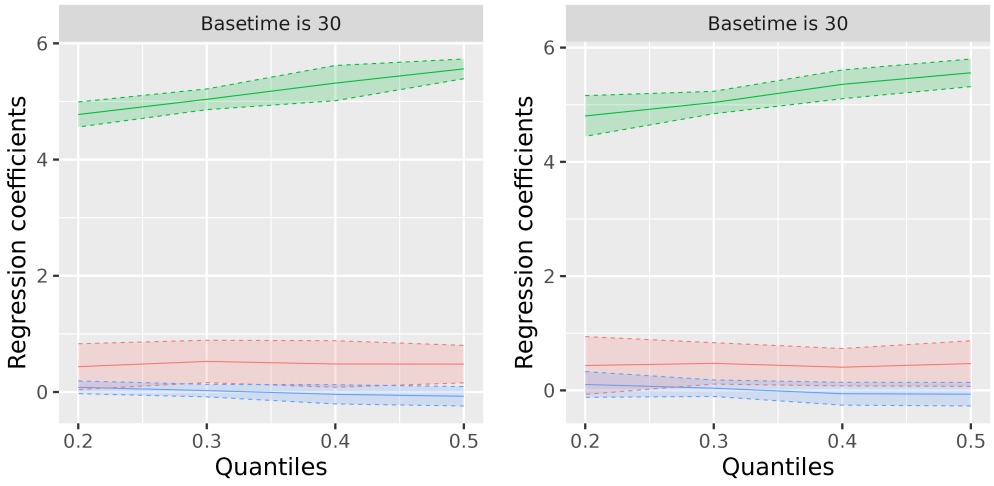
The ‘`qris`’ object is designed to be compatible with `S3` methods: `predict()` and `residuals()` functions. The following presents the fitted survival times for two hypothetical male and female patients with no weight loss, as well as the first five residual values for the dataset.

```
> lung.new <- data.frame(male = c("Male", "Female"), std.wt.loss = 0)
> predict(fit2, newdata = lung.new)
       1        2
444.9026 289.4422
> head(residuals(fit2), 5)
       1          2          3          4          5
-20.86127 -575.86127  232.44474 -416.82295 -555.82295
```

To better understand the covariate effects on different quantiles of residual time and across different base times, we plot the estimated regression coefficients of the intercept, sex, and weight loss in `fit1` and `fit2`. Figures 4a and 4b display the estimated regression coefficients when `method = "smooth"` and `method = "nonsmooth"`, respectively, at different quantiles ranging from 0.2 and 0.5 at $t_0 = 30$ days. The `plot.qris()` function is currently not available for the iterative estimator. This is mainly due to an extended computation time involved, as indicated by our simulation results, and the nature of plotting that necessitates computations across various quantiles or base times. As expected, the two plots show very similar patterns. We plot the estimated regression coefficients of the intercept, sex, and weight loss for different quantiles in the range of 0.2 to 0.5 at $t_0 = 50$, 60, 70, and 80 days (Figure 4c), as well as for different base times in the range of 50 to 80 days at $\tau = 0.2$, 0.3, 0.4, and 0.5 (Figure 4d). The estimation method used is non-iterative induced smoothed estimation (`method = "smooth"`). In Figure 4c, the estimated intercept increases as the quantile increases (for a given base time). The estimated slopes for sex remain largely the same, but those for weight loss tend to decrease slightly across different quantiles (for a given base time). These patterns remain consistent for different base times. In Figure 4d, the estimated intercepts increase as the quantiles increase, but with a given quantile, they remain flat across the different base times considered. The estimated regression coefficients for the two covariates do not appear to change significantly for different base times.

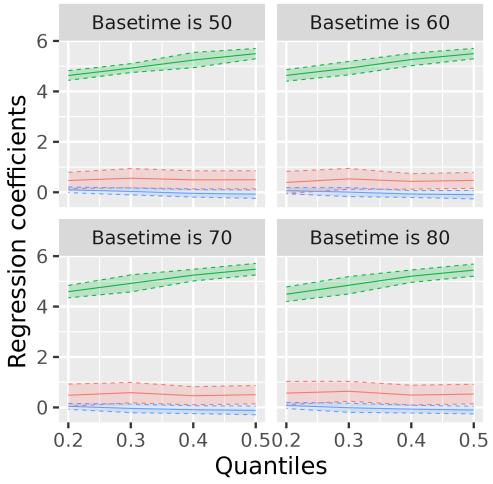
```
> hide <- theme(legend.position = "none")
> plot(fit1, Qs = 2:5 / 10, byQs = TRUE, ggextra = hide)
> plot(fit2, Qs = 2:5 / 10, byQs = TRUE, ggextra = hide)
```

```
> plot(fit1, Qs = 2:5 / 10, t0s = 5:8 * 10, byQs = TRUE, ggextra = hide)
> plot(fit1, Qs = 2:5 / 10, t0s = 5:8 * 10, byQs = FALSE, ggextra = hide)
```

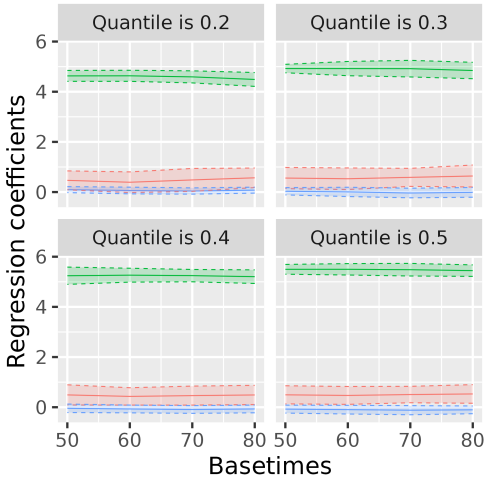


(a) `method = "smooth"` and `se = "pmb"`

(b) `method = "nonsmooth"` and `se = "fmb"`



(c) `method = "smooth"` and `se = "pmb"`



(d) Multiple covariate effect plot against base time

Figure 4: Green, red and blue lines are the point estimates of regression parameters for intercept, covariate sex and covariate weight loss, respectively. Solid line and dotted line are the point estimates and the upper and lower bounds of 95% pointwise confidence intervals for each regression coefficient. (a) `method = "smooth"` and `se = "pmb"` ($\tau = 0.2, 0.3, 0.4, 0.5, t_0 = 30$) (b) `method = "nonsmooth"` and `se = "fmb"` ($\tau = 0.2, 0.3, 0.4, 0.5, t_0 = 30$) (c) `method = "smooth"` and `se = "pmb"` against quantiles ($\tau = 0.2, 0.3, 0.4, 0.5, t_0 = 50, 60, 70, 80$) (d) `method = "smooth"` and `se = "pmb"` against base times ($\tau = 0.2, 0.3, 0.4, 0.5, t_0 = 50, 60, 70, 80$)

5 Conclusion

The purpose of the qris package is to provide a comprehensive tool for fitting quantile regression models on residual life for right-censored survival data, with the aim of promoting widespread dissemination and utilization. This package implements one estimation method based on non-smooth estimating functions and two estimation methods based on their induced smoothed versions. The non-smooth estimator is calculated through L_1-type minimization while incorporating the IPCW technique, and its variance is calculated using full multiplier bootstrapping. The first type of the induced smoothed estimator, a non-iterative version, directly solves estimating functions, and its variance can be calculated using either the full multiplier bootstrapping or the robust sandwich form with partial multiplier bootstrapping. As evidenced by the simulation results, this enables one to substantially reduce computing times without sacrificing estimation accuracy and stability

compared to the original non-smooth function-based method. The iterative smoothed estimator has an advantage in obtaining more precise estimates than its non-iterative version, although it requires longer computing times. For all these methods, estimates of the regression coefficients and their variances can be calculated at user-defined quantiles and base times, as long as they are identifiable. Additionally, the package provides features for plotting estimates with associated 95% confidence intervals against quantiles and base times using the generic `plot` function. These plots visualize patterns of estimates at different quantiles and base times, helping users to easily grasp the overall picture. The package **qris** and its included functions are verified through illustrations using simulated data with interpretation of the results demonstrated through a real data application.

Some possible directions for extending our package are as follows. Efforts can be made to reduce the computational burden associated with variance estimation, which currently accounts for a significant portion of the computing time. In particular, the iterative-induced smoothed method employs the partial multiplier bootstrap method to calculate variance estimates in each iteration. Since this method requires multiple iterations, it is crucial to explore more computationally efficient variance estimation procedures for each iteration to reduce the currently relatively longer computation time. One approach is to utilize a closed-form estimation of the mid-part of the sandwich-type variance, as discussed in Chiou et al. (2014); Choi et al. (2018). Implementing this direct variance estimation in each iteration is expected to further enhance computation efficiency. Another direction is to generalize the approaches to allow for the inclusion of sampling weights, which is useful for bias correction when failure time data are generated from non-random sampling designs, such as case-cohort designs (Prentice, 1986; Chiou et al., 2015b). The current estimating functions implemented in the **qris** package assume that the data are randomly sampled, with sampling weights set to 1." To the best of our knowledge, there is a lack of model-checking procedures and model-comparison methods specifically designed for the non-smooth estimator, and a logical next step would be to develop these procedures for subsequent integration into the package.

References

R. Alhamzawi. *Brq: Bayesian Analysis of Quantile Regression Models*, 2020. URL https://CRAN.R-project.org/package=Brq. R package version 3.0. [p115]

B. Brown and Y.-G. Wang. Standard errors and covariance matrices for smoothed rank estimators. *Biometrika*, 92(1):149–158, 2005. URL https://doi.org/10.1093/biomet/92.1.149. [p114]

P.-C. Bürkner. Advanced Bayesian multilevel modeling with the R package brms. *The R Journal*, 10(1):395–411, 2018. doi: 10.32614/RJ-2018-017. [p115]

Y. Chen, N. Jewell, X. Lei, and S. Cheng. Semiparametric estimation of proportional mean residual life model in presence of censoring. *Biometrics*, 61(1):170–178, 2005. URL https://doi.org/10.1111/j.0006-341X.2005.030224.x. [p114]

Y. Q. Chen. Additive expectancy regression. *Journal of the American Statistical Association*, 102 (477):153–166, 2007. URL https://doi.org/10.1198/016214506000000870. [p114]

Y. Q. Chen and S. Cheng. Linear life expectancy regression with censored data. *Biometrika*, 93(2): 303–313, 2006. URL https://doi.org/10.1093/biomet/93.2.303. [p114]

S. Chiou, S. Kang, and J. Yan. Rank-based estimating equations with general weight for accelerated failure time models: An induced smoothing approach. *Statistics in Medicine*, 34(9):1495–1510, 2015a. URL https://doi.org/10.1002/sim.6415. [p117]

S. H. Chiou, S. Kang, and J. Yan. Fast accelerated failure time modeling for case-cohort data. *Statistics and Computing*, 24(4):559–568, 2014. URL https://doi.org/10.1007/s11222-013-9388-2. [p118, 131]

S. H. Chiou, S. Kang, and J. Yan. Semiparametric accelerated failure time modeling for clustered failure times from stratified sampling. *Journal of the American Statistical Association*, 110(510): 621–629, 2015b. URL https://doi.org/10.1080/01621459.2014.917978. [p115, 118, 131]

S. H. Chiou, S. Kang, and J. Yan. *aftgee: Accelerated failure time model with generalized estimating equations*, 2021. URL https://CRAN.R-project.org/package=aftgee. R package version 1.1.6. [p115, 118]

S. Choi, S. Kang, and X. Huang. Smoothed quantile regression analysis of competing risks. *Biometrical Journal*, 60(5):934–946, 2018. URL https://doi.org/10.1002/bimj.201700104. [p115, 118, 131]

S. Dlugosz, L. Peng, R. Li, and S. Shi. *cmprskQR: Analysis of competing risks using quantile regressions*, 2019. URL https://CRAN.R-project.org/package=cmprskQR. R package version 0.9.2. [p115]

D. Eddelbuettel, R. Francois, J. Allaire, K. Ushey, Q. Kou, N. Russell, I. Ucar, D. Bates, and J. Chambers. *Rcpp: Seamless R and C++ Integration*, 2022a. URL https://CRAN.R-project.org/package=Rcpp. R package version 1.0.9. [p118]

D. Eddelbuettel, R. Francois, D. Bates, B. Ni, and C. Sanderson. *RcppArmadillo: 'Rcpp' Integration for the 'Armadillo' Templated Linear Algebra Library*, 2022b. URL https://CRAN.R-project.org/package=RcppArmadillo. R package version 0.11.1.1.0. [p118]

P. Frumento. *ctqr: Censored and truncated quantile regression*, 2021. URL https://CRAN.R-project.org/package=ctqr. R package version 2.0. [p115]

Y. Huang. Quantile calculus and censored regression. *Annals of Statistics*, 38(3):1607, 2010. doi: 10.1214/09-AOS771. [p114]

L. M. Johnson and R. L. Strawderman. Induced smoothing for the semiparametric accelerated failure time model: Asymptotics and extensions to clustered data. *Biometrika*, 96(3):577–590, 2009. URL https://doi.org/10.1093/biomet/asp025. [p115, 118, 123]

S.-H. Jung. Quasi-likelihood for median regression models. *Journal of the American Statistical Association*, 91(433):251–257, 1996. URL https://doi.org/10.1080/01621459.1996.10476683. [p114]

S.-H. Jung, J.-H. Jeong, and H. Bandos. Regression on quantile residual life. *Biometrics*, 65(4): 1203–1212, 2009. URL https://doi.org/10.1111/j.1541-0420.2009.01196.x. [p114]

S. Kang. Fitting semiparametric accelerated failure time models for nested case–control data. *Journal of Statistical Computation and Simulation*, 87(4):652–663, 2017. URL https://doi.org/10.1080/00949655.2016.1222611. [p115]

K. Kim, J. Ko, and S. Kang. Comparison of variance estimation methods in semiparametric accelerated failure time models for multivariate failure time data. *Japanese Journal of Statistics and Data Science*, 4(2):1179–1202, 2021. URL https://doi.org/10.1007/s42081-021-00126-y. [p123]

K. H. Kim, S. Kang, and S. H. Chiou. *qris: Quantile regression model for residual lifetime using an induced smoothing approach*, 2022. URL https://CRAN.R-project.org/package=qris. R package version 1.0.0. [p114, 115]

K. H. Kim, D. J. Caplan, and S. Kang. Smoothed quantile regression for censored residual life. *Computational Statistics*, 38:1001–1022, 2023. URL https://doi.org/10.1007/s00180-022-01262-z. [p115, 116, 117, 121]

M.-O. Kim, M. Zhou, and J.-H. Jeong. Censored quantile regression for residual lifetimes. *Lifetime Data Analysis*, 18(2):177–194, 2012. URL https://doi.org/10.1007/s10985-011-9212-2. [p114, 115, 116]

R. Koenker. *quantreg: Quantile regression*, 2022. URL https://CRAN.R-project.org/package=quantreg. R package version 5.87. [p115, 122]

R. Koenker and G. Bassett Jr. Regression quantiles. *Econometrica: Journal of the Econometric Society*, pages 33–50, 1978. URL https://doi.org/10.2307/1913643. [p114, 116]

R. Koenker and I. Mizera. Penalized triograms: Total variation regularization for bivariate smoothing. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 66(1):145–163, 2004. URL https://doi.org/10.1111/j.1467-9868.2004.00437.x. [p115]

R. Koenker, P. Ng, and S. Portnoy. Quantile smoothing splines. *Biometrika*, 81(4):673–680, 1994. URL https://doi.org/10.1093/biomet/81.4.673. [p115]

R. Li, X. Huang, and J. E. Cortes. Quantile residual life regression with longitudinal biomarker measurements for dynamic prediction. *Journal of the Royal Statistical Society. Series C: Applied Statistics*, 65(5):755–773, 2016. URL http://www.jstor.org/stable/44681854. [p114, 116, 121]

S. Liu and S. K. Ghosh. Regression analysis of mean residual life function. Technical report, North Carolina State University. Dept. of Statistics, 2008. URL https://repository.lib.ncsu.edu/bitstream/handle/1840.4/3041/mimeo2613.pdf?sequence=1. [p114]

C. L. Loprinzi, J. A. Laurie, H. S. Wieand, J. E. Krook, P. J. Novotny, J. W. Kugler, J. Bartel, M. Law, M. Bateman, and N. E. Klatt. Prospective evaluation of prognostic variables from patient-completed questionnaires. North Central Cancer Treatment Group. *Journal of Clinical Oncology*, 12(3):601–607, 1994. URL https://doi.org/10.1200/JCO.1994.12.3.601. [p126]

G. Maguluri and C.-H. Zhang. Estimation in the mean residual life regression model. *Journal of the Royal Statistical Society: Series B (Methodological)*, 56(3):477–489, 1994. URL https://doi.org/10.1111/j.2517-6161.1994.tb01994.x. [p114]

D. Oakes and T. Dasu. A note on residual life. *Biometrika*, 77(2):409–410, 1990. URL https://doi.org/10.1093/biomet/77.2.409. [p114]

D. Oakes and T. Dasu. Inference for the proportional mean residual life model. *Lecture Notes-Monograph Series*, pages 105–116, 2003. URL http://www.jstor.org/stable/4356266. [p114]

L. Peng and Y. Huang. Survival analysis with quantile regression models. *Journal of the American Statistical Association*, 103(482):637–649, 2008. URL https://doi.org/10.1198/016214508000000355. [p114, 115]

S. Portnoy. Censored regression quantiles. *Journal of the American Statistical Association*, 98(464): 1001–1012, 2003. URL https://doi.org/10.1198/016214503000000954. [p114, 115]

S. Portnoy and R. Koenker. The gaussian hare and the laplacian tortoise: computability of squared-error versus absolute-error estimators. *Statistical Science*, 12(4):279–300, 1997. URL https://doi.org/10.1214/ss/1030037960. [p114]

J. L. Powell. Censored regression quantiles. *Journal of Econometrics*, 32(1):143–155, 1986. URL https://doi.org/10.1016/0304-4076(86)90016-3. [p115]

R. L. Prentice. A case-cohort design for epidemiologic cohort studies and disease prevention trials. *Biometrika*, 73(1):1–11, 1986. URL https://doi.org/10.1093/biomet/73.1.1. [p131]

R Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2021. URL https://www.R-project.org/. [p120]

R. L. Siegel, K. D. Miller, H. E. Fuchs, and A. Jemal. Cancer statistics, 2021. *CA: A Cancer Journal for Clinicians*, 71(1):7–33, 2021. URL https://doi.org/10.3322/caac.21654. [p127]

T. M. Therneau. *survival: Survival analysis*, 2021. URL https://CRAN.R-project.org/package=survival. R package version 3.2-13. [p118, 127]

Y. Wei, A. Pere, R. Koenker, and X. He. Quantile regression methods for reference growth charts. *Statistics in Medicine*, 25(8):1369–1382, 2006. URL https://doi.org/10.1002/sim.2271. [p114]

H. Wickham, W. Chang, L. Henry, T. L. Pedersen, K. Takahashi, C. Wilke, K. Woo, H. Yutani, and D. Dunnington. *ggplot2: Create elegant data visualisations using the grammar of graphics*, 2022. URL https://CRAN.R-project.org/package=ggplot2. R package version 3.3.6. [p115]

Z. Ying, S.-H. Jung, and L.-J. Wei. Survival analysis with median regression models. *Journal of the American Statistical Association*, 90(429):178–184, 1995. URL https://doi.org/10.1080/01621459.1995.10476500. [p114]

Z. Zhang, X. Zhao, and L. Sun. Goodness-of-fit tests for additive mean residual life model under right censoring. *Lifetime Data Analysis*, 16(3):385–408, 2010. URL https://doi.org/10.1007/s10985-010-9152-2. [p114]

Kyu Hyun Kim
Department of Statistics and Data Science and *Department of Applied Statistics*
Yonsei University
50 Yonsei-ro, Seodaemun-gu, Seoul
Republic of Korea
kyuhyunkim07@yonsei.ac.kr

Sangwook Kang
Department of Statistics and Data Science and *Department of Applied Statistics*
Yonsei University
50 Yonsei-ro, Seodaemun-gu, Seoul

Republic of Korea
kanggi1@yonsei.ac.kr

Sy Han Chiou
Department of Statistics and Data Science
Southern Methodist University
P.O. Box 750332, Dallas, TX
USA
schiou@smu.edu
https://www.sychiou.com/