SMOOTH QUANTILE PROCESSES FOR RIGHT CENSORED DATA

by

KATSUHIRO UECHI

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2013

ACKNOWLEDGEMENTS

I would like to express my gratitude to the entire faculty and staff of the Mathematics Department at The University of Texas at Arlington. In particular, I would like to give my deepest gratitude to Dr. Shan Sun-mitchell, my advisor, for all of her help along the way. None of this would have been possible without her help. I would also like to thank Dr. Nancy Rowe for help with SAS programming.

Without the love and support of my family and friends, this would have been much more difficult. So I would like to thank my parents, Masaharu Iwao, Harumi Uechi and Hitomi Shiokawa, my wife, Erika Uechi and all of my friends who supported me.

March 29, 2013

ABSTRACT

SMOOTH QUANTILE PROCESSES FOR RIGHT CENSORED DATA

Katsuhiro Uechi, Ph.D.

The University of Texas at Arlington, 2013

Supervising Professor: Dr. Shan Sun-Mitchell

The development of an estimator of a quantile function $Q(p)$ is discussed. The smooth nonparametric estimator $\tilde{Q}_n(p)$ of a quantile function $Q(p)$ is defined as the solution to $\tilde{F}_n(\tilde{Q}_n(p)) = p$, where$\tilde{F}_n$ is a smooth Kaplan-Meier estimator of an unknown continuous distribution function $F(x)$. The asymptotic properties of the smooth quantile process, $\sqrt{n}(\tilde{Q}_n(p) - Q(p))$, based on right censored lifetimes are studied. The asymptotic properties of the bootstrap quantile process, $\sqrt{n}(\tilde{Q}^*_n(p) - \tilde{Q}(p))$ are also investigated and shown to have the same limiting distribution as the smooth quantile process. The bootstrap method to approximate the sampling distribution of the smooth quantile process is used to construct simultaneous confidence bands for a quantile function and the difference of two quantile functions. A Monte Carlo simulation is conducted to assess the performance of these confidence bands by computing the lengths and coverage probabilities of the bands. The optimum bandwidth is also investigated.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

It is common that you often encounter right censored data in many statistical areas such as survival analysis. Right censoring occurs when the actual data value is unknown but is known to be above a certain value. In clinical trials, there are many cases where the true survival time can not be observed due to the loss of a sampling unit. For instance, the true survival time for a patient can not be recorded because they move and become unable to participate in the study further, die from factors unrelated to the study, or etc.

In this thesis, we study the asymptotic properties of the smooth quantile process $\sqrt{n}(\tilde{Q}_n(p) - Q(p))$ based on right censored data. The bootstrap method which approximates the distributions of smooth quantile processes is investigated and we use it to construct simultaneous confidence bands for quantile functions.

In section 2, we introduce the smooth quantile function estimators which we use for our research applications and simulations. The main results and proofs are given in sections 3 and 4. Later in section 4, we discuss an application of the developed results to the construction of simultaneous confidence bands for the difference of two quantile functions. These confidence bands are then used to test whether two distributions F_1 and F_2 belong to the same family with a location shift. In clinical trials, we often want to compare two treatments and determine if there is a difference between them. One may prefer to use parametric tests if the assumptions of normality, homogeneity and others. But if these assumptions are violated, non-parametric tests may have more advantages. They do, however, have various assumptions that must be met.

It is important not to be confused by not having the need to meet an assumption of "normality" with the notion of "assumptionless." One of the assumptions is that samples are drawn from the same distribution family. And this application allows one to check this assumption. In section 5, optimal bandwidths are studied. In section 6, we carry out a Monte Carlo simulation to assess the performance of the proposed confidence bands.

CHAPTER 2

Smooth quantile estimators

Let X be the survival time of an individual with an unknown continuous distribution function $F(x)$ and its quantile function $Q(p) \equiv F^{-1}(p) = \inf\{x \mid F(x) \geq p\}$, $0 < p < 1$. Let $X_1, ..., X_n$ be $i.i.d$ copies of X. X_i's may be right censored and may not be observed. Let C be the censoring time from another unknown distribution $G(x)$. If $C_1, ..., C_n$ are $i.i.d$ copies of C and if X_i's and C_i's are independent, then we observe $\{\tilde{X}_i, \delta_i\}, i = 1, ..., n$, where $\tilde{X}_i = \min(X_i, C_i)$ and $\delta_i = I(X_i \leq C_i)$. $\delta_i = 1$ indicates that the survival time X_i for the ith individual is observed and $\delta_i = 0$ indicates that the value of X_i is not observed but is known to be greater than C_i. The distribution function $H(x)$ of $\tilde{X}_i$ is defined by

$$H(x) = 1 - (1 - F(x))(1 - G(x)) \tag{2.1}$$

Kaplan and Meier [7] proposed the following product limit estimator of the survival function $1 - F(x)$ based on the right censored data $\{\tilde{X}_i, \delta_i\}$, $i = 1, .., n$.

$$1 - \hat{F}_n(x) = \begin{cases} 1 & 0 \leq x \leq \tilde{X}_{(1)} \\ \prod_{i=1}^{k-1} (\frac{n-i}{n-i+1})^{\delta_{(i)}} & \tilde{X}_{(k-1)} < x \leq \tilde{X}_{(k)}, k = 2, .., n \\ 0 & \tilde{X}_{(n)} < x, \end{cases} \tag{2.2}$$

where $\tilde{X}_{(1)}, ..., \tilde{X}_{(n)}$ are the ordered $\tilde{X}_i$'s and $\delta_{(i)}$ is the indicator for $\tilde{X}_{(i)}$. $\hat{F}_n(x)$ is called the Kaplan-Meier estimator, which is the most popular estimator used in the study of survival function $1 - F(x)$. Sander [14] proposed an estimator of the quantile function $Q(p)$ by its natural estimator $\hat{Q}_n(p) = \inf\{x \mid \hat{F}_n(x) \geq p\}$ and proved its week convergence. Cheng [2] obtained some asymptotic properties of $\hat{Q}_n(p)$ and

Csorgo [3] discussed the strong approximation results for $\hat{Q}_n(p)$. Padgett [12], Lio, Padgett and Yu [8] and Lio, Padgett, and Thombs studied a kernel smooth quantile estimator $\breve{Q}_n(p)$ from right censored data, extending the complete sample results of Yang [15].

Let $\{h \equiv h_n\}$ be a bandwidth sequence of positive numbers so that $h_n \to 0$ as $n \to \infty$, and let k be a probability density function. Then the kernel smoothed quantile estimator is given by

$$\breve{Q}_n(p) = h^{-1} \int_0^1 \hat{Q}_n(p) k((t-p)/h) dt, \ 0 < p < 1. \tag{2.3}$$

An alternative smooth nonparametric estimator of a quantile function was studied by Nadaraya[9] and was extended to the right censored data case by Padgett and Thomas [10] and [11]. That is ,let $\tilde{F}_n$ be the smooth Kaplan-Meier estimator of the distribution function defined by

$$\tilde{F}_n(x) = h^{-1} \int_0^\infty \hat{F}_n(t) k((x-t)/h) dt. \tag{2.4}$$

Then the smooth nonparametric estimator $\tilde{Q}_n(p)$ of the quantile function is defined as the solution to $\tilde{F}_n(\tilde{Q}_n(p)) = p$. The solution can be found iteratively by various numerical computational methods of locating roots of equations, such as bisection method, secant method and Newton-Raphson method. The iterations should converge rapidly if k and F are smooth and well behaved. We will use $\tilde{Q}_n(p)$ thorough out the thesis.

CHAPTER 3

Asymptotic Results of Smooth Quantile Processes

3.1 Assumptions and Definitions

We now give our main theorem and proof of weak convergence of $\tilde{Q}_n(p)$. We assume the following conditions:

$C1$. k is a continuous density function with compact support $[-c, c]$,for some constant c.

$C2$. $h \to 0$ and $\sqrt{n}h \to 0$ as $n \to \infty$.

$C3$. $F(x)$ is continuous and has a bounded density function $f(x)$.

$C4$. $\tau_F \leq \tau_G \leq \infty$, where $\tau_H \equiv sup\{t : H(t) < 1\}$ for any distribution function $H(x)$.

Gaussian processes

A *stochastic process* is defined as a collection of random variables $X(t)$, $t \in T$, defined on a common probability space, where T is a subset of $(-\infty, \infty)$. A stochastic process is called a *Gaussian process* if every finite linear combination of the random variables $X(t)$, $t \in T$, is normally distributed.

The Wiener processes

A stochastic process $W(t), -\infty < t < \infty$ is called the *Wiener process* with parameter σ^2 if the following properties are satisfied:

(1). $W(0) = 0$.

(2). $W(t) - W(s)$ has a normal distribution with mean 0 and

variance $\sigma^2(t-s)$ for $s \leq t$.

(3). $W(t_2) - W(t_1), W(t_3) - W(t_2), ..., W(t_n) - W(t_{n-1})$ are

independent for $t_1 \leq t_2 \leq ... \leq t_n$.

Let $D[a,b]$ be the space of functions on an interval $[a,b]$ that are right continuous and with left limits, equipped with the Skorohod topology. The notation $\xrightarrow{D}$ will be used for week convergence in a Skorohod space. Let $y(s) = (1-F(s))(1-G(s-)), \lambda(s) = f(s)/(1-F(s))$, i.e., the hazard function and $B(s)$ be a Wiener process.

Let

$$Z(x) = (1-F(x)) \int_0^x (\lambda(s)/y(s))^{1/2} dB(s). \tag{3.1}$$

It is shown by Gill that $\sqrt{n}(\hat{F}_n(x) - F(x)) \xrightarrow{D} Z(x)$ in $D[a,b], 0 \leq a < b < \tau_F$. In the following section, we show an analogous result for smoothed estimator of the distribution function.

3.2 Main Results

Theorem 1 *Under* $C1-C4$, *for* $0 \leq a < b < \tau_F$, $\sqrt{n}(\tilde{F}_n(x) - F(x)) \xrightarrow{D} Z(x)$ *in* $D[a,b]$ *as* $n \to \infty$.

proof : Let $F_0(x) = h^{-1} \int_0^\infty F(t) k((x-t)/h) dt$.

Note that

$$\sqrt{n}(\tilde{F}_n(x) - F(x)) = \sqrt{n}(\tilde{F}_n(x) - F_0(x) + F_0(x) - F(x)) \tag{3.2}$$

$$= \sqrt{n}(\tilde{F}_n(x) - F_0(x)) + \sqrt{n}(F_0(x) - F(x)). \tag{3.3}$$

We show that the first term of equation (3.3) converges to $Z(x)$ in $D[0,b]$ and the second term is $O(\sqrt{n}h)$,i.e., it converges to 0 as $n \to \infty$.

For all x such that $0 \leq a \leq x \leq b < \tau_F$, let $\alpha > 0$ be such that $b + \alpha < \tau_F$. Then under condition $C1$,

$$\sqrt{n}(\tilde{F}_n(x) - F_0(x)) = \sqrt{n}\Bigg(\int_0^\infty h^{-1}\hat{F}_n(t)k((x-t)/h)dt - \int_0^\infty h^{-1}F(t)k((x-t)/h)dt \Bigg) \tag{3.4}$$

$$= \sqrt{n}h^{-1}\int_0^\infty (\hat{F}_n(t) - F(t))k((x-t)/h)dt \tag{3.5}$$

$$= h^{-1}\int_0^{b+\alpha} \sqrt{n}(\hat{F}_n(t) - F(t))k((x-t)/h)dt + \sqrt{n}h^{-1}\int_{b+\alpha}^\infty (\hat{F}_n(t) - F(t))k((x-t)/h)dt. \tag{3.6}$$

The second term of equation (3.6)can be shown equal to 0 as follows. By condition $C1$,

$$\sqrt{n}h^{-1}\int_{b+\alpha}^\infty (\hat{F}_n(t) - F(t))k((x-t)/h)dt \leq 2\sqrt{n}h^{-1}\int_{b+\alpha}^\infty k((x-t)/h)dt \tag{3.7}$$

$$= 2\sqrt{n}\int_{-\infty}^{\frac{x-(b+\alpha)}{h}} k(u)du \tag{3.8}$$

$$\leq 2\sqrt{n}\int_{-\infty}^{-c} k(u)du \tag{3.9}$$

$$= 0 \tag{3.10}$$

Since $\sqrt{n}(\hat{F}_n(t) - F(t) \xrightarrow{D} Z(t)$ for $t \in [a,b]$, the first term in equation (3.6) converges in distribution to $h^{-1}\int_0^{b+\alpha} Z(t)k((x-t)/h)dt$. So we now show that $h^{-1}\int_0^\infty Z(t)k((x-t)/h)dt \to Z(x)$ uniformly in $[a,b]$ with probability 1.

Note that

$$h^{-1}\int_0^{b+\alpha} Z(t)k((x-t)/h)dt - Z(x) = \int_{\frac{x-(b+\alpha)}{h}}^{\frac{x}{h}} Z(x-hu)k(u)du - Z(x) \tag{3.11}$$

$$= \int_{-c}^{c}\Big(Z(x-hu)-Z(x)\Big)k(u)du. \tag{3.12}$$

Since $Z(x)$ is continuous, $Z(x-hu)-Z(x)$ converges to zero uniformly in $x\in[a,b]$ with probability 1. Thus $\int_{-c}^{c}\Big(Z(x-hu)-Z(x)\Big)k(u)du \xrightarrow{a.s} 0$ for $x\in[0,b]$. We have shown that $\sqrt{n}(\tilde{F}_n(x)-F_0(x)) \xrightarrow{D} Z(x)$ in $D[a,b]$.

Next, we show that $\sup_{0\le x<\infty}\sqrt{n}\mid F_0(x)-F(x)\mid = O(\sqrt{n}h)$.

$$\sqrt{n}\mid F_0(x)-F(x)\mid = \sqrt{n}\left|h^{-1}\int_0^{\infty} F(t)k((x-t)/h)dt - F(x)\right| \tag{3.13}$$

$$= \sqrt{n}\left|\int_{-\infty}^{x/h} F(x-hu)k(u)du - F(x)\right| \tag{3.14}$$

As $h\to 0$, $x/h \gg c$. So

$$\sqrt{n}\left|\int_{-\infty}^{x/h} F(x-hu)k(u)dt - F(x)\right| = \sqrt{n}\left|\int_{-c}^{c}\Big(F(x-hu)-F(x)\Big)k(u)du\right| \tag{3.15}$$

$$\le \sqrt{n}\int_{-c}^{c}\mid F(x-hu)-F(x)\mid k(u)du \tag{3.16}$$

$$= \sqrt{n}\int_{-c}^{c}\mid f(\xi)\mid huk(u)du, \qquad \xi\in(x-hu,x) \tag{3.17}$$

$$= \sqrt{n}h\mid f(\xi)\mid \int_{-c}^{c} uk(u)du \tag{3.18}$$

$$< \sqrt{n}h\mid f(\xi)\mid \int_{-c}^{c} 2ck(u)du \tag{3.19}$$

$$= 2c\mid f(\xi)\mid \sqrt{n}h \tag{3.20}$$

By condition $C3$, f is bounded. Thus $\sqrt{n}\mid F_0(x)-F(x)\mid \to 0$ with bound $\sqrt{n}h$ as $n\to\infty$. And this completes the proof of theorem 1.

Corollary 1 $\sqrt{n}\tilde{F}_n(0) \xrightarrow{p} 0$ *as* $n \to \infty$.

proof :Let $x = 0$, then theorem 1 becomes

$$\sqrt{n}(\tilde{F}_n(0) - F(0)) \xrightarrow{D} Z(0).$$

in $D[a, b]$ as $n \to \infty$. $F(0) = 0$ since x is survival time, i.e., $x \geq 0$. And

$$\begin{aligned} Z(0) &= (1 - F(0)) \int_0^0 (\lambda(s)/y(s))^{1/2} dB(s) \\ &= 0. \end{aligned}$$

So we have $\sqrt{n}\tilde{F}_n(0) \xrightarrow{D} 0$ as $n \to \infty$. Thus $\sqrt{n}\tilde{F}_n(0) \xrightarrow{P} 0$.

Theorem 2 *Let* $0 < \beta < 1$. *Suppose that* $f(Q(p))$ *is continuous and positive on the interval* $[0, \beta)$. *Then, under* $C1 - C4$,

$\sqrt{n}(\tilde{Q}_n(p) - Q(p)) \xrightarrow{D} Z(Q(p))/f(Q(p))$ *in* $D[0, \beta]$ *as* $n \to \infty$.

proof :

Let b be such that $\beta < F(b) - \epsilon$ for some $\epsilon > 0$. Let $\tilde{F}_n^0(x) = \tilde{F}_n(x) - \tilde{F}_n(0)$. Then $\tilde{F}_n^0(x)$ is non-decreasing and $\tilde{F}_n^0(0) = 0$. Now consider a process $\sqrt{n}(\tilde{F}_n^0(x) - F(x))$. By Theorem 1 and Corollary 1, it is easily seen that $\sqrt{n}(\tilde{F}_n^0(x) - F(x)) \xrightarrow{D} Z(x)$ in $D[a, b]$. Then by Theorem 1 of Doss and Gill [4], we have

$$\sup_{0 \leq p \leq F(b)-\epsilon} | \sqrt{n}(\tilde{F_n^0}^{-1}(p) - Q(p)) + \sqrt{n}\frac{\tilde{F}_n^0(Q(p)) - p}{f(Q(p))} | \xrightarrow{p} 0. \tag{3.21}$$

Note that $\tilde{F_n^0}^{-1}(p) = \tilde{Q}_n(p + \tilde{F}_n(0))$. So (3.21) becomes

$$\sup_{0 \leq p \leq F(b)-\epsilon} | \sqrt{n}(\tilde{Q}_n(p + \tilde{F}_n(0)) - Q(p)) + \sqrt{n}\frac{\tilde{F}_n^0(Q(p)) - p}{f(Q(p))} | \xrightarrow{p} 0. \tag{3.22}$$

By the definition of $\tilde{F}_n^0(x)$, we have

$$\sqrt{n}(\tilde{F}_n^0(Q(p)) - p) = \sqrt{n}(\tilde{F}_n(Q(p)) - \tilde{F}_n(0) - p) \tag{3.23}$$

$$= \sqrt{n}\tilde{F}_n(Q(p)) - \sqrt{n}\tilde{F}_n(0) - \sqrt{n}F(Q(p)) \tag{3.24}$$

By Corollary 1, Theorem 1, and (3.24) above, we see that

$$\sqrt{n}(\tilde{F}_n^0(Q(p)) - p) \xrightarrow{D} Z(Q(p)) \tag{3.25}$$

Thus,

$$\sqrt{n}\frac{\tilde{F}_n^0(Q(p)) - p}{f(Q(p))} \xrightarrow{D} \frac{Z(Q(p))}{f(Q(p))} \tag{3.26}$$

$\forall p \in [0, F(b)]$.

Now, since, for each n, $\tilde{Q}_n(p)$ is increasing on $[0,1]$, $\tilde{Q}_n$ is differentiable almost everywhere. And since $\tilde{Q}_n'(p) < \infty$ almost everywhere on $[0,1]$ for each n, there exists M such that $\sup_p \mid \tilde{Q}_n'(p) \mid < M \ \forall n$. Then we have, $\forall \epsilon > 0$,

$$P(\mid \tilde{Q}_n(p + \tilde{F}_n(0)) - \tilde{Q}_n(p) \mid > \epsilon) = P(\mid \tilde{Q}_n(c) \mid\mid \tilde{F}_n(0) \mid > \epsilon) \tag{3.27}$$

$$\leq P(M \mid \tilde{F}_n(0) \mid > \epsilon) \tag{3.28}$$

$$= P(\mid \tilde{F}_n(0) \mid > \epsilon/M) \tag{3.29}$$

$$\xrightarrow{p} 0 \qquad (Corollary 1) \tag{3.30}$$

By (3.30), (3.22) becomes

$$\sup_{0 \leq p \leq F(b)-\epsilon} \mid \sqrt{n}(\tilde{Q}_n(p) - Q(p)) + \sqrt{n}\frac{\tilde{F}_n^0(Q(p)) - p}{f(Q(p))} \mid \xrightarrow{p} 0. \tag{3.31}$$

Theorem 2 now follows from Theorem 1.

CHAPTER 4

APPLICATION OF SMOOTH QUANTILE PROCESSES

4.1 Efron's Bootstrap

A re-sampling method known as Efron's bootstrap (or simply the bootstrap) was introduced by Brad Efron. It is a computer-intensive method to approximate the sampling distribution of any statistic of interest. Bootstrap samples are samples of size n drawn at random from the original data set of size n with replacement.

4.2 Simultaneous Confidence Bnads

Let $(\tilde{X}_i^*, \delta_i^*), i = 1, 2, ..., n$ be bootstrap replicates of the original right censored data $(\tilde{X}_i, \delta_i), i = 1, 2, ..., n$ Let $\hat{F}_n^*(x)$ be the Kaplan-Meier estimator based on a bootstrap sample and $\tilde{F}_n^*(x)$ the bootstrap smoothed Kaplan-Meier estimator of $F(x)$. And let $\tilde{Q}_n^*(p)$ be the bootstrap smoothed estimator of the quantile function $Q(p)$. Here we show analogous results for bootstrap samples.

Theorem 3 *Under* $C1 - C4$, *for* $0 \leq a < b < \tau_F$, $\sqrt{n}(\tilde{F}_n^*(x) - \tilde{F}_n(x)) \xrightarrow{D} Z(x)$ *in* $D[a, b]$ *as* $n \to \infty$.

proof :

$$\sqrt{n}(\tilde{F}_n^*(x) - \tilde{F}_n(x)) = \sqrt{n}\left(\int_0^\infty h^{-1}\hat{F}_n^*(t)k((x-t)/h)dt - \int_0^\infty h^{-1}\hat{F}_n(t)k((x-t)/h)dt \right) \tag{4.1}$$

$$= \sqrt{n}h^{-1}\int_0^\infty (\hat{F}_n^*(t) - \hat{F}_n(t))k((x-t)/h)dt \tag{4.2}$$

$$= h^{-1}\int_0^b \sqrt{n}(\hat{F}_n^*(t) - \hat{F}_n(t))k((x-t)/h)dt + \sqrt{n}h^{-1}\int_b^\infty (\hat{F}_n^*(t) - \hat{F}_n(t))k((x-t)/h)dt. \tag{4.3}$$

The second term of equation (4.3) can be shown to be equal to 0 as follows. By condition $C1$,

$$\sqrt{n}h^{-1}\int_b^\infty \mid \hat{F}_n^*(t) - \hat{F}(t) \mid k((x-t)/h)dt \leq 2\sqrt{n}h^{-1}\int_b^\infty k((x-t)/h)dt \tag{4.4}$$

$$= 2\sqrt{n}\int_{-\infty}^{\frac{x-b}{h}} k(u)du \tag{4.5}$$

$$\leq 2\sqrt{n}\int_{-\infty}^{-c} k(u)du \tag{4.6}$$

$$= 0 \tag{4.7}$$

Now

$$h^{-1}\int_0^b \sqrt{n}(\hat{F}_n^*(t) - \hat{F}_n(t))k((x-t)/h)dt = \int_{\frac{x-b}{h}}^{\frac{x}{h}} \sqrt{n}(\hat{F}_n^*(x-hu) - \hat{F}_n(x-hu))k(u)du$$

$$= \int_{\frac{x-b}{h}}^{\frac{x}{h}} \sqrt{n}(\hat{F}_n^*(x) - \hat{F}_n(x))k(u)du + \int_{\frac{x-b}{h}}^{\frac{x}{h}} \sqrt{n}\Big[(\hat{F}_n^*(x-hu) - \hat{F}_n(x-hu)) - (\hat{F}_n^*(x) - \hat{F}_n(x))\Big]k(u)du \tag{4.8}$$

By theorem 2.1 of Akritas [1], $\sqrt{n}(\hat{F}_n^*(x) - \hat{F}_n(x)) \xrightarrow{D} Z(x)$ in $D[a,b]$. Then we see that

$$\lim_{n\to\infty}\lim_{\delta\to 0}\sup_{|x-y|<\delta}\left|\sqrt{n}(\hat{F}_n^*(x-hu) - \hat{F}_n(x-hu)) - \sqrt{n}(\hat{F}_n^*(x) - \hat{F}_n(x))\right| \xrightarrow{p} 0 \quad (4.9)$$

This implies that the term $(4.8) = o_p(1)$. Also we have

$$\int_{\frac{x-b}{h}}^{\frac{x}{h}} \sqrt{n}(\hat{F}_n^*(x) - \hat{F}_n(x))k(u)du = \sqrt{n}(\hat{F}_n^*(x) - \hat{F}_n(x)) \int_{\frac{x-b}{h}}^{\frac{x}{h}} k(u)du \quad (4.10)$$

$$\xrightarrow{D} Z(x) \quad (4.11)$$

This completes the proof of theorem 3.

Theorem 4 *Let $0 < \beta < 1$. Suppose that $f(Q(p))$ is continuous and positive on the interval $[0,\beta]$. Then, under $C1 - C4$, $\sqrt{n}(\tilde{Q}_n^*(p) - \tilde{Q}_n(p)) \xrightarrow{D} Z(Q(p))/f(Q(p))$ in $D[0,\beta]$ as $n \to \infty$.*

proof : The proof of theorem 4 is done in a similar manner to the proof of the main theorem, by using theorem 3 with theorem 2 of Doss and Gill (1991).

Now, applying theorem 4, a $(1-\alpha)100\%$ simultaneous confidence band for quantile function $Q(p)$ over an interval $I \subset [0,1)$ is given by

$$(\tilde{Q}_n(p) - c/\sqrt{n}, \tilde{Q}_n(p) + c/\sqrt{n}) \quad (4.12)$$

where c is a value such that

$$P(\sqrt{n} sup_{p\in I} \mid \tilde{Q}_n^*(p) - \tilde{Q}_n(p) \mid \leq c \mid \{\tilde{X}_i, \delta_i\}_1^n) \approx 1 - \alpha \quad (4.13)$$

4.3 Hypothesis Testing

Next, we construct a simultaneous confidence band for the difference between two quantile functions to test whether two distributions F_1 and F_2 belong to the same distribution family $\mathcal{F} = \{F(x-\theta) \mid \theta \in \Theta\}$, where F is an unknown continuous distribution. First, we consider the following null-hypothesis.

$H_0^{(1)}$: F_1 and F_2 are members of a distribution family $\mathcal{F}$, *i.e.* there exist θ_1 and $\theta_2 \in \Theta$ such that $F_1 = F(x-\theta_1)$ and $F_2 = F(x-\theta_2)$.

Let $Q_1(p)$ and $Q_2(p)$ be the corresponding quantile functions to F_1 and $F_2 \in \mathcal{F}$, respectively. Then note that $Q_1(p) = \theta_1 + Q(p)$ and $Q_2(p) = \theta_2 + Q(p)$. Thus, we have the following equivalent hypothesis to $H_0^{(1)}$.

$H_0 : Q_1(p) - Q_2(p) = \theta_1 - \theta_2 = \theta \ \forall \ 0 < p < 1$, where θ is some constant in Θ.

In other words, if H_0 is true, then it is equivalent to show F_1 and F_2 are from the same distribution family. We develop a method to establish a confidence band such that H_0 (equivalently $H_0^{(1)}$) is not rejected if θ is within the confidence band for any $p \in (0,1)$.

Let $\{\tilde{X}_i, \delta_i\}_1^n$ and $\{\tilde{Y}_i, \gamma_i\}_1^m$ be samples of n and m right censored data. Let $\tilde{Q}_{1,n}(p)$ and $\tilde{Q}_{2,m}(p)$ be the K-M smooth quantile estimates from the first and the second samples, respectively. and let $\tilde{Q}^*_{1,n}(p)$ and $\tilde{Q}^*_{2,m}(p)$ be the bootstrap estimates. Suppose $n/m \to \rho$ as $n, m \to \infty$. Then by theorem 4, the distribution of $\sqrt{n}(\tilde{Q}_{1,n}(p) - Q_1(p)) - \sqrt{n/m}\sqrt{m}(\tilde{Q}_{2,m}(p) - Q_2(p))$ can be estimated by the distribution of $\sqrt{n}(\tilde{Q}^*_{1,n}(p) - \tilde{Q}_{1,n}(p)) - \sqrt{n/m}\sqrt{m}(\tilde{Q}^*_{2,m}(p) - \tilde{Q}_{2,m}(p))$ conditional on the two data sets $\{\tilde{X}_i, \delta_i\}_1^n$ and $\{\tilde{Y}_i, \gamma_i\}_1^m$. Therefore, a 90 % simultaneous confidence band for $Q_1(p) - Q_2(p)$ over an interval $I \subset [0,1)$ is given by

$$(\tilde{Q}_{1,n}(p) - \tilde{Q}_{2,m}(p) \pm d/\sqrt{n}), \tag{4.14}$$

where d is a value such that

$$P(\sqrt{n}sup_{p\in I} \mid (\tilde{Q}^*_{1,n}(p)-\tilde{Q}^*_{2,m}(p))-(\tilde{Q}_{1,n}(p)-\tilde{Q}_{2,m}(p)) \mid \leq d \mid \{\tilde{X}_i,\delta_i\}_1^n,\{\tilde{Y}_i,\gamma_i\}_1^m) \approx .9. \tag{4.15}$$

4.4 Example

In this section, we provide an example of testing the hypothesis H_0 in section 4.3. The data are from a randomized CTE brain tumor clinical trial [13]. This is a trial of BCNU impregnated implantable polymer for the treatment of recurrent malignant tumor in the brain. 222 patients were randomized with equal probability to receive either BCNU polymer or placebo polymer, implanted in the cavity remaining after surgical resection of recurrent tumors. We test the hypothesis that the group receiving BCNU and the group receiving the placebo belong to the same location distribution family. We construct a 90% confidence band for the difference of two quantile functions using the formulas (19) and (20). The confidence band in figure 4.1 shows the existence of a constant θ in H_0 within the band. Thus we conclude that these two samples are drawn from the same location family.

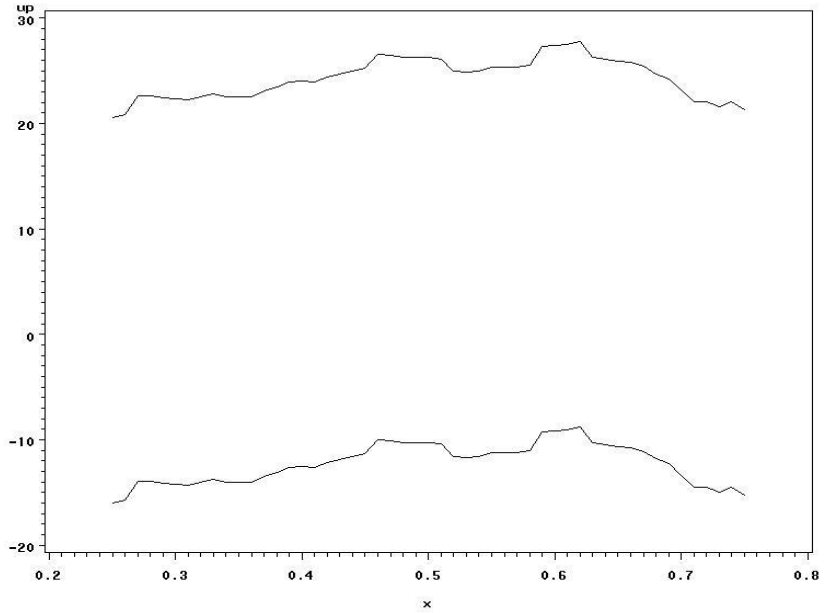

Figure 4.1. Confidence band for the difference between the two quantile functions.

CHAPTER 5

BANDWIDTH CONSIDERARION

Determining an appropriate value for the bandwidth h plays an important role in constructing confidence bands since coverage probabilities and lengths of bands depend on h as well as n. A bandwidth selection method for a point estimator of $Q(p)$ to choose the best value of h for computing the quantile estimator $\tilde{Q}_n(p)$ has been proposed by Padgett and Thombs [11]. They choose h^* to be the bandwidth if it minimizes the bootstrap estimate of the mean squared error, $MSE^*(\tilde{Q}_n, h)$, which is defined as follows.

First, let $\tilde{Q}_n^{*i}(p)$ denote the quantile estimate obtained from the ith bootstrap sample, $i = 1, 2, ..., B$. Then the bootstrap estimate of variance is defined by

$$Var^*(\tilde{Q}_n(p)) = \frac{1}{B-1}\sum_{i=1}^{B}[\tilde{Q}_n^{*i}(p) - \overline{\tilde{Q}_n^*}(p)]^2, \tag{5.1}$$

where $\overline{\tilde{Q}_n^*}(p) = \frac{1}{B}\sum_{i=1}^{B}\tilde{Q}_n^{*i}(p)$. The bootstrap estimate of bias is

$$Bias^*(\tilde{Q}_n(p)) = \overline{\tilde{Q}_n^*}(p) - \hat{Q}_n(p), \tag{5.2}$$

where $\hat{Q}_n(p)$ is the Kaplan Meier estimate obtained from the original data. So for some fixed p and h, the bootstrap estimate of the mean squared error is given by

$$MSE^*(\tilde{Q}_n(p), h) = Var^*(\tilde{Q}_n(p)) + [Bias^*(\tilde{Q}_n(p))]^2. \tag{5.3}$$

Now, to construct confidence bands over an interval $p \in I$, we need to select a bandwidth to minimize a "global" mean squared error, so called the mean integrated squared error (MISE). The bootstrap estimate of MISE is given by

$$\begin{aligned} MISE^*(\tilde{Q}_n(p), h) &= \int_{p \in I} MSE^*(\tilde{Q}_n(p), h) w(p) dp \\ &\approx \sum_{j=1}^{J} MSE^*(\tilde{Q}_n(p_j), h) w(p_j) \ \triangle_j, \end{aligned}$$

where $w(p)$ is a weight function. $p_1 < p_2 < ... < p_J$ is a partition of the interval I and $\triangle_J = p_j - p_{j-1}$. The selection of h^* is the value minimizing $MISE_*(\tilde{Q}_n(p), h)$. Once h^* is selected, a simultaneous confidence band for $Q(p)$, $p \in I$ can be constructed based on $\tilde{Q}_n^{*i}(p)$ which are obtained using h^*.

CHAPTER 6

SIMULATION STUDIES

6.1 Performance of the Confidence Bands

In this chapter, we carry out a Monte Carlo simulations to assess the performance of the proposed confidence bands using the smoothed quantile estimates. We compare the coverage probabilities and lengths of the confidence bands computed from the smoothed and non-smoothed quantile estimates.

First, we construct the original right censored sample of size n. The survival times are generated from the exponential distribution with mean 1,

$$F(x) = \begin{cases} 0 & \text{if x} \leq 0 \\ 1 - e^{-x} & \text{if x} > 0. \end{cases} \tag{6.1}$$

The censoring times are generated from the exponential distribution with mean 7/3,

$$G(x) = \begin{cases} 0 & \text{if x} \leq 0 \\ 1 - e^{-3x/7} & \text{if x} > 0. \end{cases} \tag{6.2}$$

The kernel density function used here is called the *Epanechnikov* kernel and defined by

$$K(u) = \frac{3}{4}(1 - u^2)1_{\{|u|<1\}}. \tag{6.3}$$

From the original sample of size n, we generate 1000 bootstrap samples. The Kaplan-Meier estimator, $\hat{F}_n(x)$ and bootstrap K-M estimators, $\hat{F}^*_{n,j}(x), j = 1, .., 1000$ are calculated based on the original right censored sample and bootstrap samples, respec-

tively. The smooth K-M estimator, $\tilde{F}_n(x)$, and smooth bootstrap K-M estimators, $\tilde{F}^*_{nj}(x), j = 1, ..., 1000$, can be calculated by using the formulae (2.2) in Padgett and Thombs [11]. The smooth estimator $\tilde{Q}_n(p)$ of the quantile function is the solution of $\tilde{F}(\tilde{Q}_n(p)) = p$ and all of the smooth quantile estimates for the original and 1000 bootstrap samples were found by applying the Newton method. Once all the smooth quantile estimates are computed, we find the value of c_j, defined by

$$c_j = \sqrt{n} \sup_{p \in I} \mid \tilde{Q}^*_{(n,j)}(p) - \tilde{Q}_n(p) \mid$$

$\forall j = 1, .., 1000$, where , in our simulations, $I = .25(.01).75$ and then construct an ascending ordered set, $\{c_{(1)}, c_{(2)}, ..., c_{(1000)}\}$. To construct a 90 % confidence band for $Q(x)$ over $[.25, .75]$, we use $\tilde{c} = c_{(900)}$ as the estimate of c in (4.13) .

To assess the performance of the bootstrapping method, we carry out 1000 simulations and calculate the coverage probabilities. The coverage probability for all 1000 simulations of size n and a specific value of bandwidth h can be calculated by calculating the relative frequency of all the one thousand 90 % confidence bands defined in (4.12) containing $Q(p)$ for $p \in I$. The average length of the confidence bands for a given pair of n and h is determined as the mean value of $2\tilde{c}/\sqrt{n}$ for all simulations.

The coverage probabilities and average lengths of the 90 % simultaneous confidence bands for the quantile function $Q(p) \equiv F^{-1}(p) = -log(1-p)$ over $p \in I$ for sample sizes $n = 50$, 100, 150, 200 and 300 are reported in tables 6.1 and 6.2 below.

For tables 6.1 and 6.2, their first columns show the coverage probabilities and average lengths of the confidence bands for non-smoothed method. For the smoothed method, the results are shown for the bandwidths $h = .10(.10).80$. We see from table 6.1 that, for a given n, the coverage probabilities of the smoothed method are closer

to the nominal level of .90 than those of non-smoothed method for all bandwidths. As h increases, the coverage probabilities are generally decreasing. Similarly, it is seen from table 6.2 that for any n, the average lengths of bands are shorter than those of non-smoothed method. And the average lengths decrease as the sample size n or bandwidth h increases.

Fighre 6.1 is a demonstration of the simultaneous confidence bands of the quantile function over $p \in [.25, .75]$ using the smooth and non-smooth estimates with $n = 300$ and $h = .36$. The solid line is the true quantile function. The dotted lines are the smooth quantile estimate and the confidence band. And stepped lines are the non-smooth estimate and the confidence band.

Table 6.1. Coverage probabilities of 90 % simultaneous confidence bands for $Q(p)$, $0.25 \leq p \leq 0.75$
with 1000 replicates, using smoothing and non-smoothing bootstrap methods

$n\backslash h$	non-smooth	smooth							
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
50	.947	.936	.919	.907	.897	.891	.888	.896	.890
100	.942	.935	.925	.915	.906	.898	.896	.893	.891
150	.947	.937	.924	.918	.910	.905	.908	.903	.903
200	.941	.937	.926	.914	.911	.905	.902	.895	.881
300	.942	.935	.922	.914	.907	.894	.894	.878	.863

6.2 Optimum Bandwidth Selection

We simulated the data from the same exponential distributions as in section 6.1. For simplicity, the weight function for the computation of $MSE^*(\tilde{Q}_n(p), h)$ is chosen to be $w(p_j) = 1$ over the partition $p_j = .25(.01).75$. The values of $MISE^*(\tilde{Q}_n(p), h)$ was computed by the formula in Chapter 5 for the bandwifth $h = 0(.01)1$ with 1000

Table 6.2. Average lengths of 90 % simultaneous confidence bands for $Q(p)$, $0.25 \leq p \leq 0.75$
with 1000 replicates, using smoothing and non-smoothing bootstrap methods

$n\backslash h$	non-smooth	smooth							
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
50	1.4393	1.3708	1.2893	1.2120	1.1480	1.0863	1.0321	0.9842	0.9423
100	0.9295	0.8763	0.8274	0.7756	0.7308	0.6939	0.6641	0.6396	0.6191
150	0.7359	0.6942	0.6499	0.6090	0.5754	0.5490	0.5283	0.5109	0.4958
200	0.6176	0.5854	0.5445	0.5100	0.4834	0.4637	0.4480	0.4350	0.4235
300	0.4910	0.4638	0.4315	0.4056	0.3868	0.3728	0.3616	0.3520	0.3432

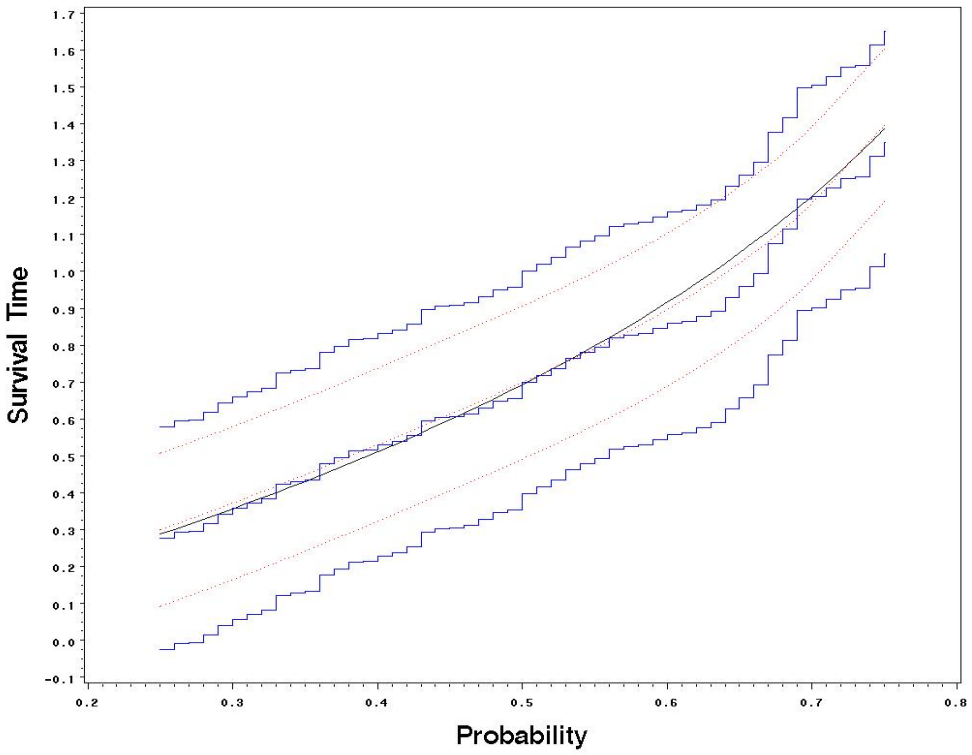

Figure 6.1. Quantile estimates and 90 % simultaneous confidence bands for exponential survival times .

bootstrap samples. The optimum bandwidth h^* which minimizes $MISE^*(\tilde{Q}_n(p), h)$ is given for each of the samples of size $n = 50, 100, 150, 200, 300, 500$ and 1000 in table 6.3. We find that as the size n of the sample increases, h^* decreases as well as $MISE^*(\tilde{Q}_n(p), h)$.

Table 6.3. Bootstrap selections of smooth bandwidth h^* minimizing $MISE^*(\tilde{Q}_n(p), h)$ for
$0.25 \leq p \leq 0.75$ for a single right censored sample.

n	50	100	150	200	300	500	1000
h^*	.88	.59	.60	.45	.36	.36	.25
$MISE^*(\tilde{Q}_n(p), h^*)$	.020914	.006516	.005074	.003402	.002623	.001814	.000859

APPENDIX A

SAS Code

In this chapter, the SAS codes, which were used in the simulation studies, are presented.

Coverage probabilities and average lengths of confidence bands

```
libname saslib ' /sasoutputs/' ;
% let nsam=1; ** this must be equal to 1;
% let nnsam=1001; ** number of samples to generate - must be at least 2.
% let nnnsam= & nnsam-1; ** number of bootstrap samples, which is 1000;
% let nsize=200; ** sample size n;
% let seed1=5129589; ** seed for initial sample;
% let seed2=7200117; ** seed for bootstrap samples;
% let ntri=1000; ** number of trials;
% let numqt=51; ** number of quantiles, i.e., .25(.01).75;
% let h=.2; **bandwidth h;

**********************************************************************
This macro generates a sample of size n from each of
two exponential distributions.
One contains true life-times and the other contains censoring times.
**********************************************************************;
%macro picksampone;
** generate nsam samples of size nsize from F(x)and nsam samples of size nsize
from G(c);
call streaminit(&seed1);
data one;
isam=0; ** sample index;
do trial=1 to &ntri;
```

```
    do i=1 to &nsize; ** observation index within the sample;
       x=rand('EXPONENTIAL'); ** life times;
       cen=rand('WEIBULL',1,7/3); ** censoring times;
       x _ tilda=min(x,cen); ** right censored data;
       delta=(x <= cen); ** delta=1 if life time is observed, 0 if censored;
       output;
    end;
end;

proc sort data=one;
      by trial x _ tilda;

% mend picksampone;


% macro samps;
data keepone1;
      set one;
         isam=1;
         keep trial isam x _ tilda delta;
            % do iisam=2 % to & nnnsam; ** Duplicating the original sample;
               data keepone & iisam; ** Generating many sets in the do loop;
               set keepone1;
               isam= &iisam;
         % end;
```

```
data keepall;
    set
      % do iisam=1 % to & nnnsam;
        keepone & iisam
      % end;
      ;

proc sort data=keepall;
    by trial isam x _ tilda;


%mend samps;


*****************************************************************
This macro generates bootstrap samples from the original sample.
*****************************************************************;
%macro pickallsamps;
data one;
    set one;
     %samps
proc surveyselect data=keepall
    method = urs
    sampsize = & nsize
    seed= & seed2
    out=bstrap1;
    strata trial isam ;
    run;
```

```
data bstrap;
    set bstrap1;
        do i = 1 to numberhits;
            output; ** outputs each datum numberhits times;
        end;

data keepone;
    set one bstrap;
        keep trial isam x _ tilda delta numberhits;

proc sort data=keepone;
    by trial isam x _ tilda;
%mend pickallsamps;

*************************************************************
This macro computes Non-smoothed K-M estimates of F.
*************************************************************;
%macro f_hat;
proc transpose data=keepone out=xout prefix=xo;
    var x _ tilda;
    by trial isam;


proc transpose data=keepone out=dout prefix=do;
    var delta;
    by trial isam;
```

```
proc transpose data=keepone out=sout prefix=so;
      var x_ tilda;
      by trial isam;

data transdata;
      merge xout dout sout;
      by trial isam;

data fvalue;
      set transdata;
         array xo(*) xo1-xo &nsize;
         array do(*) do1-do &nsize;
         array so(*) so1-so &nsize;
            do j=1 to &nsize;
            if(so(j) le xo1) then do;
                                    f1=1;
                                end;
            else do;
               do k=2 to &nsize;
                  if(xo(k-1) lt so(j) and so(j) le xo(k)) then do;
                                                         p=1;
                                                         do i=1 to k-1;
                                                            p=p*((&nsize - i)/(&nsize -
i+1))**do(i);
                                                         end;
                                                         f1=p;
```

```
                                                    end;
        end;
    end;
        if(xo(&nsize) < so(j)) then f1=0;
    f _ hat=1-f1;
    x=so(j);
    delta=do(j);

        output;
    end;
     keep trial isam f _ hat x delta;

data work.stuff;
    set fvalue;
        by trial isam x;
        if first.x;
run;
%mendf_hat;

*****************************************************************************
This macro finds K-M estimates of the quantiles.
And these quantile estimates are used as the initial values of the newton method to
compute
the smoothed K-M estimates.
*****************************************************************************;
```

```
%macro km_quant;
proc transpose data=work.stuff out=fout prefix=fo;
      var f _ hat;
      by trial isam;

proc transpose data=work.stuff out=gout prefix=go;
      var x;
      by trial isam;

proc transpose data=work.stuff out=delout prefix=del;
      var delta;
      by trial isam;

data step1;
      merge fout;
      by trial isam;

data step2;
      set step1;
         array fo(*) fo1-fo & nsize;
         do i=2 to & nsize;
            f _ km=fo(i);
            output;
         end;
        keep trial isam f _ km;

data step3;
      do i=1 to & ntri;
```

```
        do j=1 to & nnsam;
            trial=i;
            isam=j-1;
            f _ km=1;
            output;
        end;
    end;
     keep trial isam f _ km;

data step4;
    set step2 step3;
    if f _ km ne .;

proc sort data=step4;
    by trial isam f _ km;

proc transpose data=step4 out=kmout prefix=kmo;
    var f _ km;
    by trial isam;


data f _ value;
    merge fout kmout gout delout;
    by trial isam;

data step5;
    set f _ value;
        array go(*) go1-go &nsize;
```

```
        array del(*) del1-del &nsize;
        array fo(*) fo1-fo &nsize;
        array kmo(*) kmo1-kmo &nsize;
            do i=1 to &nsize;
            x=go(i);
            if kmo(i)=. then delete;
            else do;
                    f _ km=kmo(i);
                end;

              if fo(i)=. then delete;
            else do;
                    f _ km2=fo(i);
                end;
              jump=kmo(i)-fo(i); ** Jump size of F _ hat at Xi _ tilde;
            delta=del(i);
            output;
    end;
      keep trial isam f _ km f _ km2 x jump delta ;

data work.stuff2;
    set step5;
        by trial isam f _ km;
        if first.f _ km;
run;
```

```
proc transpose data=work.stuff2 out=kout prefix=ko;
    var f _ km;
    by trial isam;

proc transpose data=work.stuff2 out=zout prefix=zo;
    var x;
    by trial isam;

data step6;
    merge kout zout;
    by trial isam;

data quantile;
    set step6;
      array ko(*) ko1-ko&nsize;
      array zo(*) zo1-zo&nsize;
            do p=.25 to .75 by .01;
        fm=100000;
          do l=1 to &nsize;
            if (ko(l) ge p) then do;
                              if (ko(l) le fm) then do;
                                                fm=ko(l);
                                                qt_est=zo(l);
                                                f_hat=ko(l);
                                                prob=p;
                                               end;
                              end;
```

```
            end;
            output;
          end;
      keep trial isam prob f_hat qt_est;
proc datasets library=work;
    save stuff2 quantile zout;
run;
%mend km_quant;

*************************************************************************
The derivative of F _ tilda, which is computed by the formula in a paper
by Padgett and Thombs (1988)
*************************************************************************;
%macro derivative;
      t=x;
derivative=0;
    do i=1 to &nsize;
       y=abs(t-zo(i));
       if y lt &h then do;
                        ker=(1/&h)*jo(i)*(3/4)*(1-((t-zo(i))/&h)**2);
                    end;
                    else do;
                          ker=0;
                        end;
       if ker=. then do;
```

```
                derivative = derivative;
            end;
        else do;
            derivative=derivative+ker;
        end;
    end;

%mend;

*************************************************************
Smooth K-M estimator
*************************************************************;
%macro f_tilde;
    t=x;
f_tilde=0;
    do j=1 to &nsize;
        if t le zo(j)-&h then do; sjw=0; end;
        if zo(j)-&h lt t lt zo(j)+&h then do;
                sjw=jo(j)*((-1/4)*((t-zo(j))/&h)**3+(3/4)*((t-
zo(j))/&h)+.5);
            end;
        if zo(j)+&h le t then do; sjw=jo(j); end;
        if sjw=. then do; f_tilde=f_tilde; end;
        else do; f_tilde=f_tilde+sjw; end;
    end;
```

```
%mend;
%macro f_tilde2;
      t=x;
f_tilde2=0;
          do j=1 to &nsize;
       if t le zo(j)-&h then do; sjw=0; end;
       if zo(j)-&h lt t lt zo(j)+&h then do;
                                        sjw=jo(j)*((-1/4)*((t-zo(j))/&h)**3+(3/4)*((t-
zo(j))/&h)+.5);
                                    end;
       if zo(j)+&h le t then do; sjw=jo(j); end;
       if sjw=. then do; f_tilde2=f_tilde2; end;
       else do; f_tilde2=f_tilde2+sjw; end;
    end;
%mend;

*************************************************************************
Computing smooth quantle estimates
*************************************************************************;
%macro smoothqtestimate;
proc transpose data=work.stuff2 out=jout prefix=jo;
    var jump;
    by trial isam;
```

```
proc transpose data=quantile out=prout prefix=pro;
    var prob;
    by trial isam;
proc transpose data=quantile out=qout prefix=qo;
    var qt _ est;
    by trial isam;
data step7;
    merge jout zout prout qout;
data smoothqt;
    set step7;
      array jo(*) jo1-jo & nsize;
    array zo(*) zo1-zo & nsize;
    array pro(*) pro1-pro & numqt;
    array qo(*) qo1-qo & numqt;
        do mm=1 to & numqt;
       x=qo(mm);
       p=pro(mm);
        do jjj=1 to 20;
          %derivative
          %f_tilde
          d=(f _ tilde-p)/derivative;
          x=x-d;
          %f_tilde2;
            if abs(d) < .000000000000001 then do; jjj=20; end;
```

```
            end;
        output;
    end;
     keep trial isam d x jjj p f _ tilde2;
% mend;

*************************************************************************************
This macro computes the suprimum difference between original quantile estimates
and
bootstrap quantile estimates.
*************************************************************************************;
%macro qtdifference;
data bestquant;
    set smoothqt;
data originalquant;
    set bestquant;
      if isam = 0;
      rename x = quantile;
data bootstrapquant;
    set bestquant;
      if isam = 0 then delete;
          % do iisam=1 % to &nnnsam;
            data originalquant &iisam; ** Generating many sets in the do loop;
            set originalquant;
```

```
            isam=&iisam;
        % end;
data duplicatingoriginal;
    set
        % do iisam=1 % to &nnnsam;
            originalquant & iisam /*no semicolon here*/
        % end;
; ** this semicolon is for set;
proc sort data=duplicatingoriginal;
    by trial;
proc transpose data=duplicatingoriginal out=pout prefix=po;
    var quantile;
    by trial isam;
proc transpose data=duplicatingoriginal out=ftilout prefix=ftilo;
    var f _ tilde2;
    by trial isam;
proc transpose data=bootstrapquant out=ppout prefix=ppo;
    var x;
    by trial isam;
proc transpose data=bootstrapquant out=fftilout prefix=fftilo;
    var f _ tilde2;
    by trial isam;
data supremum;
    merge pout ppout ftilout fftilout;
        array po(*) po1-po &numqt;
    array ftilo(*) ftilo1-ftilo &numqt;
```

```
    array ppo(*) ppo1-ppo &numqt;
    array fftilo(*) fftilo1-fftilo &numqt;
p=.20;
      do i=1 to &numqt by 1;
      p=p+.05;
      originalqt=po(i);
      bootqt=ppo(i);
      quantdiff=abs(ppo(i)-po(i));
      output;
    end;
     keep trial i isam p quantdiff originalqt bootqt;
data supremum2;
    set supremum;
proc sort data=supremum2;
    by trial isam quantdiff; ** by sorting the set, it is easier to spot the largest
quantdiffs;
run;
proc transpose data=supremum2 out=ddout prefix=qdif;
    var quantdiff;
    by trial isam;
data findingc;
    merge ddout;
      array qdif(*) qdif1-qdif &numqt;
      do i= &numqt;
      sup_q_diff=qdif(i);
```

```
        output;
    end;
keep trial isam sup _ q_ diff;
%mend;


%picksampone
%pickallsamps
%f_hat
%km_quant
%smoothqtestimate
%qtdifference


** Outputs quantile estimates from the initial samples;
saslib.originalqt;
      set originalquant;
** Outputs all the suprimum quantile differences between the initial sample and each
of the bootstrap sample;
data saslib.qtdiff;
    set findingc;
run;


% let p _ cent=.9; * Pth percentile;
% let ntri=1000; ** number of trials;
% let nsam=1000; ** number of bootstrap samples for each trial;
```

```
% let nsize=200; ** size of each sample;
% let numqt=51; ** number of quantiles;


%macro findC;
data one;
     set saslib.qtdiff;
proc sort data=one;
     by trial sup _ q _ diff;
proc transpose data=one out=qout prefix=qo;
     var sup _ q _ diff;
     by trial;
data two;
     set one nobs=setsize;
     merge qout;
     by trial;
data three;
     set two;
         array qo(*) qo1-qo &nsam;
         PxNSAM= & p _ cent* &nsam;
     int=int(PxNSAM);
     dec=PxNSAM-int;
         do i=1 to &ntri;
        if dec = 0 then c = ( &nsize ** .5) * (qo(int)+ qo(int+1))/2;
     else do;
             c = (& nsize ** .5) * qo(int+1);
```

```
        end;
    end;
data four;
    set three;
    keep trial c;
%mend findC;

************************************************
This macro finds the average of C's
************************************************;
%macro findaverageC;
proc transpose data=three out=cout prefix=co;
    var c;
data five;
    merge cout;
        array co(*) co1-co &ntri;
        average _ of _ C = mean(of co1-co &ntri);
    keep average _ of _ c;
%mend findaverageC;

%findC
%findaverageC;
```

```
** Outputs all C's from 1000 trials;
data saslib.cvalue;
      set four;
** Outputs the average value of the 1000 C's;
data saslib.aveCn;
      set five;
run;


%macro constructingsets;
data one;
      set saslib.originalqt;
      keep trial c f _ tilda;
proc sort data=one;
      by trial f _ tilda;
run;
data truequant;
      do p=.25 to .75 by .05;
         true _ quant=quantile('EXPO',p);
         trial=1;
         output;
      end;
          % do itri=1 % to &ntri;
            data truequant &itri;
                 set truequant;
```

```
                trial= & itri;
        % end;
data keepall;
      set
         % do itri=1 % to &ntri;
           truequant &itri
         % end;
;
proc sort data=keepall;
      by trial true_quant;
%mend constructingsets;


************************************************************
Finding the supremum difference between the estimated
quantiles and true quantiles.
************************************************************;
%macro supdiff;
proc transpose data=one out=xout prefix=xo;
      var c;
      by trial;
proc transpose data=keepall out=dout prefix=do;
      var true_quant;
by trial;
```

```
data supdiff;
      merge xout dout;
      by trial;
          array xo(*) xo1-xo &numqt;
      array do(*) do1-do &numqt;
          do i=1 to &numqt;
          diff=( &nsize ** .5) * abs(xo(i)-do(i));
          output;
      end;
        keep diff trial;
proc sort data=supdiff;
      by trial diff;
proc transpose data=supdiff out=supout prefix=sup;
      var diff;
      by trial;
data choosesup;
      merge supout;
      by trial;
      array sup(*) sup1-sup &numqt;
      keep trial sup &numqt;
%mend supdiff;

**********************************************************************
Computing the coverage probability of P % simultaneous confidence
bands for Q(p), .25 <= p <= .75
```

```
*****************************************************************;
%macro coverageprob;
data two;
     set saslib.cvalue;
proc transpose data=choosesup out=tout prefix=to;
     var sup & numqt;
     by trial;
proc transpose data=two out=ccout prefix=cco;
     var c;
     by trial;
data three;
     merge tout ccout;
     by trial;
        array to(*) to1;
     array cco(*) cco1;
        do i=1;
       if to(i) le cco(i) then cover=1;
     else cover=0;
     end;
      keep trial cover;
proc transpose data=three out=cpout prefix=cp;
     var cover;
data four;
     merge cpout;
     coverprob= sum(of cp1-cp &ntri)/ &ntri;
     keep coverprob;
```

%*mendcoverageprob*;

%*constructingsets*;

%*supdiff*;

%*coverageprob*;

** Outputs the coverage probability;

data saslib.covprob;

    set four;

run;

Optimum bandwidth

For the computations of optimal bandwidth, we use the same macros *picksampone*, *pickallsamps*, *f_hat*, *km_quant* and *smoothqtestimate* as defined in section **??** along with the new macros shown below.

```
%macro variance_n_bias;
data group_by_p;
    set smoothqt;
      if isam ne 0;

proc sort data=group_by_p;
    by bw p;
proc transpose data=group_by_p out=k1uout prefix=k1u;
    var x;
    by bw p;

data btstrap_qt_mean;
    merge k1uout;
      array k1u(*) k1u1-k1u&numbt;
      qt_mean=sum(of k1u1-k1u&numbt)/ &numbt;
      output;
keep bw p qt_mean;

data duplicate_qt_mean;
    set btstrap_qt_mean;
        do e1u=1 to &numbt;
          output;
```

```
        end;

proc transpose data=duplicate_qt_mean out=k2uout prefix=k2u;
    var qt_mean;
    by bw p;

data squarediff;
    merge k1uout k2uout;
        array k1u(*) k1u1-k1u&numbt;
    array k2u(*) k2u1-k2u&numbt;
        do e2u=1 to &numbt;
        bt_qt=k1u(e2u);
        qt_mean=k2u(e2u);
        sq_diff=(k1u(e2u)-k2u(e2u))**2;
        output;
     end;
keep bw p bt_qt qt_mean sq_diff;

proc transpose data=squarediff out=k3uout prefix=k3u;
    var sq_diff;
    by bw p;

data variance;
    merge k3uout;
        array k3u(*) k3u1-k3u&numbt;
```

```
        bt_var=sum(of k3u1-k3u&numbt)/ (&numbt-1);
    output;
keep bw p bt_var;

data original_km_qt;
    set quantile;
      if isam=0;
      rename prob=p;

data duplicate_KM_qt;
    set original_km_qt;
        do h=.7 to .9 by .01; ****************************;
       bw=h;
       output;
      end;
keep bw p qt_est;

proc sort data=duplicate_KM_qt;
    by bw p;

proc transpose data=duplicate_KM_qt out=k4uout prefix=k4u;
    var qt_est;
    by bw p;

proc transpose data=btstrap_qt_mean out=k5uout prefix=k5u;
    var qt_mean;
```

```
    by bw p;

data bias;
    merge k4uout k5uout;
        array k4u(*) k4u1;
    array k5u(*) k5u1;
          bt_bias=k5u1-k4u1;
      bias_sq=(k5u1-k4u1)**2;
      output;
keep bw p bt_bias bias_sq;

%mend;

%macro mise;
proc transpose data=variance out=k6uout prefix=k6u;
    var bt_var;
    by bw p;
proc transpose data=bias out=k7uout prefix=k7u;
    var bias_sq;
    by bw p;

data mse;
    merge k6uout k7uout;
        array k6u(*) k6u1;
    array k7u(*) k7u1;
```

```
            bt_var=k6u1;
        bias_sq=k7u1;
        bt_mse=k6u1+k7u1;
        output;
keep bw p bt_var bias_sq bt_mse;

proc transpose data=mse out=k8uout prefix=k8u;
     var bt_mse;
     by bw;

data mise;
     merge k8uout;
         array k8u(*) k8u1-k8u&numqt;
           bt_mise=(.01)* sum(of k8u1-k8u&numqt);
       output;
keep bw bt_mise;

proc sort data=mise;
     by bt_mise;
%mend;

%variance_n_bias
%mise
```

```
data saslib.mise;
    set mise;
run;
```

Confidence band for the difference between two quantile functions

The code for constructing the simultaneous confidence band for the difference between two quantile functions shown in figure 4.1 in section 4.3 is given here. The data are given in Tables 19.16 - 19.21 in ([]). Once the data are put into a data set *one*, we use the the macros *samps*, *picksampone*, *pickallsamps*, f_hat, *km_quant* and *smoothqtestimate* as defined in section **??** to compute the quantile estimates for the samples, placebo and polymer though there were a few changes in codes. Those estimates are in the sets called *smoothqt* and *smoothqt_cat*, which are then used in the macros shown below to construct the confidence band.

```
libname saslib 'C:sasresult' ;
% let nsam=1; ** this must be equal to 1;
% let nnsam=1001; ** number of samples to generate - must be at least 2;
% let nnnsam=%eval(&nnsam − 1); ** number of bootstrap samples;
% let size1=110; ** sample size of polymer tx=1;
% let size2=112; ** sample size of placebo tx=0;
% let numqt=51; ** number of quantiles;
% let h=.65; ** bandwidth;
% let p_cent=.9;
%let seed1=1852163;
%let seed2=3612581;


data one;
    input tx weeks event;
    datalines;
```

[the data themselves are given in Tables 19.16 - 19.21 in ([])]

%samps

%picksampone

%pickallsamps

%f_hat

%km_quant

%smoothqtestimate

%macro choosesupdiff;

```
proc transpose data=smoothqt out=x1out prefix=x1o;
    var x;
    by isam;

proc transpose data=smoothqt_cap out=x2out prefix=x2o;
    var x;
    by isam;

data part1;
    merge x1out x2out;

data part2;
    set part1;
```

```
        array x1o(*) x1o1-x1o&numqt;
    array x2o(*) x2o1-x2o&numqt;
p=.24;
        do ss=1 to &numqt;
        p=p+.01;
        x_rad=x1o(ss);
        x_cap=x2o(ss);
        qtdiff=x1o(ss)-x2o(ss);
        output;
    end;
keep isam qtdiff x_rad x_cap p;

data originalqtdiff;
     set part2;
     if isam = 0;

data bootstrapqtdiff;
    set part2;
    if isam = 0 then delete;
        %do iisam=1 %to &nnnsam;
        data originalqtdiff&iisam; * Generating many sets in the do loop;
        set originalqtdiff;
        isam=&iisam;
     %end;
```

```
data duplicatingoriginal;
      set
        %do iisam=1 %to &nnnsam;
           originalqtdiff&iisam /*no semicolon here*/
        %end;
        ; * this semicolon is for set;

proc transpose data=duplicatingoriginal out=pout prefix=po;
      var qtdiff;
       by isam;

proc transpose data=bootstrapqtdiff out=ppout prefix=ppo;
      var qtdiff;
      by isam;

data supremum;
      merge pout ppout;
      array po(*) po1-po&numqt;
      array ppo(*) ppo1-ppo&numqt;
p=.24;
      do i=1 to &numqt by 1;
         p=p+.01;
         og_qtdiff=po(i);
         bt_qtdiff=ppo(i);
         diffofqtdiff=abs(ppo(i)-po(i));
         output;
```

```
    end;
keep isam p og_qtdiff bt_qtdiff diffofqtdiff;

data supremum2;
    set supremum;

proc sort data=supremum2;
    by isam diffofqtdiff;
run;

proc transpose data=supremum2 out=ddout prefix=qdi      var diffofqtdiff;
    by isam;

data findingd;
     merge ddout;
        array qdif(*) qdif1-qdif&numqt;
        do i=&numqt;
       sup_d_diff=qdif(i);
       output;
    end;
keep isam sup_d_diff;

proc sort data=findingd;
    by sup_d_diff;

%mend;
```

```
%macro findD;
proc transpose data=findingd out=qout prefix=qo;
     var sup_d_diff;

data two;
     merge qout;

data findd;
     set two;
        array qo(*) qo1-qo&nnnsam;
        PxNSAM=&p_cent*&nnnsam;
     int=int(PxNSAM);
     dec=PxNSAM-int;
        if dec = 0 then do;
                      d = (&size1 ** .5)*(qo(int)+ qo(int+1))/2;
                    end;
     else do;
            d = (&size1 ** .5) * qo(int+1);
         end;
         output;

keep d;
%mend findD;

%macro existenceofaconstant;
```

```
proc transpose data=originalqtdiff out=eout prefix=eo;
    var qtdiff;
data confband;
    set findd;
    merge eout;
        array eo(*) eo1-eo&numqt;
p=.24;
    do z=1 to &numqt;
        p=p+.01;
        uplim=eo(z)+(d /&size1 **.5);
        lowlim=eo(z)-(d /&size1 **.5);
        output;
    end;
keep p uplim lowlim;
proc transpose data=confband out=upout prefix=upo;
    var uplim;

proc transpose data=confband out=lowout prefix=lowo;
    var lowlim;

data yesorno;
    merge upout lowout;
        array upo(*) upo1-upo&numqt;
    array lowo(*) lowo1-lowo&numqt;
```

```
        if min(of upo1-upo&numqt) ¡ max(of lowo1-lowo&numqt) then yes=0;
    else do;
            yes=1;
        end;
keep yes;
%mend existenceofaconstant;
%choosesupdiff
%findD
%existenceofaconstant
data saslib.braintumor1B;
    set supremum2;
data saslib.braintumor2B;
    set findingd;
data saslib.braintumor3B;
    set findd;
data saslib.braintumor4B;
    set confband;
data saslib.braintumor5B;
    set originalqtdiff;
run;
*************************************************************************
Graph the difference between the two quantile functions over [.25, .75]
Test the nullhypothesis H0 : b=0
*************************************************************************;
data plot1;
    set tmp1.braintumor4B;
```

```
    rename p=x;
    rename uplim=up;

data plot2;
    set tmp1.braintumor4B;
    rename p=x;
    rename lowlim=low;

data plot;
    set plot1 plot2;
    keep x up low;

symbol1 i=join color=black;
symbol2 i=join color=black;

proc print data=plot;

proc gplot data=plot;
    plot up*x low*x/overlay;
run;
quit;

data originalqtdiff;
    set tmp1.braintumor5B;
```

```
proc gplot data=originalqtdiff;
      plot qtdiff*p;
run;
quit;
```

REFERENCES

[1] Akritas, M.G. (1986). Bootstrapping the Kaplan-Meier estimator. Journal of the American Statistical association, 81, 1032-1038.

[2] Cheng, K.F. (1984). On almost sure represenations for quantiles of the product-limit estimator with applications. Sankhya Series A, 46, 426-443.

[3] Csögő, M (1983). Quantile Processes with Statistical Applications. CMBS-NSF Regional Conference Series in Applied Mathematics, No. 42 (Society for Industrial and Applied Mathematics, Philadelphia).

[4] Doss, H and Gill, R. (1992). An elementary approach to weak convergence for quantile processes, with applications to censored survival data. Journal of the Ametican statistical Association, 87, 869-877.

[5] Gill, R.D. (1983). Largest sample behaviour of the produc-limit estimator on the whole line. Annals of Statistics, 11, 49-58.

[6] Hoel, P.G, Port, S.C, and Stone, C.J. (1972). Instroduction to stochastic processes, University of California, Los Angeles.

[7] Kaplan, E.L and Meier, P. (1958). Nonparametric estimator from imcomplette observations. Journal of the American Statistical Association, 53, 457-481.

[8] Lio, Y.L., Padgett, W.J. and Ýu, K.F. (1987). On Asymptotic Properties of a Kernel-Type Quantile Estimator from Censored Samples, Journal of Statistical Planning and Inference 14, 169-177.

[9] Nadaraya, E.A. (1964). Some New Estimates for Distribution Functions. Theory of Probability and Applications 9. 497-500.

[10] Padgett. W.J. and Thombs, L.A. (1986). Smooth Nonparametric Quantile Estimation Under Censoring: Simulations and Bootstrap Methods, Communications in Statistics. Simulation and Computation 15. 1003-1025.

[11] Padgett, W.J and Thombs, L.A. (1989). A smooth nonparametric quantile estimator from right-censored data. Statistics & probability letters, 7, 113-121.

[12] Padgett, W.J. (1986). A Kernel-Type Estimator of a Quantile Function from Right-censored Data. Journal of the American Statistical Association 81, 215-222.

[13] Piantadosi, S. (1997). Clinical Trial A Methdologic Perspective.

[14] Sander, J. (1975). The weak convergence of qualities of the product-limit estimator. Technical Report Number 5, Stanford University, Department of Statistics.

[15] Ying, Z. (1989). A note on the asymptotic properties of the product-limit estimator on the whole line, Statistics & probability letters, 7, 311-314.

BIOGRAPHICAL INFORMATION

Katsuhiro Uechi earned his B.Sc. in 2006 and M.Sc. in 2008 from Texas A&M University Commerce, Commerce, TX. He started his Ph.D. under the supervision of Dr. Shan Sun-Mitchell at the University of Texas at Arlington in 2008.

While a graduate student at UTA, he had the opportunity to teach, as an graduate assistant, several undergraduate courses such as college algebra, pre-calculus, business calculus and business statistics. Also, he was involved in several collaborative research projects with Biomedical Engineering department. His interest lies in clinical trial.