11-28-2007

Inference for Cox's Regression Model via a New Version of Empirical Likelihood

Ali Jinnah

INFERENCE FOR COX'S REGRESSION MODEL VIA A NEW VERSION OF EMPIRICAL LIKELIHOOD

by

ALI JINNAH

Under the direction of Yichuan Zhao

ABSTRACT

Cox Proportional Hazard Model is one of the most popular tools used in the study of Survival Analysis. Empirical Likelihood (EL) method has been used to study the Cox Proportional Hazard Model. In recent work by Qin and Jing (2001), empirical likelihood based confidence region is constructed with the assumption that the baseline hazard function is known. However, in Cox's regression model the baseline hazard function is unspecified. In this thesis, we re-formulate empirical likelihood for the vector of regression parameters by estimating the baseline hazard function. The EL confidence regions are obtained accordingly. In addition, Adjusted Empirical Likelihood (AEL) method is proposed. Furthermore, we conduct extensive simulation studies to evaluate the performance of the proposed empirical likelihood methods in terms of coverage probabilities by comparing with the Normal Approximation based method. The simulation studies show that all the three methods produce similar coverage probabilities.

INDEX WORDS: Cox's regression model, confidence range, Empirical Likelihood, Normal Approximation, Adjusted Empirical Likelihood, coverage probability.

INFERENCE FOR COX'S REGRESSION MODEL VIA A NEW VERSION OF EMPIRICAL LIKELIHOOD

by

ALI JINNAH

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Science

Georgia State University

2007

INFERENCE FOR COX'S REGRESSION MODEL VIA A NEW VERSION OF EMPIRICAL LIKELIHOOD

by

ALI JINNAH

Major Professor: Dr. Yichuan Zhao
Committee: Dr. Yu-Sheng Hsu
Dr. Xu Zhang
Dr. Yuanhui Xiao

Electronic Version Approval:

Office of Graduate Studies
College of Arts and Sciences
Georgia State University
December 2007

ACKNOWLEDGEMENT

I would like to acknowledge those who have helped me in the completion of this thesis.

First of all, I like to thank Dr. Zhao who is an extraordinary advisor, a great professor, and a great person. There is no way I could have finished this thesis without his generous support. I have learned tremendously under Dr. Zhao's guidance on this thesis and also by attending his classes in different semesters during the past two years. He has always been available whenever I had any questions.

I would also like to acknowledge the other members of my thesis committee, Dr. Yu-Sheng Hsu, Dr. Yuanhui Xiao and Dr. Xu Zhang for taking the time to read this thesis and provide useful comments. I have attended at least one class taught by two of the above mentioned professors and have found all of the classes taught by them very helpful.

I'd also like to thank all the other professors in the Mathematics & Statistics Department at Georgia State University for their support and guidance. It would not have been possible for me to complete the requirements of the graduate program without their guidance.

Lastly, I want to acknowledge the support of my parents, brother and sister-in-law who have always been there for me with words of encouragement and wisdom.

TABLE OF CONTENTS

LIST OF TABLES

Chapter One: Introduction

Survival Analysis is used in various industries such as Finance, Economics and Public Health. One of the applications of Survival Analysis is clinical trial where failure times of the subjects are monitored and the participants of the study group are subject to random censoring. One of the popular models used in the study of Survival Analysis is the Cox Proportional Hazard model, which is also known as Cox's regression model. This model allows us to quantify the relationship between the failure times and a set of explanatory variables. The equation for the Cox proportional hazard model given by Cox (1972) is as follows:

$$\lambda(t \mid \mathbf{Z}) = \lambda_0(t)\exp(\boldsymbol{\beta}_0^{\tau}\mathbf{Z}),$$

where λ_0 is an unknown baseline hazard function, $\mathbf{Z}$ is a p-vector covariates and $\boldsymbol{\beta}_0$ is a vector of regression coefficients. There are some papers in which their authors have evaluated the effectiveness of Empirical Likelihood (EL) method for Cox's regression model by conducting simulation studies and comparing with the results produced by the Normal Approximation (NA) method for Cox's regression model. One such paper is Qin and Jing (2001). In this paper, $\lambda_0(s)ds$ where s is any given time, is used in one of the formulae that are used in the construction of the region. But λ_0 is unknown and thus needs to be estimated first. In this thesis, we will first estimate $\lambda_0(t)$, and then propose the estimated EL method for the vector of regression parameters in the Cox's regression model. In order to improve coverage probability, we develop Adjusted Empirical Likelihood method. Finally, we will conduct simulation studies for NA, EL and Adjusted Empirical Likelihood (AEL) methods.

The NA method can be used for Cox's regression model by first estimating the regression parameters using the partial likelihood function, which is based on the data that as Cox (1975) suggested, does not carry information about baseline hazard function $\lambda_0(\cdot)$. We can use the property that an asymptotic $100(1-\alpha)\%$ confidence region for regression parameters can be obtained through normal approximation as Cox (1975) have suggested that the partial likelihood function of the regression parameters share the asymptotic properties of a full likelihood. The asymptotic $100(1-\alpha)\%$ confidence region for $\boldsymbol{\beta}$ based on normal approximation method is as follows

$$R_1 = \{\boldsymbol{\beta} : n(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta})^{\tau} I_n(\hat{\boldsymbol{\beta}})(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}) \leq \chi_p^2(\alpha)\},$$

where $I_n(\hat{\boldsymbol{\beta}})$ is the information matrix. (See Chapter 2 for definition).

The EL method, which has a number of features such as range respecting, transformation-preserving, asymmetric confidence interval, and Bartlett correctability, is a powerful nonparametric method that was first introduced by Thomas & Grunkemeier (1975). It is also known that the EL method also provides better coverage probability for small sample sizes as shown by DiCiccio et al. (1991), DiCiccio and Romano (1989, 1990) and Hall (1990). Unlike the NA method, the EL method does not require to estimate the limiting variance matrix. Also, the confidence region produced by the EL method is not necessarily symmetric and is adapted to the dataset. Thus, it gives a more representative way to make inferences about the parameter of interest by reflecting the nature of the underlying data. Owen (1988, 1990) introduced new empirical likelihood confidence regions for the mean of a vector of i.i.d. complete data. Since then, the EL method has been widely used to perform statistical inferences in various statistical

settings due to its excellent properties and well-recognized advantages when compared with the other methods such as NA and bootstrap methods.

In this thesis, we will refer to the EL method that is proposed by Owen (1990, 1991). Recent work of EL includes construction of simultaneous confidence band for right-censored data under a variety of settings as shown in Hollander et al. (1997), Einmahl and McKeague (1999), Li and Keilegom (2002) and by McKeague and Zhao (2002, 2005 and 2006). Recent works of EL also include linear model as in Owen (1991) and Chen (1993, 1994), linear model for missing data as in Wang and Rao (2001, 2002), partial linear regression model as in Wang and Jing (1999) and Shi and Lau (2000), regression analysis of long-term survival rate as in Zhao (2005), the semi-parametric additive risk model as in Zhao and Hsu (2005), missing response problem and the application in observational studies as in Qin and Zhang (2007), nonlinear errors in co-variables models with validation data as in Stute et al. (2007) and weighted EL as in Glenn and Zhao (2007) among others.

The method proposed by Owen (1990, 1991) involves testing whether $\boldsymbol{\beta}_0$ is the true parameter of $\boldsymbol{\beta}$ is equivalent to testing whether $EU(\boldsymbol{\beta}_0)=0$. Then the empirical log-likelihood ratio is calculated, which in turn is used to construct the $100(1-\alpha)\%$ confidence region. The confidence region for $\boldsymbol{\beta}$ is constructed as follows

$$R_2(\boldsymbol{\beta}) = \{\ \boldsymbol{\beta} : l(\boldsymbol{\beta}) \leq \chi_p^2(\alpha)\ \},$$

where $l(\boldsymbol{\beta}) = -2\log R\ (\boldsymbol{\beta})$ (See Chapter 2 for definition).

The AEL method is used to improve the probability coverage produced by the EL method by applying a correction. Several research studies have been conducted on the AEL method such as Zhao and Chen (2007), Rao and Scott (1981), Wang and Rao (2001, 2002) and Li and Wang (2003). Following the work of Zhao and Chen (2007), in this thesis we propose a confidence region that is constructed as follows

$$R_3 = \{\boldsymbol{\beta} : \hat{l}_{ad}(\boldsymbol{\beta}) \leq \chi_p^2(\alpha)\} \quad \text{(See Chapter 2 for definition).}$$

The organization of the thesis is as follows. Next chapter contains more detail analysis of the Cox's regression model and its application with NA, EL and the AEL methods. The subsequent chapters contain simulation study and analysis of the Cox's regression model and its application with the three methods mentioned above.

Chapter Two: Analysis of Cox's Regression Model

The partial likelihood function for the Cox's regression model is solved to estimate the regression parameters, $\boldsymbol{\beta}$. Let t_i's be failure times. Since censoring, which refers to elements of a study group leaving the study before the failure time is observed is possible, we observe $x_i = \min(t_i, c_i)$ and $\delta_i = I(x_i = t_i)$ where c_i's are censoring variables. The inference about true value $\boldsymbol{\beta}_0$ can be made if we can make the assumption that censoring is non-informative by treating the partial likelihood function as suggested by Cox (1972, 1975). The partial likelihood function, $L(\boldsymbol{\beta})$, which was suggested by Cox (1975), is calculated as follows

$$L(\boldsymbol{\beta}) = \prod_{i=1,\ldots,n;x_i \le T} \left[\frac{\exp(\boldsymbol{\beta}^\tau \mathbf{Z}_i)}{\sum_j^n \exp(\boldsymbol{\beta}^\tau \mathbf{Z}_i) I(x_j \ge x_i)} \right]^{\delta_i}, \text{ where } T \text{ satisfies } P(x_i \ge T) > 0.$$

The maximum partial likelihood estimator $\hat{\boldsymbol{\beta}}$ is calculated by solving for $U(\boldsymbol{\beta})$=0. $U(\boldsymbol{\beta})$, is the partial likelihood score function that is obtained by taking the first derivatives of log $L(\boldsymbol{\beta})$. The equation for the partial likelihood score function is as follows:

$$U(\boldsymbol{\beta}) = \sum_{i=1}^{n} \int_0^T \left(\mathbf{Z}_i - \frac{\hat{\alpha}_1(t, \boldsymbol{\beta})}{\hat{\alpha}_0(t, \boldsymbol{\beta})} \right) dN_i(t),$$

where $\mathbf{Z}$ is p-vector covariates, $\hat{\alpha}_0(t, \boldsymbol{\beta}) = n^{-1} \sum_i \exp(\boldsymbol{\beta}^\tau \mathbf{Z}_i) I(x_i \ge t)$,

$\hat{\alpha}_1(t, \boldsymbol{\beta}) = n^{-1} \sum_i \mathbf{Z}_i \exp(\boldsymbol{\beta}^\tau \mathbf{Z}_i) I(x_i \ge t)$ and $N_i(t) = I(x_i \le t, \delta_i = 1)$.

2.1 Analysis of NA Method

We need to estimate the vector of regression parameters $\boldsymbol{\beta}$ in order to use the NA method. We can use Newton-Raphson method to find the value of $\boldsymbol{\beta}$. The Newton-Raphson is an iterative method that is represented by the following equation

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)},$$

where x_{n+1} is the new estimate, x_n is the previous estimate, $f(x_n)$ is the function of x_n and $f'(x_n)$ is the derivative of $f(x_n)$. The iterative process continues until the difference between the new estimate and the previous estimate is very close to zero. Let $Y_i = (\mathbf{Z}_i^{\tau}, x_i, t_i)$ be i.i.d variables. Then, Tsiatis (1981) has shown that

$$n^{1/2}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}_0) \xrightarrow{L} N(0, I(\boldsymbol{\beta}_0)^{-1}), \quad \text{where } I(\boldsymbol{\beta}_0) = \lim_{n\to\infty} I_n(\boldsymbol{\beta}_0) \text{ and}$$

$$I_n(\boldsymbol{\beta}) = n^{-1}\sum_{i=1}^{n} \delta_i \left(\frac{\hat{\alpha}_2(x_i, \boldsymbol{\beta})}{\hat{\alpha}_0(x_i, \boldsymbol{\beta})} - \left(\frac{\hat{\alpha}_1(x_i, \boldsymbol{\beta})}{\hat{\alpha}_0(x_i, \boldsymbol{\beta})} \right)^{\otimes 2} \right)$$

is the information matrix, with $a^{\otimes 2} = aa^{\tau}$ and $\hat{\alpha}_2(t, \boldsymbol{\beta}) = n^{-1}\sum_i \mathbf{Z}_i^{\otimes 2} \exp(\boldsymbol{\beta}^{\tau}\mathbf{Z}_i) I(x_i \geq t)$. Then, an asymptotic 100(1-α) % confidence region for $\boldsymbol{\beta}$ based on normal approximation method is as follows

$$R_1 = \{ \boldsymbol{\beta} : n(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta})^{\tau} I_n(\hat{\boldsymbol{\beta}})(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}) \leq \chi_p^2(\alpha) \},$$

where $\chi_p^2(\alpha)$ is the (1-α)-th quantile of the chi-square distribution with p degrees of freedom.

2.2 Analysis of EL Method

Qin and Jing (2001) have suggested that testing of whether $\boldsymbol{\beta}_0$ is the true parameter of $\boldsymbol{\beta}$ is equivalent to testing whether $EU(\boldsymbol{\beta}_0) = 0$ where $U(\boldsymbol{\beta}_0)$ is given by

$$U(\boldsymbol{\beta}_0) = \sum_{i=1}^{n} \int_0^{\mathrm{T}} \left(\mathbf{Z}_i - \frac{\overset{\Lambda}{\alpha}_1(t, \boldsymbol{\beta}_0)}{\overset{\Lambda}{\alpha}_0(t, \boldsymbol{\beta}_0)} \right) d\, M_i(t) ,$$

where $M_i(t) = N_i(t) - \int_0^t \exp(\boldsymbol{\beta}_0^{\tau} \mathbf{Z}_i) I(x_i \geq s) \lambda_0(s) ds$, i = 1,…n are i.i.d martingales. However, the baseline line function $\lambda_0(s)$ in the Cox's regression model is unspecified and therefore needs to be estimated. We will replace $\lambda_0(s)ds$ with its estimated equivalent $d\overset{\Lambda}{\Lambda}_0(s)$ in the equation above. We propose the following updated formula for the score function

$$U(\boldsymbol{\beta}_0) = \sum_{i=1}^{n} \int_0^{\mathrm{T}} \left(\mathbf{Z}_i - \frac{\overset{\Lambda}{\alpha}_1(t, \boldsymbol{\beta}_0)}{\overset{\Lambda}{\alpha}_0(t, \boldsymbol{\beta}_0)} \right) d\, \overset{\Lambda}{M}_i(t)$$

where $\overset{\Lambda}{M}_i(t) = N_i(t) - \int_0^t \exp(\boldsymbol{\beta}_0^{\tau} \mathbf{Z}_i) I(x_i \geq s) d\overset{\Lambda}{\Lambda}_0(s)$, and

$$\hat{\Lambda}_0(t) = \sum_{i=1}^{n} \int_0^t \frac{dN_i(s)}{\sum_{j=1}^{n} I(x_j \geq s) e^{(\boldsymbol{\beta}_0^{\tau} \mathbf{Z}_j)}} .$$

The suggestion by Qin and Jing (2001) that testing whether $EU(\boldsymbol{\beta}_0) = 0$ can be done is based on the EL method proposed by Owen (1988, 1990). Let $p = (p_1,..., p_n)$ be a probability vector, i.e., $\sum_{i=1}^{n} p_i = 1$ and $p_i \geq 0$ for all i. For $1 \leq i \leq n$, let

$$W_{n,i} = \int_0^T \left(\mathbf{Z}_i - \frac{\overset{\Lambda}{\alpha}_1(t, \boldsymbol{\beta}_0)}{\overset{\Lambda}{\alpha}_0(t, \boldsymbol{\beta}_0)} \right) d\overset{\Lambda}{M}_i(t),$$

Then, the profile empirical likelihood ratio, evaluated at true parameter value $\boldsymbol{\beta}_0$, is defined by

$$R(\boldsymbol{\beta}_0) = \frac{\sup\{\prod_{i=1}^{n} p_i : \sum_{i=1}^{n} p_i = 1, \sum_{i=1}^{n} p_i W_{ni} = 0\}}{\sup\{\prod_{i=1}^{n} p_i : \sum_{i=1}^{n} p_i = 1\}}.$$

By using the Lagrangian multipliers, we can easily find that the numerator in $R(\boldsymbol{\beta}_0)$ is maximized when

$$p_i = \frac{1}{n}\{1 + \lambda^{\tau} W_{n,i}\}^{-1}, \quad i = 1,...,n,$$

where $\lambda = (\lambda_1,....,\lambda_p)^{\tau}$ is the solution of

$$\frac{1}{n}\sum_{i=1}^{n} \frac{W_{n,i}}{1 + \lambda^{\tau} W_{n,i}} = 0.$$

Similarly, by using Lagrangian multipliers, the denominator in $R(\boldsymbol{\beta}_0)$ attains its maximum n^{-n} at $p_i = n^{-1}$. Therefore we have,

$$R(\boldsymbol{\beta}_0) = \prod_{i=1}^{n} (np_i) = \prod_{i=1}^{n} \{1 + \lambda^{\tau} W_{n,i}\}^{-1}.$$

Zhao and Huang (2004) pointed out that under the above regularity conditions, the following two properties of $W_{n,i}$ holds:

(i) $n^{-1/2}\sum_{i=1}^{n} W_{n,i} \xrightarrow{D} N(0, I).$

(ii) Denote $\hat{I}_1(\boldsymbol{\beta_0}) = n^{-1}\sum_{i=1}^{n} W_{n,i}W_{n,i}^T$. Then, $\hat{I}_1(\boldsymbol{\beta_0}) \to I_1(\boldsymbol{\beta_0}) = I(\boldsymbol{\beta_0})$ in probability.

In general, if $I_1(\boldsymbol{\beta_0}) \neq I(\boldsymbol{\beta_0})$ with the assumption that the regularity conditions holds, then $-2\log R(\boldsymbol{\beta})$ converges in distribution to $r_1\chi^2_{1,1} + \cdots + r_p\chi^2_{p,1}$, where $\chi^2_{1,1}, \cdots, \chi^2_{p,1}$ are independent chi-square random variables with one degree of freedom and $r_1, ..., r_p$ are the eigenvalues of $\mathbf{I}_1^{-1}\mathbf{I}$. If we denote $\hat{l}(\boldsymbol{\beta}) = -2\log R(\boldsymbol{\beta})$. We know that it converges in distribution to χ^2_p. Thus, the asymptotic 100(1-α)% confidence region can be constructed as follows

$$R_2(\boldsymbol{\beta}) = \{\beta : \hat{l}(\boldsymbol{\beta}) \leq \chi^2_p(\alpha)\}.$$

2.3 Analysis of AEL Method

In order to improve the coverage probability that is produced by the EL method, Zhao and Chen (2007) following the works of Rao and Scott (1981), Wang and Rao (2001, 2002) and Li and Wang (2003) have proposed an alternative EL method, referred to as AEL method. Let $\rho(\boldsymbol{\beta}) = p / tr\{I_1^{-1}(\boldsymbol{\beta})I(\boldsymbol{\beta})\}$ with $tr(\cdot)$ denoting the trace vector. Then, following Rao and Scott (1981), the distribution of $\rho(\boldsymbol{\beta_0})(\chi^2_p)$ may be

approximated by χ_p^2. This implies that the asymptotic distribution of the Rao–Scott adjusted empirical likelihood ratio, $\tilde{l}_{ad}(\boldsymbol{\beta_0}) = \hat{\rho}(\boldsymbol{\beta_0})\hat{l}(\boldsymbol{\beta_0})$, may be approximated by χ_p^2, where the adjustment factor $\hat{\rho}(\boldsymbol{\beta})$ is $\rho(\boldsymbol{\beta})$ with $I_1(\boldsymbol{\beta})$ and $I(\boldsymbol{\beta})$ replaced by $\hat{I}_1(\boldsymbol{\beta})$ and $\hat{I}(\boldsymbol{\beta})$, respectively.

We now define an adjusted empirical likelihood ratio, by modifying $\rho(\boldsymbol{\beta})$ in $\tilde{l}_{ad}(\boldsymbol{\beta})$, whose asymptotic distribution exactly follows χ_p^2. Note that

$$\hat{\rho}(\boldsymbol{\beta}) = \frac{tr\{\hat{I}^{-1}(\boldsymbol{\beta})\hat{I}(\boldsymbol{\beta})\}}{tr\{\hat{I}_1^{-1}(\boldsymbol{\beta})\hat{I}(\boldsymbol{\beta})\}}.$$

We define $\hat{r}(\boldsymbol{\beta})$ to be $\hat{\rho}(\boldsymbol{\beta})$ with $\hat{I}(\boldsymbol{\beta})$ replaced by

$$\hat{S}(\boldsymbol{\beta}) = \left\{\sum_{i=1}^{n} W_{n,i}(\boldsymbol{\beta})/n\right\} \times \left\{\sum_{i=1}^{n} W_{n,i}(\boldsymbol{\beta})/n\right\}^{\tau}.$$

That is,

$$\hat{r}(\boldsymbol{\beta}) = \frac{tr\{\hat{I}^{-1}(\boldsymbol{\beta})\hat{S}(\boldsymbol{\beta})\}}{tr\{\hat{I}_1^{-1}(\boldsymbol{\beta})\hat{S}(\boldsymbol{\beta})\}}.$$

We define an AEL ratio by

$$\hat{l}_{ad}(\boldsymbol{\beta}) = \hat{r}(\boldsymbol{\beta})\hat{l}(\boldsymbol{\beta}).$$

Zhao and Chen (2007) have pointed out that under the above regularity conditions, the EL statistic $\hat{l}_{ad}(\boldsymbol{\beta})$ converges in distribution to χ_p^2. Therefore, an asymptotic $100(1-\alpha)\%$ AEL confidence region for $\boldsymbol{\beta}$ is given by

$$R_3 = \{\boldsymbol{\beta} : \hat{l}_{ad}(\boldsymbol{\beta}) \leq \chi_p^2(\alpha)\},$$

where $\chi_p^2(\alpha)$ is the same as before.

Chapter Three: Simulation Study

In this chapter, we will implement the proposed NA, EL and AEL methods based confidence regions by conducting simulation studies. Our objective is to compare coverage probabilities produced by the three methods. For the simulation studies, we will consider one-dimensional covariates.

To generate the simulated data, we have modified the method that Xu (2004) described to generate the simulation data for additive-multiplicative model and used the modified method to generate data for our Cox's regression model.

Let $\beta_0 = 1$, then our model becomes

$$\lambda(t \mid Z) = \lambda_0 e^{Z(t)}.$$

Let T be the failure time and C be the censoring time. We will choose censoring rates that are approximately 10%, 20%, 30%, 40% and 70%. For sample size n, values 30, 50, 100 and 150 are chosen.

1. Simulating covariate $Z_i(t)$

   Values for Z are drawn from uniform distribution U(0,1)

2. Simulating failure time T_i, i=1,…,n

   To simulate failure time T_i, we assume λ_0 follows a Weibull distribution. For any variable t with Weibull distribution, we can define as t ~ Weibull (α, λ), where α and λ are two parameters in Weibull distribution. The hazard function for Weibull distribution is

   $$\lambda_0(t) = \alpha\lambda t^{\alpha-1}$$

To simplify the simulation, we choose Weibull distribution with parameters α =1 and λ =1. Thus we get $\lambda_0(t) = 1$. Then our model follows an exponential distribution with hazard function $\lambda(t \mid Z) = e^{Z(t)}$. Survival function for exponentional distribution is defined as

$$S(t) = e^{-\lambda t}.$$

Since the hazard function for failure time T is known, we can derive the survival function S(t) and distribution function F(t) for T.

$$S(t) = e^{-\int_0^t \lambda(t|Z)dt} = e^{-\int_0^t e^Z t} = e^{-(e^Z)t}$$

$$F(t) = 1 - S(t) = 1 - e^{-e^Z t}.$$

We apply inverse transformation method to simulate failure time samples T distributed with cumulative function F(t). U is simulated from the uniform distribution U (0, 1), and the failure time T is obtained as follows:

$$T = (-e^{-Z}) \ln(1-U),$$

where Z is covariate simulated from previous step.

3. Simulating censoring time C_i, i=1,…,n. Censoring time C_i's are drawn from the uniform distribution U(0,k) where k is chosen to ensure a desired censoring rate (CR). CR is chosen to be 10%, 20%, 30%, 40%, and 70%, respectively. It is checked programmatically that the number of simulated censored data elements are in the range of $\pm .05$ of the CR.

The simulated data is generated 2000 times. The approximate coverage probabilities for the NA, EL and AEL methods based on the simulated data sets are simply the proportions of the data sets that satisfy the inequalities mentioned in sections 2.1, 2.2 and 2.3 respectively. Same simulated data is used to calculate confidence region for all the three methods. The nominal confidence levels α chosen are 0.90, 0.95 and 0.99, respectively. Confidence regions are created for $T = 3 \text{ and } T = 5$, where all the observed failure times or censored times are less than or equal to T. β_0, which is the true value is chosen as one. The results are presented in Tables 3-1, 3-2 and 3-3 for nominal confidence levels 0.90, 0.95, and 0.99, respectively.

Table 3-1. Coverage Probabilities for $\beta_0 = 1$ at Level 0.90

		T=3			**T=5**		
CR	n	NA	EL	AEL	NA	EL	AEL
10%	30	90.85	87.00	88.80	90.90	87.00	88.80
	50	91.60	89.10	91.00	91.60	89.05	91.00
	100	91.60	90.20	91.90	91.60	90.20	91.90
	150	90.22	89.96	90.22	90.31	90.05	90.13
20%	30	90.40	85.95	88.85	90.30	85.95	88.80
	50	90.25	87.55	89.70	90.20	87.55	89.70
	100	90.80	89.65	91.25	90.9	89.6	91.25
	150	90.55	89.75	91.05	90.5	89.85	91.05
30%	30	92.00	86.90	90.90	92.00	86.90	90.90
	50	91.00	88.45	90.90	91.00	88.45	90.90
	100	90.10	89.15	90.35	90.10	89.15	90.35
	150	90.45	88.65	90.57	90.45	88.65	90.57
40%	30	91.20	87.20	89.80	91.20	87.20	89.80
	50	89.85	88.00	89.90	89.85	88.00	89.90
	100	90.65	89.00	90.45	90.65	89.00	90.45
	150	89.70	88.30	89.80	89.70	88.30	89.80
70%	30	90.35	85.80	87.15	90.35	85.80	87.15
	50	90.15	88.05	89.30	90.15	88.05	89.30
	100	90.50	90.00	90.55	90.50	90.00	90.55

	150	89.90	89.35	89.75	89.90	89.35	89.75

Table 3-2. Coverage Probabilities for $\beta_0 = 1$ at Level 0.95

		T=3			**T=5**		
CR	n	NA	EL	AEL	NA	EL	AEL
10%	30	95.90	92.20	93.55	95.90	92.20	93.55
	50	96.05	94.30	95.15	96.05	94.30	95.15
	100	96.30	96.00	95.95	96.30	96.00	95.95
	150	95.32	94.37	95.15	95.32	94.37	95.15
20%	30	96.30	92.15	94.50	96.30	92.15	94.50
	50	95.50	93.45	95.00	95.50	93.45	95.00
	100	95.90	94.60	96.00	95.85	94.55	96.00
	150	94.80	94.48	95.37	94.92	94.48	95.24
30%	30	96.60	93.30	95.35	96.60	93.30	95.35
	50	95.95	93.35	95.30	95.95	93.35	95.30
	100	95.65	94.40	95.70	95.65	94.40	95.70
	150	95.01	94.53	95.07	95.01	94.53	95.07
40%	30	95.50	92.60	94.25	95.50	92.60	94.25
	50	95.35	93.35	94.80	95.35	93.35	94.80
	100	95.60	94.70	95.55	95.60	94.70	95.55
	150	95.10	94.00	95.55	95.10	94.00	95.55
70%	30	95.85	91.85	92.40	95.85	91.85	92.40
	50	96.10	93.95	94.15	96.10	93.95	94.15
	100	95.80	95.30	95.25	95.80	95.30	95.25

	150	95.15	94.80	95.15	95.15	94.80	95.15

Table 3-3. Coverage Probabilities for $\beta_0 = 1$ at Level 0.99

		T=3			**T=5**		
CR	n	NA	EL	AEL	NA	EL	AEL
10%	30	99.30	96.90	97.60	99.30	96.90	97.60
	50	99.25	98.30	98.80	99.25	98.30	98.80
	100	99.15	98.50	98.85	99.15	98.50	98.85
	150	99.30	99.48	99.22	99.30	99.48	99.22
20%	30	99.50	97.75	98.25	99.50	97.75	98.25
	50	99.50	98.35	99.05	99.50	98.35	99.05
	100	99.39	98.73	99.27	99.39	98.73	99.27
	150	98.92	98.35	98.98	98.92	98.41	99.04
30%	30	99.60	97.70	98.90	99.60	97.70	98.90
	50	99.40	98.65	98.95	99.40	98.65	98.95
	100	99.40	98.80	99.15	99.40	98.80	99.15
	150	99.39	98.73	99.27	99.39	98.73	99.27
40%	30	99.50	97.55	98.20	99.50	97.55	98.20
	50	99.20	98.35	98.75	99.20	98.35	98.75
	100	98.90	98.30	98.90	98.90	98.30	98.90
	150	99.15	98.65	99.30	99.15	98.65	99.30
70%	30	99.75	97.50	96.90	99.75	97.50	96.90
	50	99.15	98.25	98.05	99.15	98.25	98.05
	100	99.35	99.10	99.10	99.35	99.10	99.10

	150	98.90	98.80	99.00	98.90	98.80	99.00

Chapter Four: Conclusion

Our main contribution in this thesis is to apply the correction to EL method in Qin and Jing (2001), where the confidence region for EL method is constructed with the assumption that the baseline line hazard function $\lambda_0(s)$ is known. Hence, that is not an EL method for the true Cox's regression model. In this thesis we make the correction, which is to estimate $\lambda_0(t)dt$ by $d\hat{\Lambda}_0(t)$ for the EL method. Then, we apply the plug-in estimator and further explore the EL method by constructing confidence regions with EL and AEL methods and comparing them with NA method to observe as to which method is better in terms of coverage probability.

In the simulation studies related to this thesis, it is observed that all three methods produce similar accurate coverage probability for small and large sample sizes. However, for very small sample size 30 we observed that the NA method has some over coverage problem and the EL and AEL methods have some under coverage problem. For example in the case of heavy censoring rate of 70%, it was observed that the probability coverage produced by the NA method was significantly higher than the nominal levels. Even though NA method produced some over coverage for sample size 30, the coverage produced by the NA method were much closer to the nominal levels than the coverage produced by the EL method.

The other important observation made from the simulation studies is that the AEL method produced better probability coverage than the EL method. Similar results were obtained for both $T = 3$ and $T = 5$ from these tables.

As the censoring rate increases, the simulation studies show that the accuracy of coverage probability decreases for fixed sample size. This observation is not so apparent

when we compare results produced for censoring rates 10%, 20% and 30%. But it is relatively clearer when we compare results produced for censoring rates 10%, 40% and 70%. The observation, that the accuracy of coverage probability decreases as the censoring rate increases is expected because of the fact that information on the subjects is lost when they leave a cohort or study group.

It is clearly evident from the simulation studies that the AEL method produces better results than the EL method. For all the censoring rates, sample sizes, nominal levels and values for T, AEL method outperforms the EL method. For sample sizes larger than 30, it is observed that the AEL method in same cases also outperforms the NA method.

In conclusion, the accuracy of the EL and AEL methods proposed in this thesis has been confirmed for sample sizes greater than 30. However, similar results were obtained for NA method and as mentioned above, NA method outperforms both of these methods and also has an over coverage problem for sample size 30. Perhaps, it can be mathematically proven that given the conditions used for the simulation studies in this thesis, NA method is better than the EL and AEL methods for sample size 30. We recommend that we conduct more simulation studies such as for true values different from one to see if better results are observed for EL and AEL methods for samples size 30. And, also to observe the performance of NA method for true values different from one. Moreover, given the conditions used for the simulation studies in this thesis, we conclude that AEL method produces better results than the EL method. We also recommend that we explore the Bartlett correction to improve the coverage probability for the EL and AEL methods for small sample size 30.

References

Chen, S. X. (1993). *On the accuracy of empirical likelihood confidence regions for linear regression model.* Ann. Inst. Statist. Math. 45, 621-637.

Chen, S. X. (1994). *Empirical likelihood confidence intervals for linear regression coefficients.* J. Multivariate Anal. 49, 24-40.

Cox, D.R. (1972). *Regression model and life tables (with discussion).* Biometrika, 62, 269-276.

DiCiccio, T.J., Hall, P. and Romano, J.P. (1991). *Empirical likelihood is Bartlett-correctable.* Ann. Statist. 19, 1053-1061.

DiCiccio, T.J. and Romano, J.P. (1989). *On adjustments based on the signed root of the empirical likelihood ratio statistic.* Biometrika 76, 447-456.

DiCiccio, T.J. and Romano, J.P. (1990). *Nonparametric confidence-limits by resampling methods and least favorable families.* Int. Statist. Rev. 58 (1990) 59-76.

Einmahl, J.H. and McKeague, I.W. (1999). *Confidence tubes for multiple quantile plots via empirical likelihood.* Ann. Statist. 27, 1348-1367.

Glenn, N.L. and Zhao, Y. (2007). *Weighted empirical likelihood estimates and their robustness properties* . Comput. Statist. Data Anal. 51, 5130-5141.

Hall, P. (1990). *Pseudo-likelihood theory for empirical likelihood.* Ann. Statist. 18, 121-140.

Hollander, M., McKeague, I.W. and Yang, J. (1997). *Likelihood ratio-based confidence bands for survival functions.* J. Amer. Statist. Assoc. 92, 215-226.

Li, G. and Kielegom, I. Van. (2002). *Likelihood ratio confidence bands in nonparametric regression with censored data.* Scand. J. Statist. 29, 547-562.

Li, G. and Wang, Q. H. (2003). *Empirical likelihood regression analysis for right censored data.* Statist. Sinica 13, 51-68.

McKeague, I. W. and Zhao, Y. (2002). *Simultaneous confidence bands for ratios of survival functions via empirical likelihood.* Statist. Probab. Lett. 60, 405-415.

McKeague, I. W. and Zhao, Y. (2005). *Comparing distribution functions via empirical likelihood.* Int. J. Biostatist., article 5, 1-20.

McKeague, I. W. and Zhao, Y. (2006). *Width-scaled confidence bands for survival functions.* Statist. Probab. Lett. 76, 327-339.

Owen, A. (1988). *Empirical likelihood ratio confidence intervals for a single functional.* Biometrika, 75, 237-249.

Owen, A. (1990). *Empirical likelihood ratio confidence Regions.* Ann. Statist. 18, 90-120.

Owen, A. (1991). *Empirical likelihood for linear models.* Ann. Statist. 19, 1725-1747.

Qin, G. and Jing, B.Y.(2001). *Empirical likelihood for Cox regression model under random censorship.* Commun. Statist. Simulation Comput., 30, 79-90.

Qin, J. and Zhang, B. (2007). *Empirical-likelihood-based inference in missing response problems and its application in observational studies.* J. R. Statist. Soc. B 69, 101-122.

Rao, J.N.K and Scott, A.J. (1981). *The analysis of categorical data from complex sample surveys: chi-squared tests for goodness of fit and independence in two-way tables.* J.Amer. Statist. Assoc. 76, 221-230.

Shi, J. and Lau, T.S. (2000). *Empirical likelihood for partially linear models.* J. Multivariate Anal. 72, 132-148.

Stute, W., Xue, L. and Zhu, L. (2007). *Empirical likelihood inference in nonlinear error-in-covariables models with validation data* . J. Amer. Statist. Assoc. 102, 332-346.

Thomas, D.R. and Grunkemeier, G.L. (1975). *Confidence interval estimation for survival probabilities for censored data.* 70, 865-871.

Tsiatis, A.A. (1981). *A large sample study of Cox regression model.* Ann. Statist. 9, 93-108.

Wang, Q. H and Jing, B.Y. (1999). *Empirical likelihood for partial linear models with fixed designs.* Statist. Probab. Lett. 41, 425-433.

Wang, Q. H and Rao, J.N.K. (2001). *Empirical likelihood for linear regression models under imputation for missing responses.* Canad. J. Statist. 29, 597-608.

Wang, Q. H and Rao, J.N.K. (2002). *Empirical likelihood-based inference in linear models with missing data.* Scand. J. Statist. 29, 563-576.

Xu, Z. (2004). *Inference for general additive-multiplicative hazard model and mixed discrete-continuous Cox regression model via empirical likelihood.* Thesis at Georgia State University.

Zhao, Y. (2005). *Regression analysis for long-term survival rate via empirical likelihood.* J. Nonparametric Statist. 17, 995-1007.

Zhao, Y. and Chen, F (2007). *Empirical likelihood inference for censored median regression model via nonparametric kernel estimation.* J. Multivariate Anal. doi: 10.1016/j.jmva.2007.05.002.

Zhao, Y. and Hsu, Y.S. (2005). *Semiparametric analysis for additive risk model via empirical likelihood.* Comm. Statist. Simulation Comput. 34, 135-143.

Zhao, Y. and Huang, Y. (2004). Empirical likelihood inference for calibration regression model with lifetime medical cost, manuscript.

APPENDIX

```
#*************************************************************************
#Function: GenerateSimulatedData
#Parameters:  1) n = sample size
#            2) mu= The configured value that would assist in
#                  generating simulated data with the desired
#                  censored rate.
#Description: This function generates simulated data needed for simulation
#            studies
#
#*************************************************************************
GenerateSimulatedData<-function(n,mu)
{

          prop<-0
          censor<-0
          Z<-runif(n)
          delta<-0
          Data<-0
          C<-0
          U<-0
          T<-0

          TimeData<-0

          for (i in 1:n)
            { C[i]<-mu*runif(1)
            U<-runif(1)
           T[i]<- (-1* exp(-1*Z[i]))*log(1-U)
          if (T[i] > C[i])
            {  delta[i]<-0
            censor<-censor + 1
          }
       else
          delta[i] <-1
       end
     }

prop<-censor/n

for (i in 1:n)
{
                    TimeData[i]=min(T[i],C[i])
```

```
}

TimeDataSort<-sort(TimeData,method="shell",index=TRUE)
Deltasort<-0
Zsort<-0

for (k in 1:n)
  {  i<-TimeDataSort$ix[k]
  Deltasort[k]=delta[i]
  Zsort[k]=Z[i]
}

return(TimeDataSort,Deltasort,Zsort,prop,censor)

}

#*********************************************************************
#Function: findBeta2
#Parameters:  1) n = sample size
#          2) TimeDataSort = It contains two vectors; 1st vector
#                            contains the generated time in order and
#                            the second contains the index value
#                            corresponding to the generated time
#          3) Zsort = The value simulated corresponding to a particular
#                   time (index) in the TimeDataSort time vector
#
#Purpose: This function is used to generate the estimated value of Beta.
#       This function uses the partial log-likelihood method and
#       calculates score function to generate the estimate value of Beta.
#       Calls a function that calculates score function.  True
#       Beta of one is assumed.  Calculats Beta Est using Newton Raphson
#       method.
#*********************************************************************

findBeta2<-function(n,TimeDataSort,Zsort,DeltaSort,bigT)
{
 BetaEst<-1
 AlphaZeroEstInitialValues<-0

 # The following two for-loops calculates Alpha0.  Refer to the thesis for more info.

 for (i in 1:n)
    AlphaZeroEstInitialValues[i]<-exp(BetaEst*Zsort[i])

 AlphaZeroEstCumValues<-0
```

```
        for (i in 1:n)
    AlphaZeroEstCumValues[i]<-sum(AlphaZeroEstInitialValues[i:n])


  # The following two for-loops calculates Alpha1.  Refer to the thesis for more info.
  AlphaOneEstInitialValues<-0
  for (i in 1:n)
    AlphaOneEstInitialValues[i]<-Zsort[i]* (exp(BetaEst*Zsort[i]))

  AlphaOneEstCumValues<-0
        for (i in 1:n)
    AlphaOneEstCumValues[i]<-sum(AlphaOneEstInitialValues[i:n])


  # Calculates Alphabar.
  AlphaBar<-AlphaOneEstCumValues/AlphaZeroEstCumValues




  valueFunction<-
findValueScoreFunction3(BetaEst,TimeDataSort,Zsort,DeltaSort,n,bigT,AlphaBar)
  valueDerivative<-
findValueDerivative3(BetaEst,TimeDataSort,Zsort,DeltaSort,n,bigT,AlphaBar)
  newEst<-BetaEst - valueFunction/valueDerivative




  diff<-newEst - BetaEst
  BetaEst<-newEst
  diffOutOfRangeFlag<-0
  iter<-1

  if (is.nan(diff)==TRUE)
        { BetaEst<-9999
          diffOutOfRangeFlag=1
          diff=0
         }
  #Calculating Beta Est using Newton Raphson method

  while ( abs(diff) > .001 && diffOutOfRangeFlag==0 && iter<=10)

    {
                AlphaZeroEstInitialValues<-0
                        for (i in 1:n)
```

```
                    AlphaZeroEstInitialValues[i]<-exp(BetaEst*Zsort[i])

                        AlphaZeroEstCumValues<-0
                            for (i in 1:n)
                    AlphaZeroEstCumValues[i]<-
sum(AlphaZeroEstInitialValues[i:n])

                        AlphaOneEstInitialValues<-0
                        for (i in 1:n)
                    AlphaOneEstInitialValues[i]<-Zsort[i]* (exp(BetaEst*Zsort[i]))

                        AlphaOneEstCumValues<-0
                            for (i in 1:n)
                    AlphaOneEstCumValues[i]<-sum(AlphaOneEstInitialValues[i:n])

                        AlphaBar<-
AlphaOneEstCumValues/AlphaZeroEstCumValues



        valueFunction<-
findValueScoreFunction3(BetaEst,TimeDataSort,Zsort,DeltaSort,n,bigT,AlphaBar)
                    valueDerivative<-
findValueDerivative3(BetaEst,TimeDataSort,Zsort,DeltaSort,n,bigT,AlphaBar)
                    newEst<-BetaEst - valueFunction/valueDerivative
                    diff<-newEst - BetaEst
                    BetaEst<-newEst

                    if (abs(diff)>100)
                        {  diffOutOfRangeFlag=1
                            cat("diff= ")
                            cat(diff)
                            cat("\n")
                            cat("BetaEst= ")
                            cat(BetaEst)
                            cat("\n")
                            cat("newEst= ")
                            cat(newEst)
                            cat("\n")
                            cat("valueFunction= ")
                            cat(valueFunction)
                            cat("\n")
                            cat("valueDerivative= ")
                            cat(valueDerivative)
        cat("\n")
        BetaEst<-9999
```

```
        }
        iter<-iter+1

                }

if (iter>10)
        {
                cat("Iteration is=")
                cat(iter)
                cat("\n")
                cat("diff is=")
                cat(diff)
                cat("\n")
                cat("BetaEst is=")
                cat(BetaEst)
                cat("\n")
                cat("valueFunction= ")
  cat(valueFunction)
        cat("\n")
        cat("valueDerivative= ")
        cat(valueDerivative)
  cat("\n")
         BetaEst<-9999
        }


findBeta2<-BetaEst
}

#*************************************************************************
#Function: findValueScoreFunction3
#Parameters:  1) betaEst = Current estimate of beta
#          2) TimeDataSort = It contains two vectors; 1st vector
#                            contains the generated time in order and
#                            the second contains the index value
#                            corresponding to the generated time
#          3) Zsort = The value simulated corresponding to a particular
#                   time (index) in the TimeDataSort time vector
#          4) DeltaSort = contains a vector indicating if a particular
#                     simulated value is censored (1) or not (0)
#          5)  n = Sample size
#          6)  bigT = Upper limit for the integration
#          7)  AlphaBar = AlphaOne/AlphaZero
#
#Purpose: This function is to get the newest beta estimate by being
#        called recursively from the findBeta function
```

```
#*********************************************************************



findValueScoreFunction3<-
function(betaEst,TimeDataSort,Zsort,Deltasort,n,bigT,AlphaBar)
 {

 S2<-0
Nt2<-0
Nt<-0
for (i in 1:n)
    Nt2[i]<-0

start<-0

#cat("In findValueScoreFunction\n")

for (i in 1:n)
   { t<-TimeDataSort$x[i]
     TimeTemp<-TimeDataSort$x




    if (Deltasort[i]==1 && t<=bigT)
        { ig2<-(Zsort[i]- AlphaBar[i])
          S2<-S2+ig2
        }


           #S2 was manually checked
    Nt2<-Nt
    start<-t
   }
 findValueScoreFunction3<-S2

 }

#*********************************************************************
#Function: findValueDerivative3
#Parameters:  1) betaEst = Current estimate of beta
#            2) TimeDataSort = It contains two vectors; 1st vector
#                              contains the generated time in order and
#                              the second contains the index value
#                              corresponding to the generated time
```

```
#           3) Zsort = The value simulated corresponding to a particular
#                      time (index) in the TimeDataSort time vector
#           4) DeltaSort = contains a vector indicating if a particular
#                        simulated value is censored (1) or not (0)
#           5)  n = Sample size
#           6)  bigT = Upper limit for the integration
#           7)  AlphaBar = AlphaOne/AlphaZero
#
#Purpose: This function is to get the derivative of the score function,
#         which is used in recursively called function to get the beta
#         estimate by using the Newton Raphson method
#*********************************************************************


findValueDerivative3<-function(betaEst,TimeDataSort,Zsort,Deltasort,n,bigT,AlphaBar)

{

Jacob22<-0

AlphaZeroEstInitialValues<-0
  for (i in 1:n)
     AlphaZeroEstInitialValues[i]<-exp(betaEst*Zsort[i])

  AlphaZeroEstCumValues<-0
          for (i in 1:n)
     AlphaZeroEstCumValues[i]<-sum(AlphaZeroEstInitialValues[i:n])
  Xbarbot<-AlphaZeroEstCumValues

Nt2<-0
for (i in 1:n)
   Nt2[i]<-0
Nt<-0


start<-0
for (i in 1:n)
  {
   TimeTemp<-TimeDataSort$x
   t<-TimeTemp[i]


    if (t<=bigT)
    {
     Yt<-0
     for (k in 1:n)
```

```
      {
       if (TimeTemp[k]>=t)
            Yt[k]<-1
       else
            Yt[k]<-0
       end
       }

     temp<-0
         for (k in 1:n)
       {
       temp<-temp+ (Yt[k]*((Zsort[k])^2)*exp(betaEst*Zsort[k]))

       }
          if (Deltasort[i]==1 && t<=bigT)
                                    {

                                        integrand22=-temp/Xbarbot[i]+AlphaBar[i]^2;
                                                Jacob22=Jacob22+integrand22
                          }
    #}


   Nt2<-Nt
   start<-t
 }
 }
findValueDerivative3<-Jacob22

}

#*********************************************************************
#Function: calculateInformationMatrix3
#Parameters:  1) betaEst = Current estimate of beta
#          2) TimeDataSort = It contains two vectors; 1st vector
#                            contains the generated time in order and
#                            the second contains the index value
#                            corresponding to the generated time
#          3) Zsort = The value simulated corresponding to a particular
#                   time (index) in the TimeDataSort time vector
#          4) DeltaSort = contains a vector indicating if a particular
#                     simulated value is censored (1) or not (0)
#          5)  n = Sample size
#          6)  bigT = Upper limit for the integration
#
#
```

```
#Purpose: This function calculates the Information scalor quantity that
#       is eventually needed to calculate the Normal Approximation based
#       region
#*********************************************************************

calculateInformationMatrix3<-function(TimeDataSort,betaEst,Zsort,n,Deltasort,bigT)
{ S2<-0
Nt2<-0
Nt<-0

for (i in 1:n)
   Nt2[i]<-0

  AlphaZeroEstInitialValues<-0
  for (i in 1:n)
     AlphaZeroEstInitialValues[i]<-exp(betaEst*Zsort[i])

  AlphaZeroEstCumValues<-0
          for (i in 1:n)
     AlphaZeroEstCumValues[i]<-sum(AlphaZeroEstInitialValues[i:n])

  AlphaOneEstInitialValues<-0
  for (i in 1:n)
     AlphaOneEstInitialValues[i]<-Zsort[i]* (exp(betaEst*Zsort[i]))

  AlphaOneEstCumValues<-0
          for (i in 1:n)
     AlphaOneEstCumValues[i]<-sum(AlphaOneEstInitialValues[i:n])

          AlphaTwoEstInitialValues<-0
  for (i in 1:n)
     AlphaTwoEstInitialValues[i]<-Zsort[i]^2* (exp(betaEst*Zsort[i]))

  AlphaTwoEstCumValues<-0
          for (i in 1:n)
     AlphaTwoEstCumValues[i]<-sum(AlphaTwoEstInitialValues[i:n])

  InfoMatrix<-Deltasort*( (AlphaTwoEstCumValues/AlphaZeroEstCumValues) -
(AlphaOneEstCumValues/AlphaZeroEstCumValues)^2 )

  InfoMatrixSum<-sum(InfoMatrix)
                  InfoMatrixSum<-InfoMatrixSum/n
                  return(InfoMatrixSum)
}

#*********************************************************************
```

```
#Function: FindWni6
#Parameters:  1) beta0 = True Beta.  In this case is 1
#          2) TimeDataSort = It contains two vectors; 1st vector
#                            contains the generated time in order and
#                            the second contains the index value
#                            corresponding to the generated time
#          3) Zsort = The value simulated corresponding to a particular
#                   time (index) in the TimeDataSort time vector
#          4) DeltaSort = contains a vector indicating if a particular
#                     simulated value is censored (1) or not (0)
#          5)  n = Sample size
#          6)  bigT = Upper limit for the integration
#
#
#Purpose: The function calculates the Wni Vector, which is eventually used
#        in constructing region based on the Empirical Likelihood method
#*********************************************************************



FindWni6<-function(beta0,TimeDataSort,Zsort,Deltasort,n,bigT)
 {
  number<-1000
  Wni<-c(rep(0,n))
  Yt<-0
  cumVector<-0

  for (i in 1:n)
                cumVector[i]<-exp(beta0*Zsort[i])

  cumVectorTotal<-0

  for (i in 1:n)
                cumVectorTotal[i]<-sum(cumVector[i:n])

  AlphaZeroEstInitialValues<-0
 for (i in 1:n)
    AlphaZeroEstInitialValues[i]<-exp(beta0*Zsort[i])

 AlphaZeroEstCumValues<-0
        for (i in 1:n)
    AlphaZeroEstCumValues[i]<-sum(AlphaZeroEstInitialValues[i:n])

 AlphaOneEstInitialValues<-0
 for (i in 1:n)
    AlphaOneEstInitialValues[i]<-Zsort[i]* (exp(beta0*Zsort[i]))
```

```
 AlphaOneEstCumValues<-0
        for (i in 1:n)
    AlphaOneEstCumValues[i]<-sum(AlphaOneEstInitialValues[i:n])

 AlphaBar<-AlphaOneEstCumValues/AlphaZeroEstCumValues



for (i in 1:n)
 {
  TimeTemp<-TimeDataSort$x
  t<-TimeTemp[i]

     if (Deltasort[i]==1 && t<=bigT)
                Wni[i]<-(Zsort[i] - AlphaBar[i])



        cumValue<-0
   for (j in 1:i)
    {   s<-TimeTemp[j]




                if (Deltasort[j]==1 && TimeTemp[j]<=bigT)
                                        Wni[i]<-Wni[i] - ( (Zsort[i]-AlphaBar[j]) *
exp(beta0 * Zsort[i]) * Deltasort[j] / cumVectorTotal[j])

        }



 }

FindWni6<-Wni
}

#*********************************************************************
#Function: findLambda
#Parameters:  1) n = Sample size
#            2) WniVector = Wni Vector.
#
#
#Purpose: The function calculates the lambda by Netwon Raphson Method
```

```
#*********************************************************************


findLambda<-function(n,WniVector)
{

lambda<-0
diffOutOfRangeFlag<-0

valuefx<-0

for (i in 1:n)
   {
     valuefx<-valuefx + (WniVector[i]/(1+lambda*WniVector[i]))
   }

valueDerivativefx<-0

for (i in 1:n)
    {
      valueDerivativefx<-valueDerivativefx +
(WniVector[i]/(1+lambda*WniVector[i]))^2
    }

valueDerivativefx<-valueDerivativefx*(-1)
newlambda<-lambda - (valuefx/valueDerivativefx)

diff<-newlambda - lambda

while ( abs(diff) > 0.01 && diffOutOfRangeFlag==0)
  {  lambda<-newlambda
     valuefx<-0

                  for (i in 1:n)
         {
         valuefx<-valuefx + (WniVector[i]/(1+lambda*WniVector[i]))
         }

                  valueDerivativefx<-0

                  for (i in 1:n)
          {
           valueDerivativefx<-valueDerivativefx +
(WniVector[i]/(1+lambda*WniVector[i]))^2
                  }
```

```
   valueDerivativefx<-valueDerivativefx*(-1)

                newlambda<-lambda - (valuefx/valueDerivativefx)

                diff<-newlambda - lambda

   if (is.nan(diff)==FALSE)
        { if (abs(diff) > 100)
                        {
                                diffOutOfRangeFlag<-1
                                }
    }
   else
                diffOutOfRangeFlag<-1
  }


if (diffOutOfRangeFlag==1)
  newlambda<-(-9999)

findLambda<-newlambda
}

#*************************************************************************
#Function: calculateELRegion
#Parameters:  1) lambda = The estimated value calcuated by applying
#                    Newton Raphson method on the WniVector
#           2) WniVector = Wni Vector.
#           3) n = Sample size
#
#
#Purpose: The function calculates the EL region
#*************************************************************************



calculateELregion<-function(lambda,WniVector,n)
{
 Region<-0

 for (i in 1:n)
     { Region<-Region + log(1+ (lambda * WniVector[i]))
     }
 Region<-Region*2

 calculateELregion<-Region
```

```
 }


#*********************************************************************
#Function: runSimulation
#Parameters:  1) df = degrees of freedom
#            2) n = Sample Size
#            3) mu = configured value that generates the optimum amount
#                  of censored simulation data
#            4) censorProp = Desired censor proportion.
#            5) numOfSimulations = Number of simulations
#            6) trueBeta = True Beta.  In this case 1
#
#
#Purpose: The function is the overall function that generates probability
#         coverages based on NA, EL and AEL methods.
#*********************************************************************



runSimulation<-function(df,n,mu,censorProp,numOfSimulations,trueBeta)
{
#alpha<-0.1
#df<-1
resultsNA90T1<-0
resultsNA95T1<-0
resultsNA99T1<-0
resultsNA90T3<-0
resultsNA95T3<-0
resultsNA99T3<-0
resultsNA90T5<-0
resultsNA95T5<-0
resultsNA99T5<-0
resultsEL90T1<-0
resultsEL95T1<-0
resultsEL99T1<-0
resultsEL90T3<-0
resultsEL95T3<-0
resultsEL99T3<-0
resultsEL90T5<-0
resultsEL95T5<-0
resultsEL99T5<-0
resultsAEL90T1<-0
resultsAEL95T1<-0
resultsAEL99T1<-0
```

```
resultsAEL90T3<-0
resultsAEL95T3<-0
resultsAEL99T3<-0
resultsAEL90T5<-0
resultsAEL95T5<-0
resultsAEL99T5<-0




#mu<-5
#targetProp<-.20
simulations<-1
#bigT<-3
while (simulations <=numOfSimulations)
      {
           SimulatedData<-GenerateSimulatedData(n,mu)
           TimeDataSort<-SimulatedData$TimeDataSort
           Zsort<-SimulatedData$Zsort
           prop<-SimulatedData$prop
          cancelSimulationFlag<-0
          DeltaSort<-SimulatedData$Deltasort
          print (prop)
          {  cat("Current Simulation Number is:")
             cat(simulations)
             cat("\n")
            simulations<-simulations+1
           bigT<-1
   RegionNAT1<-runSimulationNA(n,TimeDataSort,Zsort,DeltaSort,bigT,trueBeta)
 if (RegionNAT1==9999)
          cancelSimulationFlag<-1
 bigT<-3
    RegionNAT3<-runSimulationNA(n,TimeDataSort,Zsort,DeltaSort,bigT,trueBeta)
 if (RegionNAT3==9999)
          cancelSimulationFlag<-1
 bigT<-5
 RegionNAT5<-runSimulationNA(n,TimeDataSort,Zsort,DeltaSort,bigT,trueBeta)
 if (RegionNAT5==9999)
          cancelSimulationFlag<-1
 bigT<-1
 RegionELT1<-runSimulationEL(n,TimeDataSort,Zsort,DeltaSort,bigT,trueBeta)
 if (RegionELT1==9999)
          cancelSimulationFlag<-1
 bigT<-3
 RegionELT3<-runSimulationEL(n,TimeDataSort,Zsort,DeltaSort,bigT,trueBeta)
```

```
 if (RegionELT3==9999)
        cancelSimulationFlag<-1
        bigT<-5
        RegionELT5<-runSimulationEL(n,TimeDataSort,Zsort,DeltaSort,bigT,trueBeta)
        if (RegionELT5==9999)
              cancelSimulationFlag<-1
        bigT<-1
LambdaAdjELT1<-
calculateInformationMatrix3(TimeDataSort,trueBeta,Zsort,n,DeltaSort,bigT)
bigT<-3
LambdaAdjELT3<-
calculateInformationMatrix3(TimeDataSort,trueBeta,Zsort,n,DeltaSort,bigT)
bigT<-5
LambdaAdjELT5<-
calculateInformationMatrix3(TimeDataSort,trueBeta,Zsort,n,DeltaSort,bigT)
LambdaAdjELT1<-LambdaAdjELT1/RegionELT1$LambdaAdjEL
LambdaAdjELT3<-LambdaAdjELT3/RegionELT3$LambdaAdjEL
LambdaAdjELT5<-LambdaAdjELT5/RegionELT5$LambdaAdjEL
RegionNAT1<-RegionNAT1$Region
RegionNAT3<-RegionNAT3$Region
RegionNAT5<-RegionNAT5$Region
RegionELT1<-RegionELT1$Region
RegionELT3<-RegionELT3$Region
RegionELT5<-RegionELT5$Region
chisqcomp90<-qchisq((0.90),df)
chisqcomp95<-qchisq((0.95),df)
chisqcomp99<-qchisq((0.99),df)

if (cancelSimulationFlag==0)
        {  if (RegionNAT1<=chisqcomp90)
              resultsNA90T1<-resultsNA90T1 + 1
        if (RegionNAT1<=chisqcomp95)
              resultsNA95T1<-resultsNA95T1 + 1
        if (RegionNAT1<=chisqcomp99)
              resultsNA99T1<-resultsNA99T1 + 1
        if (RegionNAT3<=chisqcomp90)
              resultsNA90T3<-resultsNA90T3 + 1
        if (RegionNAT3<=chisqcomp95)
              resultsNA95T3<-resultsNA95T3 + 1
        if (RegionNAT3<=chisqcomp99)
              resultsNA99T3<-resultsNA99T3 + 1
        if (RegionNAT5<=chisqcomp90)
              resultsNA90T5<-resultsNA90T5 + 1
        if (RegionNAT5<=chisqcomp95)
              resultsNA95T5<-resultsNA95T5 + 1
        if (RegionNAT5<=chisqcomp99)
```

```
                resultsNA99T5<-resultsNA99T5 + 1
if (RegionELT1<=chisqcomp90)
        resultsEL90T1<-resultsEL90T1 + 1
if (RegionELT1<=chisqcomp95)
        resultsEL95T1<-resultsEL95T1 + 1
if (RegionELT1<=chisqcomp99)
        resultsEL99T1<-resultsEL99T1 + 1
if (RegionELT3<=chisqcomp90)
        resultsEL90T3<-resultsEL90T3 + 1
if (RegionELT3<=chisqcomp95)
        resultsEL95T3<-resultsEL95T3 + 1
if (RegionELT3<=chisqcomp99)
        resultsEL99T3<-resultsEL99T3 + 1
if (RegionELT5<=chisqcomp90)
        resultsEL90T5<-resultsEL90T5 + 1
if (RegionELT5<=chisqcomp95)
        resultsEL95T5<-resultsEL95T5 + 1
if (RegionELT5<=chisqcomp99)
        resultsEL99T5<-resultsEL99T5 + 1
if (RegionELT1<=(chisqcomp90*LambdaAdjELT1))
        resultsAEL90T1<-resultsAEL90T1 + 1
if (RegionELT1<=(chisqcomp95*LambdaAdjELT3))
        resultsAEL95T1<-resultsAEL95T1 + 1
if (RegionELT1<=(chisqcomp99*LambdaAdjELT1))
        resultsAEL99T1<-resultsAEL99T1 + 1
if (RegionELT3<=(chisqcomp90*LambdaAdjELT3))
        resultsAEL90T3<-resultsAEL90T3 + 1
if (RegionELT3<=(chisqcomp95*LambdaAdjELT3))
        resultsAEL95T3<-resultsAEL95T3 + 1
if (RegionELT3<=(chisqcomp99*LambdaAdjELT3))
        resultsAEL99T3<-resultsAEL99T3 + 1
if (RegionELT5<=(chisqcomp90*LambdaAdjELT5))
        resultsAEL90T5<-resultsAEL90T5 + 1
if (RegionELT5<=(chisqcomp95*LambdaAdjELT5))
        resultsAEL95T5<-resultsAEL95T5 + 1
if (RegionELT5<=(chisqcomp99*LambdaAdjELT5))
        resultsAEL99T5<-resultsAEL99T5 + 1
cat("Current Results NA 90 T1:")
cat((resultsNA90T1/(simulations-1)*100))
cat("\n")
cat("Current Results NA 95 T1:")
cat((resultsNA95T1/(simulations-1)*100))
cat("\n")
cat("Current Results NA 99 T1:")
cat((resultsNA99T1/(simulations-1)*100))
cat("\n")
```

```
cat("Current Results NA 90 T3:")
cat((resultsNA90T3/(simulations-1)*100))
cat("\n")
cat("Current Results NA 95 T3:")
cat((resultsNA95T3/(simulations-1)*100))
cat("\n")
cat("Current Results NA 99 T3:")
cat((resultsNA99T3/(simulations-1)*100))
cat("\n")
cat("Current Results NA 90 T5:")
cat((resultsNA90T5/(simulations-1)*100))
cat("\n")
cat("Current Results NA 95 T5:")
cat((resultsNA95T5/(simulations-1)*100))
cat("\n")
cat("Current Results NA 99 T5:")
cat((resultsNA99T5/(simulations-1)*100))
cat("\n")
cat("Current Results EL 90 T1:")
cat((resultsEL90T1/(simulations-1)*100))
cat("\n")
cat("Current Results EL 95 T1:")
cat((resultsEL95T1/(simulations-1)*100))
cat("\n")
cat("Current Results EL 99 T1:")
cat((resultsEL99T1/(simulations-1)*100))
cat("\n")
cat("Current Results EL 90 T3:")
cat((resultsEL90T3/(simulations-1)*100))
cat("\n")
cat("Current Results EL 95 T3:")
cat((resultsEL95T3/(simulations-1)*100))
cat("\n")
cat("Current Results EL 99 T3:")
cat((resultsEL99T3/(simulations-1)*100))
cat("\n")
cat("Current Results EL 90 T5:")
cat((resultsEL90T5/(simulations-1)*100))
cat("\n")
cat("Current Results EL 95 T5:")
cat((resultsEL95T5/(simulations-1)*100))
cat("\n")
cat("Current Results EL 99 T5:")
cat((resultsEL99T5/(simulations-1)*100))
cat("\n")
cat("Lambda Adj EL T1")
```

```
cat((LambdaAdjELT1))
cat("\n")
cat("Lambda Adj EL T3")
cat((LambdaAdjELT3))
cat("\n")
cat("Lambda Adj EL T5")
cat((LambdaAdjELT5))
cat("\n")
cat("Current Results Adjusted EL 90 T1:")
cat((resultsAEL90T1/(simulations-1)*100))
cat("\n")
cat("Current Results Adjusted EL 95 T1:")
cat((resultsAEL95T1/(simulations-1)*100))
cat("\n")
cat("Current Results Adjusted EL 99 T1:")
cat((resultsAEL99T1/(simulations-1)*100))
cat("\n")
cat("Current Results Adjusted EL 90 T3:")
cat((resultsAEL90T3/(simulations-1)*100))
cat("\n")
cat("Current Results Adjusted EL 95 T3:")
cat((resultsAEL95T3/(simulations-1)*100))
cat("\n")
cat("Current Results Adjusted EL 99 T3:")
cat((resultsAEL99T3/(simulations-1)*100))
cat("\n")
cat("Current Results Adjusted EL 90 T5:")
cat((resultsAEL90T5/(simulations-1)*100))
cat("\n")
cat("Current Results Adjusted EL 95 T5:")
cat((resultsAEL95T5/(simulations-1)*100))
cat("\n")
cat("Current Results Adjusted EL 99 T5:")
cat((resultsAEL99T5/(simulations-1)*100))
cat("\n")
  }
       else
         { cat("Current Simualation cancelled and will be redone \n")
           simulations<-simulations - 1
         }
       end
            }
              }
            }

#*********************************************************************
```

```
#Function: runSimulationNA
#Parameters:  1)  n = Sample size
#           2) TimeDataSort = It contains two vectors; 1st vector
#                              contains the generated time in order and
#                              the second contains the index value
#                              corresponding to the generated time
#           3) Zsort = The value simulated corresponding to a particular
#                      time (index) in the TimeDataSort time vector
#           4) DeltaSort = contains a vector indicating if a particular
#                        simulated value is censored (1) or not (0)
#           5)  bigT = Upper limit for the integration
#           6)  trueBeta = True Beta
#
#Purpose: This function calcuates region for NA method
#*********************************************************************



runSimulationNA<-function(n,TimeDataSort,Zsort,DeltaSort,bigT,trueBeta)
{
  betaHat<-findBeta2(n,TimeDataSort,Zsort,DeltaSort,bigT)
  InfoMatrixSum<-0
  if (betaHat!=9999)
                { cat("In InfoMatrixSum..\n")
                        InfoMatrixSum<-
calculateInformationMatrix3(TimeDataSort,betaHat,Zsort,n,DeltaSort,bigT)
                        Region <- n* (betaHat - trueBeta) * InfoMatrixSum * (betaHat -
trueBeta)
                }
  if (betaHat==9999)
                        Region<-9999

  return(Region,InfoMatrixSum,betaHat)
# runSimulationNA<-Region
 }

#*********************************************************************
#Function: runSimulationEL
#Parameters:  1)  n = Sample size
#           2) TimeDataSort = It contains two vectors; 1st vector
#                              contains the generated time in order and
#                              the second contains the index value
#                              corresponding to the generated time
#           3) Zsort = The value simulated corresponding to a particular
#                      time (index) in the TimeDataSort time vector
```

```
#           4) DeltaSort = contains a vector indicating if a particular
#                       simulated value is censored (1) or not (0)
#           5)  bigT = Upper limit for the integration
#           6)  trueBeta = True Beta
#
#Purpose: This function calcuates region for EL
 method
#*********************************************************************


runSimulationEL<-function(n,TimeDataSort,Zsort,DeltaSort,bigT,trueBeta)
{
     WniVector<-FindWni6(trueBeta,TimeDataSort,Zsort,DeltaSort,n,bigT)
     lambda <-findLambda(n,WniVector)
     if (lambda!=(-9999))
                         {  Region<-calculateELregion(lambda,WniVector,n)
                  if (is.nan(Region)==TRUE)
                                     { lambda=-9999
                                     cat("Region=Nan\n")
                                     print(WniVector)
                                     print(Zsort)
                                     print(TimeDataSort)
                                     print(DeltaSort)
                                     }
               }
                         if (lambda==-9999)
                                     Region<-9999
                         LambdaAdjEL<-sum(WniVector^2)/n
                         cat("LambdaAdjEL:")
                         cat(LambdaAdjEL)
                         cat("\n")
                         #LambdaAdjEL<-LambdaAdjEL^2
                         return(Region,LambdaAdjEL)

 }


#/*******************Examples of how the simulation is generated *******/
#*Example 1:  2000 simulations with Censor Rate = 70, Sample Size=15
df<-1
n<-15
mu<-.66
censorProp<-.70
NumSimulations<-2000
trueBeta<-1
```

```
runSimulation(df,n,mu,censorProp,NumSimulations,trueBeta)

#*Example 2:  2000 simulations with Censor Rate = 10%, Sample Size=15

df<-1
n<-15
mu<-5
censorProp<-.10
NumSimulations<-2000
trueBeta<-1

runSimulation(df,n,mu,censorProp,NumSimulations,trueBeta)

#*Example 3:  2000 simulations with Censor Rate = 10%, Sample Size=30

df<-1
n<-30
mu<-5
censorProp<-.10
NumSimulations<-2000
trueBeta<-1
```