2016

Statistical Inference on Trimmed Means, Lorenz Curves, and Partial Area Under ROC Curves by Empirical Likelihood Method

Yumin Zhao
University of Kentucky, zyyztwo@gmail.com
Digital Object Identifier: https://doi.org/10.13023/ETD.2016.512

STUDENT AGREEMENT:

I represent that my thesis or dissertation and abstract are my original work. Proper attribution has been given to all outside sources. I understand that I am solely responsible for obtaining any needed copyright permissions. I have obtained needed written permission statement(s) from the owner(s) of each third-party copyrighted matter to be included in my work, allowing electronic distribution (if such use is not permitted by the fair use doctrine) which will be submitted to UKnowledge as Additional File.

I hereby grant to The University of Kentucky and its agents the irrevocable, non-exclusive, and royalty-free license to archive and make accessible my work in whole or in part in all forms of media, now or hereafter known. I agree that the document mentioned above may be made available immediately for worldwide access unless an embargo applies.

I retain all other ownership rights to the copyright of my work. I also retain the right to use in future works (such as articles or books) all or part of my work. I understand that I am free to register the copyright to my work.

REVIEW, APPROVAL AND ACCEPTANCE

The document mentioned above has been reviewed and accepted by the student's advisor, on behalf of the advisory committee, and by the Director of Graduate Studies (DGS), on behalf of the program; we verify that this is the final, approved version of the student's thesis including all changes required by the advisory committee. The undersigned agree to abide by the statements above.

Yumin Zhao, Student

Dr. Mai Zhou, Major Professor

Dr. Constance Wood, Director of Graduate Studies

STATISTICAL INFERENCE ON TRIMMED MEANS, LORENZ CURVES, AND PARTIAL AREA UNDER ROC CURVES BY EMPIRICAL LIKELIHOOD METHOD

DISSERTATION

A dissertation submitted in partial fulfillment of

the requirements for the degree of Doctor of

Philosophy in the College of Arts and Sciences

at the University of Kentucky

By

Yumin Zhao

Lexington, Kentucky

Director: Dr. Mai Zhou, Professor of Statistics

Lexington, Kentucky

2016

ABSTRACT OF DISSERTATION

STATISTICAL INFERENCE ON TRIMMED MEANS, LORENZ CURVES, AND PARTIAL AREA UNDER ROC CURVES BY EMPIRICAL LIKELIHOOD METHOD

Traditionally the inference on trimmed means, Lorenz Curves, and partial AUC (pAUC) under ROC curves have been done based on the asymptotic normality of the statistics. Based on the theory of empirical likelihood, in this dissertation we developed novel methods to do statistical inferences on trimmed means, Lorenz curves, and pAUC. A common characteristic among trimmed means, Lorenz curves, and pAUC is that their inferences are not based on the whole set of samples. Qin and Tsao (2002), Qin et al. (2013), and Qin et al. (2011) recently published their researches on the inferences of trimmed means, Lorenz curves, and pAUC based on empirical likelihood method, where they treated the cutting points in the samples fixed at the sample quantiles. They concluded that the limiting distributions of the empirical likelihood tests had scaled chi-square distributions under the null hypotheses. In our novel empirical likelihood methods, we treat the cutting points as the nuisance parameter(s). We conduct the inferences on trimmed means, Lorenz Curves, and pAUC in two steps. First, we make inferences on the parameter interested (trimmed means, Lorenz curves, or pAUC) and the nuisance parameter(s) (the cutting point(s) in the samples) simultaneously. Then we profile out the nuisance parameter(s) from the test statistics. Under the null hypotheses, the limiting distributions of our empirical likelihood methods are chi-square. We innovate a computational algorithm 'ELseesaw' to accomplish our empirical likelihood method for the inference on pAUC. Eventually, we contribute a R package to implement our empirical likelihood inferences on trimmed means, Lorenz curves, and pAUC. The R package we have developed can be downloaded free-of-charge on the internet at `http://www.ms.uky.edu/~mai/EmpLik.html`.

KEYWORDS: Empirical Likelihood Ratio Test, Trimmed Means, Lorenz Curves, pAUC of ROC Curves

Author's signature: Yumin Zhao

Date: December 14, 2016

STATISTICAL INFERENCE ON TRIMMED MEANS, LORENZ CURVES, AND PARTIAL AREA UNDER ROC CURVES BY EMPIRICAL LIKELIHOOD METHOD

By
Yumin Zhao

Director of Dissertation: Dr. Mai Zhou

Director of Graduate Studies: Dr. Constance Wood

Date: December 14, 2016

To my dream over the years since I started learning Statistics.

ACKNOWLEDGMENTS

I am very grateful for the insights I received during my dissertation research. First, my Dissertation Chair, Dr. Mai Zhou, an expert in empirical likelihood theories and other statistical fields inspires me on every innovating solution of the dissertation research. In addition, Dr. Zhou provides timely and instructive comments and evaluation at every stage of the dissertation process. I appreciate Dr. Zhou's humble attitude to his students as well. Next, I wish to thank the complete Dissertation Committee: Dr. William Griffith, Dr. Christopher Bollinger, Dr. Yanbing Zheng, Dr. Xiangrong Yin, and Dr. Simon Bonner. I am very thankful for Dr. Bollinger's guidance on the household income data from IPUMS-CPS. Finally, I also want to thank all the professors in the Department of Statistics who taught me in their classes during my Master and Doctoral curriculum, from which I built the theoretical foundation for my dissertation research and my career.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1 Review of Likelihood Ratio Test and Outline of the Dissertation

In this chapter, we briefly review the basics of empirical likelihood ratio test method, which is the fundamental statistical theory that our research is based on.

1.1 Parametric Likelihood Ratio Test

Based on Casella and Berger (2002) pages 374 - 375, parametric likelihood ratio Test is defined as the following.

Suppose $X_1, \cdots, X_n$ is a random sample from a population with pdf or pmf $f(x|\theta)$ (θ may be a vector), the likelihood function is defined as

$$L(\theta|X_1, \cdots, X_n) = L(\theta|X) = \prod_{i=1}^{n} f(x_i|\theta) \tag{1.1}$$

Let Θ denote the entire parameter space. Θ_0 is a subset of Θ and Θ_0^c is the complement of Θ_0 in Θ. The likelihood ratio test statistic for testing the hypotheses

$$H_0\text{: } \theta \in \Theta_0 \text{ versus } H_1\text{: } \theta \in \Theta_0^c \tag{1.2}$$

is formulated as

$$R(X) = \frac{\sup_{\Theta_0} L(\theta|X)}{\sup_{\Theta} L(\theta|X)} \tag{1.3}$$

A likelihood ratio test (LRT) is any test that has a rejection region

$$\{X : R(X) \leq c\}, \tag{1.4}$$

where c is any number in $0 \leq c \leq 1$.

$\hat{\theta}$, a maximum likelihood estimator (MLE), is obtained by maximizing $L(\theta|X)$ over the entire parameter space. In other words, $\hat{\theta}$ is an unrestricted maximizer of $L(\theta|X)$. We define $\hat{\theta}_0$ as the restricted maximizer of $L(\theta|X)$ in the null hypothesis parameter space Θ_0. That is, $\hat{\theta_0}$ maximizes $L(\theta|X)$ over $\theta \in \Theta_0$. Then, the LRT statistics in (1.3) is

$$R(X) = \frac{L(\hat{\theta}_0|X)}{L(\hat{\theta}|X)} \tag{1.5}$$

To define a level α test, the constant c in (1.4) must be chosen so that

$$\sup_{\theta \in \Theta_0} P_\theta(R(X) \leq c) \leq \alpha \tag{1.6}$$

Wilks (1938) showed that if H_0 is true, then $-2 \log R(X)$ has an asymptotic χ^2_p distribution under certain regularity conditions, where p is the number of restrictions imposed on the parameters by H_0. Thus, the constant c in (1.6) is chosen based on χ^2_p.

It is convenient to formulate LRT (1.5) as

$$-2 \log R(X) = 2(\log L(\hat{\theta}|X) - \log L(\hat{\theta}_0|X)) \tag{1.7}$$

From (1.1),

$$\log L(\hat{\theta}|X) = \sum_{i=1}^{n} \log f(x_i|\hat{\theta}) \tag{1.8}$$

$$\log L(\hat{\theta_0}|X) = \sum_{i=1}^{n} \log f(x_i|\hat{\theta}_0) \tag{1.9}$$

Then, the level α test is defined as

$$\sup_{\theta \in \Theta_0} P_\theta(-2 \log R(X) > c_{1-\alpha}) \leq \alpha, \tag{1.10}$$

The level $1 - \alpha$ likelihood confidence set is

$$\{\theta : -2 \log R(X) \leq c_{1-\alpha}\} \tag{1.11}$$

where $c_{1-\alpha}$ is the $1 - \alpha$ quantile of χ^2_p, i.e. $p(\chi^2_p > c_{1-\alpha}) = \alpha$.

The advantages of parametric LRT are as follows:

1. The likelihood (and log likelihood) function is only defined over the parameter space Θ. Consequently, the likelihood ratio confidence set will only ever contain valid values of the parameter, while Wald interval may accommodate invalid values, i.e. values outside of the parameter space.

2. The likelihood ratio set is transformation invariant. That is, we will get the same confidence set for θ from the transformation of set $\{g(\theta) : -2 \log R(X) \leq c_{1-\alpha}\}$ as the one directly from $\{\theta : -2 \log R(X) \leq c_{1-\alpha}\}$, where $g(\cdot)$ is a function.

3. It is not necessary to construct a variance-covariance matrix in order to form a confidence set for a parameter θ.

1.2 Empirical Likelihood Ratio Test

In parametric likelihood methods described in the previous section, we suppose that the joint distribution of all available data has a known form. However, a problem with parametric likelihood inference is that we might not know which parametric distribution family the data derive from. Misspecification of the distribution family may fail the confidence sets and tests.

Empirical likelihood is a nonparametric method of statistical inference, which utilize likelihood methods without having to assume that the data come from a known family of distribution. Empirical likelihood ratio test inherits all the advantages of parametric likelihood ratio test. Thus, empirical likelihood ratio method has been applied in many situations since Owen (1988) extended earlier work of Thomas and Grunkemeier (1975) who employed a nonparametric likelihood ratio idea to construct confidence intervals for the survival function. The review of this section on empirical likelihood ratio method is largely based on the book by Owen (2001) on pages 1 - 74.

Let $X_1, \cdots, X_n \in \mathbb{R}$. The empirical cumulative distribution function (ECDF) of $X_1, \cdots, X_n$ is

$$F_n(x) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{I}(x_i \leq x) \tag{1.12}$$

for $-\infty < x < \infty$.

$$\mathbf{I}(x_i \leq x) = \begin{cases} 1 & \text{if } x_i \leq x \\ \\ 0 & \text{otherwise} \end{cases}$$

From (1.12), we can see that the probability jump for each data point of the empirical distribution is $\frac{1}{n}$.

Assume $X_1, \cdots, X_n$ independent with common CDF F_0, the nonparametric likelihood of the CDF F is

$$L(F) = \prod_{i=1}^{n} (F(X_i) - F(X_i-)), \tag{1.13}$$

where $F(X_i) = \mathbf{P}(X_i \leq x)$, $F(X_i-) = \mathbf{P}(X_i < x)$, and $F(X_i = x) = F(X_i) - F(X_i-)$.

Owen (2001) on page 8 proved that the nonparametric likelihood in (1.13) is maximized by the ECDF in (1.12). That is, the ECDF is the nonparametric MLE (NPMLE) of $\mathbf{F}$.

Thus, nonparametric likelihood ratio is written as

$$R(F) = \frac{L(F)}{L(F_n)} \tag{1.14}$$

To test

$$H_0\text{: } T(F_0) = \theta_0 \text{ versus } H_1\text{: } T(F_0) \neq \theta_0, \tag{1.15}$$

where $T(\cdot)$ is some function of distribution function F and F is a member of a set $\mathcal{F}$

of distribution, we formulate the empirical likelihood function as

$$R(\theta) = \sup\{R(F)|T(F) = \theta,\, F \in \mathcal{F}\} \tag{1.16}$$

When $R(\theta_0) < r_0$, we reject the null hypothesis in (1.15). The empirical likelihood confidence regions of θ are of the form

$$\{\theta|R(\theta) \geq r_0\} \tag{1.17}$$

The constant r_0 for mean type of hypotheses may be chosen using an empirical likelihood theorem (ELT), a nonparametric analogue of Wilks theorem.

Let w_i be the weight that F places on sample X_i. Based on Owen (2001) page 30, the empirical likelihood ratio function for the mean μ is

$$R(\mu) = \max\{\prod_{i=1}^{n} nw_i | \sum_{i=1}^{n} w_i X_i = \mu.\ w_i \geq 0.\ \sum_{i=1}^{n} w_i = 1\} \tag{1.18}$$

and the resulting empirical likelihood confidence region for the mean as

$$C_{r,\,n} = \{\sum_{i=1}^{n} w_i X_i | \prod_{i=1}^{n} nw_i \geq r_0.\ w_i \geq 0.\ \sum_{i=1}^{n} w_i = 1\} \tag{1.19}$$

Empirical Likelihood Theorem (Owen (2001) page 30)

Let $X_1, \cdots, X_n$ be independent random vectors in $\mathbb{R}^d$ ($d \geq 1$) with common distribution F_0 having mean μ_0 and finite variance covariance matrix V_0 of rank $q > 0$. Then $C_{r,\,n}$ is a convex set and $-2\log R(\mu_0)$ converges in distribution to a χ^2_q random

variable as $n \to \infty$

Owen (2001) proved ELT on pages 219 - 222.

1.3 Profile Likelihood Ratio Test

Parametric likelihood ratio test can handle nuisance parameters, that is, parameters that are present in a model but are not of direct inferential interest. The presence of such nuisance parameters does not affect the LRT construction method (see Casella and Berger (2002) page 378).

Let $\theta^T = (\psi^t, \lambda^t)$, ψ is a $p \times 1$ vector of parameters of interest, λ is a $q \times 1$ vector of nuisance parameters. The test statistics for the null hypotheses for ψ is

$$W_p(\psi_0) = 2\{\log L(\hat{\psi}, \hat{\lambda}|X) - \log L(\psi_0, \hat{\lambda}_{\psi_0}|X)\} \tag{1.20}$$

Davison (2003) pages 127 - 128 defined the profile likelihood ratio test as

$$\log L_p(\psi|X) = \max_{\lambda} \log L(\psi, \lambda|X) = \log L(\psi, \hat{\lambda}_{\psi}|X) \tag{1.21}$$

which may be used to form the confidence region for ψ.

Davison (2003) pages 138 - 139 proved that under the null hypotheses of ψ_0, the limiting distribution of $W_p(\psi_0)$ in (1.20) is a chi-square with degree of freedom p. Profile likelihood method is not as widely used in empirical likelihood test as in parametric likelihood test. Similar results about nuisance parameters and profile likelihood for empirical likelihood are studied in Qin and Lawless (1994). In empiri-

cal setting, it can be computationally challenging to optimize a likelihood over some nuisance parameters. In parametric setting, this issue can be avoided by making a quadratic approximation to the log likelihood. The maximization in parametric profile likelihood ratio method sometimes can also be done analytically by differentiation.

1.4 Outline of the Dissertation

In this dissertation research, we incorporate profile likelihood method into empirical likelihood method to conduct inferences on some very practically useful parameters: trimmed means, Lorenz curves, and partial AUC under the ROC curves. A common characteristic among trimmed means, Lorenz curves, and pAUC is that their inferences are not based on the whole set of samples. Quantile function(s) is(are) involved to determine the cutting point(s) of the samples to be included in the inferences on these parameters. Since the true quantile function(s) is(are) unknown, it(they) has(have) to be estimated, which results in a very complex variance estimation for the asymptotic normality of the parameter estimator. In our novel empirical likelihood methods, we treat the cutting points as nuisance parameters. We conduct the inferences on trimmed means, Lorenz curves, and pAUC in two steps. First, we made inferences on the parameter interested (trimmed means, Lorenz curves, or pAUC) and the nuisance parameter(s) (the cutting point(s) in the samples) simultaneously. After the test statistics is obtained, we profile out the nuisance parameter(s) from the test statistics. The limiting distributions of our novel empirical likelihood ratio methods on the inferences of these parameters are the regular chi-square distribution under the null hypotheses. Thus we do not need to formulate the variances for the

estimators of these parameters. Our novel empirical likelihood methods inherit all the advantages of parametric LRT mentioned in Section 1.1 as well. Additionally, our novel empirical likelihood ratio methods are not affected by the original distributions of the data. Thus, our novel empirical likelihood ratio methods do not need to be based on any assumptions of the original distributions.

An outline of the development of our empirical likelihood methods, the computational algorithms, and the programs is as follows:

In Chapter 2, we apply our empirical likelihood method to the one sample cases for the trimmed means and Lorenz Curves. We formulate the empirical likelihood tests on trimmed means and Lorenz curves and prove the limiting distribution of the empirical likelihood tests under the null hypotheses is a chi-square distribution. We also provide QQ chi-square plots to demonstrate the limiting distribution of the empirical likelihood tests on trimmed means and Lorenz curves. Based on simulation data with a symmetric distribution, we compare the estimated population mean by our empirical likelihood method with the ones by other well known robust population mean estimators. In the end of this chapter, we apply our empirical likelihood method on Lorenz Curve to a real data.

In Chapter 3, we apply our empirical likelihood method to the two sample case for pAUC. Besides formulating the empirical likelihood test on pAUC and presenting the theories of empirical likelihood and profile likelihood. We also provide QQ chi-square plots to demonstrate the limiting distribution of the empirical likelihood tests on pAUC and list the comparisons on coverage probabilities and the lengths of confidence intervals among several inference methods of pAUC based on simulation results.

In Chapter 4, we detail our computational algorithm for profile likelihood method and the computational algorithm "ELseesaw" for empirical likelihood method for the two sample case. Algorithm "ELseesaw" is another innovative contribution of this dissertation research, which simplifies the minimization of the empirical likelihood ratio test of two samples to the minimization of one sample empirical likelihood ratio tests. In the end of Chapter 4, we introduce the R package we developed based on our algorithms.

In Chapter 5, we suggest directions that this dissertation could be extended in the future.

In Appendix we list the annotated R-code for the simulations used in the dissertation.

Chapter 2 Statistical Inference on Trimmed Means and Lorenz Curves by Empirical Likelihood Method

2.1 Introduction

The sample mean, as one of the standard estimators of central tendency, is very frequently used because of well established inference methods based on central limit theory. However, it is extremely sensitive to outliers. The trimmed mean, which is computed after the smallest and largest observations are deleted from the sample, in other words, the observations are trimmed at each end, is insensitive to outliers. For a symmetric distribution, the symmetrically trimmed mean is an unbiased estimate of the population mean. Thus, the trimmed mean has been a very popular robust estimator of location parameters. Winsorized Mean is another robust estimator of the location that is relatively insensitive to outliers. The Winsorized mean is computed as the ordinary mean after the k smallest observations are replaced by the $(k+1)$st smallest observation and the k largest observations are replaced by the $(k+1)$st largest observation. Trimmed mean and Winsorized mean are employed to correct the vulnerability of the Student's t test of the sample mean when the population has a symmetric distribution with tails longer than the normal distribution. Many researches about the variance and asymptotic normality of the trimmed mean had been done based on asymptotic variance of Winsorized variance such as Tukey and McLaughlin (1963), Bickel (1965), Dixon and Tukey (1968), Stigler (1973), Caperra

and Rivest (1995). With the variances, the trimmed t test and Winsorized t test are built for the inference of trimmed mean and Winsorized mean, respectively (see Tukey and McLaughlin (1963), Dixon and Tukey (1968)). SAS Institute Inc. (2010) applies this approach for the inference on the trimmed mean and Winsorized mean.

Instead of the above parametric approach, we develop a nonparametric method - empirical likelihood method to do inference on trimmed means. The empirical likelihood method proposed here is based on the trimmed mean defined in equation (2.1) and its estimation in equation (2.2).

For a cumulative distribution F, the theoretical trimmed mean between given quantiles p_1 and p_2 $(0 \leq p_1 < p_2 \leq 1)$ is computed as

$$\mu_T = \int_{\xi(p_1)}^{\xi(p_2)} x dF(x) \tag{2.1}$$

here $\xi(p_i) = F^{-1}(p_i) = \inf\{x : F(x) \geq p_i\}$, $i = 1, 2$.

Let $X_1, X_2, \cdots, X_n$ be a random sample from the distribution F. The trimmed mean based on this sample is calculated as

$$\hat{\mu_T} = \frac{1}{n} \sum_{i=1}^{n} x_i I[t_1 \leq x_i \leq t_2] \tag{2.2}$$

where $t_1 = \hat{F}^{-1}(p_1) = \sup\{t : \hat{F}(t) < p_1\}$; $t_2 = \hat{F}^{-1}(p_2) = \sup\{t : \hat{F}(t) < p_2\}$;

$$I[t_1 \leq x_i \leq t_2] = \begin{cases} 1 & \text{if } t_1 \leq x_i \leq t_2 \\ 0 & \text{otherwise} \end{cases}$$

If the distribution F is symmetric and the trims at the sides are symmetric as well (i.e. $p_1 = 1 - p_2$), the trimmed mean μ_t and population mean μ have a relation as

$$\mu = \frac{\mu_T}{p_2 - p_1}$$

and

$$\hat{\mu} = \frac{\sum_{i=1}^{n} x_i I[t_1 \leq x_i \leq t_2]}{\sum_{i=1}^{n} I[t_1 \leq x_i \leq t_2]}$$

Our empirical likelihood method is not limited to symmetric distributions and symmetric cuts. In other words, our empirical likelihood method works on any distributions and any cut including cut on one side of the ordered samples, which is usually named as the truncated mean. Truncated means actually have many applications in economics, health services research and other fields. For instance, Lorenz Curve is widely used by economist to represent the inequality of wealth distribution since Lorenz (1905). On a Lorenz Curve, the 45^o diagonal line from lower left corner to the upper right corner represents the equality of wealth. $1 - 2 \times$ area under the Lorenz curve is defined as Gini index, which is usually used as the single measure of inequality. Gini index ranges from 0 for complete equality to 1 for complete inequality.

For a distribution of $F(x)$ defined on non-negative x, the generalized Lorenz Curve is defined as (see Gastwirth (1972))

$$GLC(p) = \int_0^{\xi(p)} x dF(x) \qquad \text{for} \qquad 0 \leq p \leq 1 \tag{2.3}$$

where $\xi(p) = F^{-1}(p) = \inf\{x : F(x) \geq p\}$.

The Lorenz Curve is defined as

$$LC(p) = \frac{GLC(p)}{\mu} \tag{2.4}$$

where μ is the mean of $F(\cdot)$.

Gastwirth (1972) had defined and studied the nonparametric lower and upper bounds of the Lorenz curve and Gini index by constructing the tangents to the curve at the points and by straight line connecting the points on the Lorenz Curve. Gastwirth (1972) estimated Gini index by the area under the Lorenz Curve. Beach and Davidson (1983) performed statistical inference on empirical Lorenz Curve based on asymptotic multivariate normality. Bishop et al. and Chakraborti (1994) extended Beach and Davidson's theory to build confidence intervals for several pre-selected points on generalized Lorenz Curves. They needed to estimate the covariance-variance matrix to construct the test statistics.

In this study, we apply empirical likelihood ratio method to the influence of μ_T in equation (2.1), $GLC(p)$ in equation (2.3) and $LC(p)$ in equation (2.4). The empirical likelihood ratio method has been applied in many situations (see Owen (1988)). Compared to parametric methods, the empirical likelihood ratio method is not affected by the original distribution of the data. Thus, the empirical likelihood ratio method does not need to be based on any assumptions of the original distributions. The empirical likelihood method has the following advantages over the traditional parametric method; (1) our method is workable on different original distributions, no

matter symmetric or very skewed distributions. Thus, our method can be applied to the trimmed mean of symmetric population at symmetrically trimming and to Lorenz Curve inference of population under the right skewed distribution with a long right tail at the trimming on right tail only; (2) our method is insensitive to sample size and the breakdown points (studied by Hampel (1985)) as long as there are enough unremoved data; we do not need to formulate the variance since the limiting distribution of our empirical likelihood ratio under the null hypothesis is a chi-square.

Qin and Tsao (2002) applied empirical likelihood ratio method to the trimmed mean inference and Qin et al. (2013) applied empirical likelihood ratio method to the Lorenz Curve inference. They directly used sample quantiles in the test statistics. Their methods proposed scaled chi-square limiting distributions. They derived a very complicated formulas for the scale coefficients and utilized bootstrap procedures to estimate them.

The rest of this chapter is organized as follows: in Section 2.2 we define the empirical likelihood for the trimmed means, generalized Lorenz Curves, and Lorenz Curves and prove that the limiting distribution of the empirical likelihood ratio test is a chi-square distribution if the null hypothesis is true. In Section 2.3, we present several Chi-square QQ plots based on the simulation results from different original distributions. In Section 2,4, we compare the coverage probability and average length of 95% confidence intervals of trimmed means from our empirical likelihood method and from the trimmed t test and winsorized t test that SAS Institute Inc. (2010) utilizes in the robust location estimation. In Section 2.5 we show an application of our inference method of Lorenz Curves on a real data. In Section 2.6, we finalize this

chapter with discussions and conclusions. The computational algorithm and R codes applying the algorithm will be included in Chapter 4. Simulation R codes are listed in the appendix.

2.2 The Empirical Likelihood Ratio Test for the Trimmed Mean/Generalized Lorenz Curves/Lorenz Curves and Its Limiting Distribution

Hypothesis Test on the Trimmed Mean

We test hypotheses on the trimmed mean in two steps. First, we test hypotheses on μ_T, t_1 (p_1 quantile), and t_2 (p_2 quantile) simultaneously. Then, we proile the nuisance parameters t_1 and t_2. We discuss the first step here and the second step at the end of this section.

Based on equation 2.1, we will first simultaneously test

$$H_{00}: \begin{cases} F^{-1}(p_1) = t_1 \\ F^{-1}(p_2) = t_2 \qquad \text{, for some } t_1 < t_2 \\ \int_{t_1}^{t_2} x dF(x) = \mu_T \end{cases} \tag{2.5}$$

The above hypotheses are equivalent to

$$H_{00}: \begin{cases} F(t_1) = p_1 \\ F(t_2) = p_2 \\ \int_{t_1}^{t_2} x dF(x) = \mu_T \end{cases}$$

For the discrete X, we use g1(X), g2(X), and g3(X) to generalize the hypotheses.

$$\begin{cases} g_1(X, t_1) = g_1(X) = I(X < t_1) \\ g_2(X, t_2) = g_2(X) = I(X < t_2) \\ g_3(X, t_1, t_2) = g_3(X) = XI(t_1 \leq X \leq t_2) \end{cases} \tag{2.6}$$

Let $v_1, v_2, \cdots, v_n$ be the probabilities of $X_1, X_2, \cdots, X_n$, respectively, from distribution function $\hat{F}$ ($v_i = \hat{F}(X_i) - \hat{F}(X_i^-)$); $v_i > 0$ and $\sum_{i=1}^n v_i = 1$. The above hypotheses can be written as

$$\begin{cases} \sum_{i=1}^n (g_1(x_i) - p_1)v_i = 0 \\ \sum_{i=1}^n (g_2(x_i) - p_2)v_i = 0 \\ \sum_{i=1}^n (g_3(x_i) - \mu_T)v_i = 0 \end{cases}$$

We can express the simultaneous equalities of the hypotheses using vectors as follows:

$$\sum_{i=1}^{n} (\mathbf{g}(x_i) - \theta)v_i = \mathbf{0} \tag{2.7}$$

where $(\mathbf{g}(X))^T = (g_1(X),\ g_2(X),\ g_3(X))$, $\theta^T = (p_1,\ p_2,\ \mu_T)$, and $\mathbf{0}^T = (0,\ 0,\ 0)$. $(t_1,\ t_2,\ \mu_T)$ are the parameters. t_1 and t_2 are contained in $g_1(x)$, $g_2(x)$, and $g_3(x)$.

Based on Owen (1988), the empirical distribution function of the above samples $\hat{F}_n(t) = \frac{1}{n}\sum_{i=1}^n I[x_i \leq t]$ is often considered a nonparametric maximum likelihood

estimate of F and the empirical likelihood ratio function is defined as

$$R(\hat{F}) = \frac{L(\hat{F})}{L(\hat{F}_n)} = n^n \prod_{i=1}^{n} v_i$$

The logarithm of the empirical likelihood ratio under constrains is

$$\log R(t_{1_0},\ t_{2_0},\ \mu_{T_0}) = \sup_{v_i}\{\sum_{i=1}^{n} \log n v_i \colon\ v_i > 0,\ \sum_{i=1}^{n} v_i = 1,\ \sum_{i=1}^{n} (\mathbf{g}(x_i) - \theta) v_i = \mathbf{0}\} \tag{2.8}$$

To calculate the sup in (2.8), we use Lagrangian Multiplier as usual (see Owen (1988) and Zhou (2016)). The Lagrangian function for constrained logarithm of the empirical likelihood ratio function is

$$G(v_i) = n \log n + \sum_{i=1}^{n} \log v_i + \gamma \left(\sum_{i=1}^{n} v_i - 1 \right) - n\lambda^T \sum_{i=1}^{n} (\mathbf{g}(x_i) - \theta) v_i \tag{2.9}$$

To maximize the Lagrangian function, let

$$\frac{\partial G}{\partial v_i} = \frac{1}{v_i} + \gamma - n\lambda^T (\mathbf{g}(x_i) - \theta) = 0 \tag{2.10}$$

and,

$$\sum_{i=1}^{n} (v_i \frac{\partial G}{\partial v_i}) = n + \gamma \sum_{i=1}^{n} v_i + n\lambda^T \sum_{i=1}^{n} (v_i (\mathbf{g}(x_i) - \theta)) = 0 \tag{2.11}$$

from (2.11), we have $\gamma = -n$, substitute it to (2.10), then

$$v_i = \frac{1}{n\,(1 + \lambda^T (\mathbf{g}(x_i) - \theta))} \tag{2.12}$$

Thus,

$$\frac{1}{n}\sum_{i=1}^{n}\frac{\mathbf{g}(x_i)-\theta}{1+\lambda^T(\mathbf{g}(x_i)-\theta)}=\mathbf{0} \tag{2.13}$$

By Taylor expansion about $\lambda=\mathbf{0}$, we have,

$$\frac{1}{n}\sum_{i=1}^{n}\Big((\mathbf{g}(x_i)-\theta)-(\mathbf{g}(x_i)-\theta)(\mathbf{g}(x_i)-\theta)^T\lambda+o(\lambda)\Big)=0 \tag{2.14}$$

Thus,

$$\lambda\approx\frac{\bar{\mathbf{g}}(X)-\theta}{n^{-1}\sum_{i=1}^{n}(\mathbf{g}(x_i)-\theta)(\mathbf{g}(x_i)-\theta)^T} \tag{2.15}$$

Theorem 2.1.a Suppose $X_1, X_2, \cdots, X_n$ be a random sample from the distribution F with $E(X^2)<\infty$, furthermore, the density function f of F is positive and continuous at t_1 and t_2 corresponding to the p_1 and p_2 quantile respectively, $0<p_1<p_2<1$. Under the hypotheses (2.5), the limiting distribution of $-2\log R(t_{1_0},\ t_{2_0},\ \mu_{T_0})$ is a chi-square with degree of freedom 3.

Proof:

Substitute (2.12) to the formula for $-2\log R(t_{1_0},\ t_{2_0},\ \mu_{T_0})$, we have

$$-2\log R(t_{1_0},\ t_{2_0},\ \mu_{T_0})=2\sum_{i=1}^{n}\log\big(1+\lambda^T(\mathbf{g}(x_i)-\theta)\big)$$

$$\approx 2\sum_{i=1}^{n}\left(\lambda^T(\mathbf{g}(x_i)-\theta)-\frac{1}{2}\,\lambda^T(\mathbf{g}(x_i)-\theta)(\mathbf{g}(x_i)-\theta)^T\lambda\right)$$

$$\approx 2n\lambda^T(\bar{\mathbf{g}}(X)-\theta)-\lambda^T\sum_{i=1}^{n}(\mathbf{g}(x_i)-\theta)(\mathbf{g}(x_i)-\theta)^T\lambda \tag{2.16}$$

Substitute (2.15) to (2.16),

$$\approx n^2 \frac{(\bar{\mathbf{g}}(X)-\theta)^T(\bar{\mathbf{g}}(X)-\theta)}{\sum_{i=1}^{n}(\mathbf{g}(x_i)-\theta)(\mathbf{g}(x_i)-\theta)^T} \tag{2.17}$$

By central limit theorem, (2.17) $\longrightarrow^d \chi^2_3$.

It follows from Theorem 2.1.a that for any $0 < \alpha < 1$ an empirical likelihood confidence region for $\tau = (t_1, t_2, \mu_T)$ with an asymptotic coverage probability $1-\alpha$ is given by $\{\tau | -2\log R(\tau) < c_{1-\alpha}\}$ where $c_{1-\alpha}$ is defined such as $P(\chi^2_3 > c_{1-\alpha}) = \alpha$.

Next step of our approach will be to profile the t_1 and t_2 out of the likelihood ratio test. But due to the similarity of the profile empirical likelihood ratio test among trimmed mean, Generalized Lorenz Curve, and Lorenz Curve, we discuss the profile empirical likelihood ratio test at the end of this section after we address the first step of hypothesis tests on the Generalized Lorenz Curve and Lorenz Curve.

Hypothesis Test on the Generalized Lorenz Curve

Similar to the hypothesis test on the trimmed mean, the hypothesis test on the generalized Lorenz Curve is done in two steps. We discuss the first step here.

The hypotheses about the generalized Lorenz Curve $GLC(p)$ and t (the p quantile) are expressed as

$$H_{00}: \begin{cases} F^{-1}(p) = t \\ \int_0^t x dF(x) = GLC_p \end{cases}, \text{ for some } 0 < t \tag{2.18}$$

The above hypotheses are equivalent to

$$H_{00}: \begin{cases} F(t) = p \\ \int_0^t x dF(x) = GLC_p \end{cases}$$

For the discrete X, we use h1(X), and h2(X) to generalize the hypotheses.

$$\begin{cases} h_1(X,t) = h_1(X) = I(X < t) \\ h_2(X,t) = h_2(X) = XI(0 \leq X \leq t) \end{cases}$$

Let $v_1, v_2, \cdots, v_n$ be the probabilities of $X_1, X_2, \cdots, X_n$, respectively, from distribution function $\hat{F}$ ($v_i = \hat{F}(X_i) - \hat{F}(X_i^-)$); $v_i > 0$ and $\sum_{i=1}^n v_i = 1$. The above hypotheses can be written as

$$\begin{cases} \sum_{i=1}^n (h_1(x_i) - p)v_i = 0 \\ \sum_{i=1}^n (h_2(x_i) - GLC_p)v_i = 0 \end{cases}$$

We can express the simultaneous equalities of the hypotheses using vectors as follows:

$$\sum_{i=1}^n (\mathbf{h}(x_i) - \theta)v_i = \mathbf{0} \tag{2.19}$$

where $(\mathbf{h}(X))^T = (h_1(X),\ h_2(X))$, $\theta^T = (p,\ GLC_p)$, and $\mathbf{0}^T = (0,\ 0)$. $(t,\ GLC_p)$ are the parameters. t is contained in $h_1(x)$ and $h_2(x)$.

The empirical likelihood ratio function of the hypotheses test on the generalized

Lorenz is defined as

$$R(\hat{F}) = \frac{L(\hat{F})}{L(\hat{F}_n)} = n^n \prod_{i=1}^{n} v_i \tag{2.20}$$

The logarithm of the empirical likelihood ratio under constrains is

$$\log R(t_0,\ GLC_{p_0}) = \sup_{v_i}\{\sum_{i=1}^{n} \log n v_i \colon\ v_i > 0,\ \sum_{i=1}^{n} v_i = 1,\ \sum_{i=1}^{n} (\mathbf{h}(x_i) - \theta) v_i = \mathbf{0}\} \tag{2.21}$$

Follow the same derivation steps as from (2.9) to (2.15), we have

$$v_i = \frac{1}{n\,(1 + \lambda^T(\mathbf{h}(x_i) - \theta))} \tag{2.22}$$

and

$$\lambda \approx \frac{\bar{\mathbf{h}}(X) - \theta}{n^{-1} \sum_{i=1}^{n} (\mathbf{h}(x_i) - \theta)(\mathbf{h}(x_i) - \theta)^T} \tag{2.23}$$

Theorem 2.1.b Suppose $X_1, X_2, \cdots, X_n$ be a random sample from the distribution F with $E(X^2) < \infty$, furthermore, the density function f of F is positive and continuous at t (the p quantile), $0 < p < 1$. Under the hypotheses (2.18), the limiting distribution of $-2 \log R(t_0,\ GLC_{p_0})$ is a chi-square with degree of freedom 2.

The proof of Theorem 2.1.b is the same as the proof of Theorem 2.1.a.

Hypothesis Test on the Lorenz Cure

The hypothesis test on the Lorenz Curve is accomplished in two steps as well. Here we discuss the first step.

Based on the definition in (2.4), testing the hypothesis $LC(p) = LC_p$ is equivalent

to testing $GLC(p) = \mu LC_p$. The hypotheses about the Lorenz Curve $LC(p)$ and t (the p quantile) are expressed as

$$H_{00} : \begin{cases} F^{-1}(p) = t \\ \int_0^t x dF(x) = \mu LC_p \end{cases} \text{, for some } 0 < t \tag{2.24}$$

The above hypotheses are equivalent to

$$H_{00} : \begin{cases} F(t) = p \\ \int_0^t x dF(x) - LC_p \int_0^\infty x dF(x) = 0 \end{cases}$$

For the discrete X, we use r1(X), and r2(X) to generalize the hypotheses.

$$\begin{cases} r_1(X,t) = r_1(X) = I(X < t) \\ r_2(X,t) = r_2(X) = XI(0 \leq X \leq t) - LC_p X \end{cases}$$

Let $v_1, v_2, \cdots, v_n$ be the probabilities of $X_1, X_2, \cdots, X_n$, respectively, from distribution function $\hat{F}$ ($v_i = \hat{F}(X_i) - \hat{F}(X_i^-)$); $v_i > 0$ and $\sum_{i=1}^n v_i = 1$. The above hypotheses can be written as

$$\begin{cases} \sum_{i=1}^n (r_1(x_i) - p) v_i = 0 \\ \sum_{i=1}^n r_2(x_i) v_i = 0 \end{cases}$$

We can express the simultaneous equalities of the hypotheses using vectors as

follows:

$$\sum_{i=1}^{n}(\mathbf{r}(x_i)-\theta)v_i = \mathbf{0} \tag{2.25}$$

where $(\mathbf{r}(X))^T = (r_1(X),\ r_2(X))$, $\theta^T = (p,\ 0)$, and $\mathbf{0}^T = (0,\ 0)$. $(t,\ LC_p)$ are the parameters. t is contained in $r_1(x)$ and $r_2(x)$, and LC_p is contained in $r_2(x)$.

The logarithm of the empirical likelihood ratio test on the Lorenz curve under constrains is

$$\log R(t_0,\ LC_{p_0}) = \sup_{v_i}\{\sum_{i=1}^{n}\log nv_i\colon\ v_i>0,\ \sum_{i=1}^{n}v_i=1,\ \sum_{i=1}^{n}(\mathbf{r}(x_i)-\theta)v_i=\mathbf{0}\} \tag{2.26}$$

Follow the same derivation steps as from (2.9) to (2.15), we have

$$v_i = \frac{1}{n\left(1+\lambda^T(\mathbf{r}(x_i)-\theta)\right)} \tag{2.27}$$

and

$$\lambda \approx \frac{\bar{\mathbf{r}}(X)-\theta}{n^{-1}\sum_{i=1}^{n}(\mathbf{r}(x_i)-\theta)(\mathbf{r}(x_i)-\theta)^T} \tag{2.28}$$

Theorem 2.1.c Suppose $X_1, X_2, \cdots, X_n$ be a random sample from the distribution F with $E(X^2) < \infty$, furthermore, the density function f of F is positive and continuous at t (the p quantile), $0 < p < 1$. Under the hypotheses (2.24), the limiting distribution of $-2\log R(t_0,\ LC_{p_0})$ is a chi-square with degree of freedom 2.

The proof of Theorem 2.1.c is the same as the proof of Theorem 2.1.a if $\mathbf{g}(x_i)-\theta$ is replaced by $\mathbf{r}(x_i)-\theta$.

Profiled Empirical Likelihood Ratio Test

In reality, we are just interested in the inference on μ_T, GLC_p or $LC(p)$ and not the nuisance parameter(s), which can be accomplished by profile out the nuisance parameters t_1 and t_2 in the empirical likelihood function (2.8), where t_1 and t_2 are included in the function $g(\cdot)$ or profile out the nuisance parameter t in the empirical likelihood function (2.21) for GLC_p or (2.26) for $LC(p)$, where t is included in function $h(\cdot)$ or $r(\cdot)$.

Profile likelihood statistic has been used in parametric likelihood ratio test (See review in Chapter 1).

We apply the profile likelihood statistic method to (2.8), (2.21), or (2.26) and we have

$$-2\log R(\theta) = -2\max_{\mathbf{t}} \log R(\mathbf{t},\theta) = \min_{\mathbf{t}}(-2\log R(\mathbf{t},\theta)) = -2\log R(\hat{t},\theta) \qquad (2.29)$$

where $\hat{\mathbf{t}} = (\hat{t_1},\hat{t_2})$ maximizes $\log R(\theta) = \log R(\mathbf{t},\mu_{T_0})$ with respect to $\mathbf{t} = (t_1,t_2)$ for the hypotheses test of the trimmed mean and $\mu = \mu_{T_0}$; $\hat{\mathbf{t}} = \hat{t}$ maximizes $\log R(\theta) = \log R(t, GLC_{p_0})$, or $\log R(t, LC_p)$ with respect to t for the hypotheses test of the generalized Lorenz Curve or Lorenz Curve and $\mu = GLC_{p_0}$, or LC_p.

Theorem 2.2 Under the same condition as Theorem 2.1.a, Theorem 2.1.b, and Theorem 2.1.c, the limiting distribution of the above defined profile empirical likelihood ratio test $-2\log R(\theta_0)$ is a chi-square with degree of freedom 1. Here $\theta_0 = \mu_{T0}$ or $\theta_0 = GLC_{p0}$ or $\theta_0 = LC_p$.

To prove Theorem 2.2, we start from the following Lemma.

Lemma 2.2.1. *Suppose* $\hat{\theta}_n = (\hat{\theta}_{1_n}, \hat{\theta}_{2_n}, \ldots, \hat{\theta}_{m_n})$ *are* n *sequences of* $m \times 1$ *random vectors. Assume* $\sqrt{n}(\hat{\theta}_n - \mu_0)$ *converges to a multivariate normal distribution with mean* $\boldsymbol{0}$ *and a* $m \times m$ *variance* Σ*, where* $\mu_0 = (\mu_{1_0}, \mu_{2_0}, \ldots, \mu_{m_0})$*. From the above, we have*

$$Q(\mu_0) = n(\hat{\theta}_n - \mu_0)\Sigma^{-1}(\hat{\theta}_n - \mu_0)^T \longrightarrow \chi^2_{(m)}$$

Then,

$$\min_{\mu_{1_0}, \mu_{2_0}, \ldots, \mu_{p_0}} Q(\mu_{1_0}, \mu_{2_0}, \ldots, \mu_{m_0}) \longrightarrow \chi^2_{(m-p)} \text{ for } p < m$$

Proof. To simplify the symbols, we let $A = \Sigma^{-1}$ and $A = \begin{pmatrix} A_{11} & A_{12} \\ A_{12}^T & A_{22} \end{pmatrix}$, here A_{11} is a $p \times p$ matrix; A_{12} is a $p \times (m-p)$ matrix; A_{22} is a $(m-p) \times (m-p)$ matrix. Let $X_1 = (\hat{\theta}_{1_n} - \mu_{1_0}, \hat{\theta}_{2_n} - \mu_{2_0}, \ldots, \hat{\theta}_{p_n} - \mu_{p_0})$ and $X_2 = (\hat{\theta}_{(p+1)_n} - \mu_{(p+1)_0}, \hat{\theta}_{(p+2)_n} - \mu_{(p+2)_0}, \ldots, \hat{\theta}_{m_n} - \mu_{m_0})$. Then,

$$\begin{aligned} Q &= n(X_1, X_2) \begin{pmatrix} A_{11} & A_{12} \\ A_{12}^T & A_{22} \end{pmatrix} \begin{pmatrix} X_1^T \\ X_2^T \end{pmatrix} \\ &= n(X_1 A_{11} X_1^T + X_2 A_{12}^T X_1^T + X_1 A_{12} X_2^T + X_2 A_{22} X_2^T) \end{aligned} \tag{2.30}$$

To minimize Q over $\mu_{1_0}, \mu_{2_0}, \ldots, \mu_{p_0}$, we take partial derivatives of Q with regard to $\mu_{1_0}, \mu_{2_0}, \ldots, \mu_{p_0}$ and let the partial derivatives equal to $\mathbf{0}$ (a vector of length p), then

$$A_{11} X_1^T + A_{12} X_2^T = 0$$

Solve this linear equations, we have

$$X_1^T = -A_{11}^{-1}A_{12}X_2^T \tag{2.31}$$

Substitute equation 2.31 to equation 2.30,

$$\begin{aligned}\min_{\mu_{1_0},\,\mu_{2_0},\,\ldots,\,\mu_{p_0}} Q &= n(X_2A_{12}^TA_{11}^{-1}A_{11}A_{11}^{-1}A_{12}X_2^T - X_2A_{12}^TA_{11}^{-1}A_{12}X_2^T \\ &\quad - X_2A_{12}^TA_{11}^{-1}A_{12}X_2^T + X_2A_{22}X_2^T) \\ &= nX_2(A_{22} - A_{12}^TA_{11}^{-1}A_{12})X_2^T\end{aligned} \tag{2.32}$$

The variance matrix $\Sigma = A^{-1} = \begin{pmatrix} B_{11} & B_{12} \\ B_{12}^T & B_{22} \end{pmatrix}$, B_{22} is the variance matrix of $\sqrt{n}X_2$ and $B_{22} = (A_{22} - A_{12}^TA_{11}^{-1}A_{12})^{-1}$.

$$\mathrm{Var}(\sqrt{A_{22} - A_{12}^TA_{11}^{-1}A_{12}}\sqrt{n}X_2) = (A_{22} - A_{12}^TA_{11}^{-1}A_{12})\mathrm{Var}(\sqrt{n}X_2) = \mathbf{I}_{m-p}$$

Therefore, we prove that

$$\min_{\mu_{1_0},\,\mu_{2_0},\,\ldots,\,\mu_{p_0}} Q(\mu_{1_0},\,\mu_{2_0},\,\ldots\,,\,\mu_{m_0}) \longrightarrow \chi^2_{(m-p)} \text{ for } p < m$$

□

From the proof of Theorem 2.1.a, we know that $-2\log(\theta_0)$ in (2.8) can be written as 2.17, from which and the Lemma 2.2.1, we prove Theorem 2.2.

It follows from Theorem 2.2 that for any $0 < \alpha < 1$ an empirical likelihood

confidence interval for $\theta = \mu_T$ or $\theta = GLC_{p0}$ or $\theta = LC_p$ with an asymptotic coverage probability $1 - \alpha$ is given by $\{\theta| - 2\log R(\theta) < c_{1-\alpha}\}$ where $c_{1-\alpha}$ is defined such as $P(\chi_1^2 > c_{1-\alpha}) = \alpha$.

2.3 Simulation: Chi Square QQ Plots

Section 2.2 indicates that the empirical likelihood ratio test on the trimmed mean, the generalized Lorenz Curve, or Lorenz Curve are achieved in two steps. First, we compute the test statistics of the hypothesis (2.5), (2.18) or (2.24) based on equations 2.12 and 2.15. Here we use the function 'el.test' in the R package of 'emplik' by Zhou and Yang (2014) to compute the test statistics of the hypothesis (2.5), (2.18) or (2.24). At this step, we apply the smoothing function discussed in Chapter 4 to the indicators in the test statistics. We will list the smoothing parameters used in the simulation studies. At the second step, we profile the $\mathbf{t}$ in the empirical likelihood ratio test by minimizing the test statistics obtained from Step 1 among a set of quantiles in the samples. The detail algorithm and R codes are given in Chapter 4. The R codes for the QQ plots are provided in the Appendix.

Simulation results in Figures 2.1, 2.2, and 2.3 have shown that the test statistics from Step 1 is a Chi-square with degree of freedom 3 for the trimmed mean under the null hypotheses and a Chi-square with degree of freedom 2 for the generalized Lorenz Cure under the null hypotheses and for the Lorenz Cure under the null hypotheses. The profiled empirical likelihood ratio test for the trimmed mean, the generalized Lorenz Curve, and Lorenz Curve is a chi-square distribution with one degree of freedom when the null hypothesis is true. The smoothing parameter used

in this simulation is equal to $1/n$, n is the sample size.

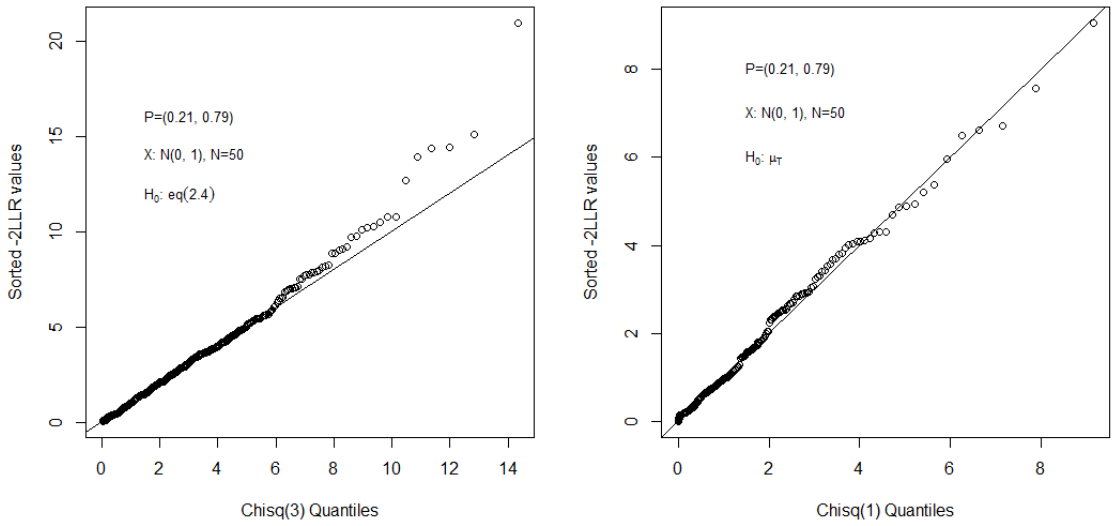

Figure 2.1: Chi Square QQ Plots for the Empirical Likelihood Test on trimmed means

2.4 Simulation: Confidence Intervals and Coverage Probabilities

To evaluate the performance of our empirical likelihood method for trimmed mean inference, we conducted a simulation study to compare the coverage probability and length of confidence interval of estimated population means from our empirical likelihood method (EL) and the ones from the trimmed t and Winsorized t that SAS Institute Inc. (2010) uses. The simulation samples are generated from logistic distribution with location parameter 0 and scale parameter 1. The logistic distribution is symmetric and has longer tails than normal distribution, which is the appropriate distribution for the trimmed t and Winsorized t tests. We generated 1000 samples at each sample size listed on the Table 2.1 in R and calculated the mean coverage probability and average length of 95% confidence intervals of the estimated popula-

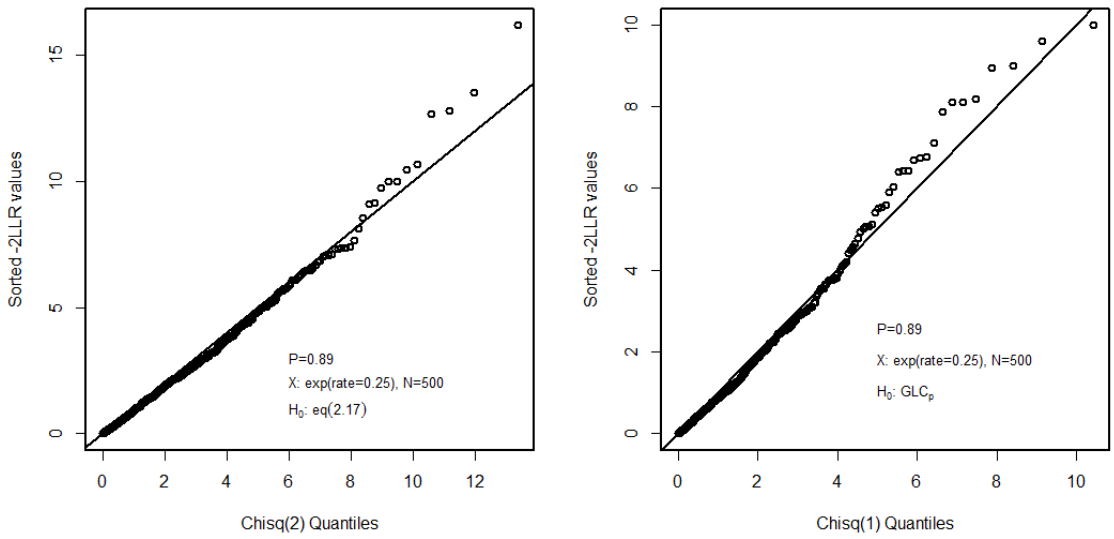

Figure 2.2: Chi Square QQ Plots for the Empirical Likelihood Test on generalized Lorenz Curve

tion means by our empirical likelihood method in R. The trimmings are at $p_1 = 0.1$ and $p_1 = 0.9$. The R codes of this simulation study are provided in the appendix. The 1000 random samples at each sample size generated in R and used by the EL method were then read into SAS. From these same random samples EL method used in R, the mean coverage probability and average length of 95% confidence intervals of the estimated population means by trimmed t and Winsorized t are calculated by proc univariate in SAS at the same trimmings. The SAS codes are provided in the appendix as well. The results in Table 2.1 shows that the lengths of confidence interval of EL method are slightly shorter than the ones from trimmed t and Winsorized t and the coverage probabilities of these three methods are about the same. The smoothing parameter used in this simulation study equals to $n^{-1/2}$, n is the sample size.

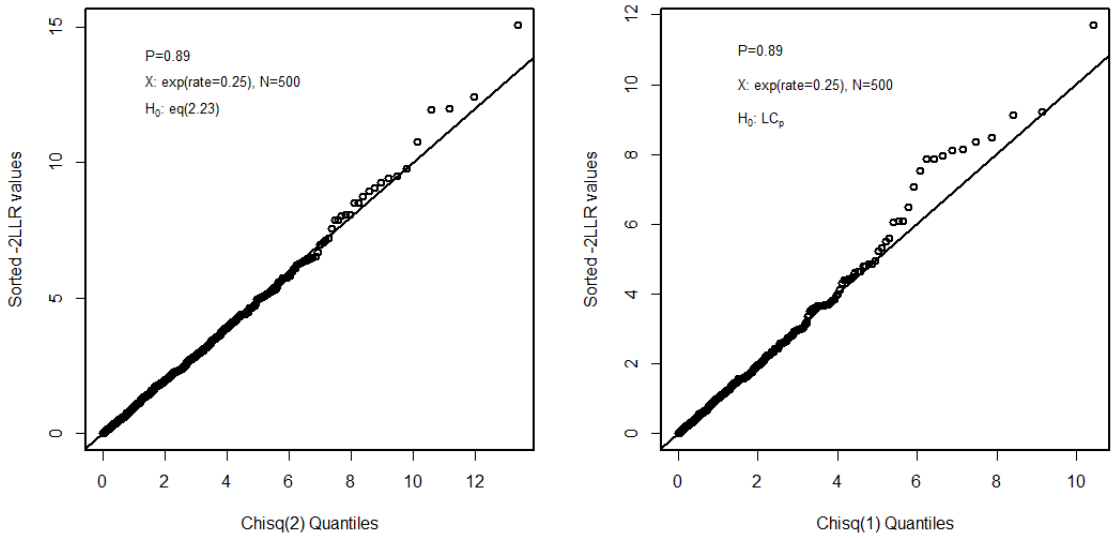

Figure 2.3: Chi Square QQ Plots for the Empirical Likelihood Test on Lorenz Curve

Table 2.1: Coverage Probability and Average Length of 95% Confidence Intervals of estimated population means by empirical likelihood trimmed mean (EL), trimmed t, and Winsorized t methods

Sample size	Method	Coverage Probability	Average Length
40	trimmed t	0.947	1.111
	Winsorized t	0.941	1.115
	EL	0.940	1.093
60	trimmed t	0.948	0.899
	Winsorized t	0.939	0.901
	EL	0.945	0.894
80	trimmed t	0.959	0.774
	Winsorized t	0.953	0.775
	EL	0.958	0.770

2.5 Lorenz Curves Based on Real Data

We applied our empirical likelihood inference method on Lorenz Curves to the total household income (variable 'HHINCOME') from IPUMS-CPS, University of Minnesota, www.ipums.org. IPUMS-CPS includes data of the Current Population Sur-

vey (CPS) since 1962. HHINCOME is from the the Annual Social and Economic Supplement (ASEC) survey, which reports the total money income during the previous calendar year of all adult household members. We selected HHINCOME in 2015 from the states of Indiana and Kentucky in our analyses. Even though our inference method on Lorenz Curves could include negative income, we excluded one record with HHINCOME = -9999 from Indiana since this negative number is most likely a coded value. The final data included in the analysis has 2053 records with the minimum 0 dollar and the maximum $2,204,000; of which 1140 records are from Indiana and 913 records are from Kentucky. We calculated the Lorenz Curves and 95% confidence intervals at every deciles of samples in each state from the low income to high income using our empirical likelihood method. By connecting all the estimated points of each state, we plotted the Lorenz Curves for each state as shown in Figure 2.4. The error bar at each point of the plots is the 95% confidence interval of the estimate on each point. The Lorenz Curve of Indiana is completely above the one of Kentucky, which indicates that the equality of Indiana is better than that of Kentucky. However, we can not know if the difference on Lorenz curves between Indiana and Kentucky is statistically significant since we did the statistical inference neither on the whole Lorenz Curve nor on the Gini index.

2.6 Discussion and Conclusions

Empirical likelihood method for the inference on the trimmed means, generalized Lorenz Curves, and Lorenz Curves does not need to estimate the variance of the estimate since the limiting distribution of test statistics is a chi-square with one

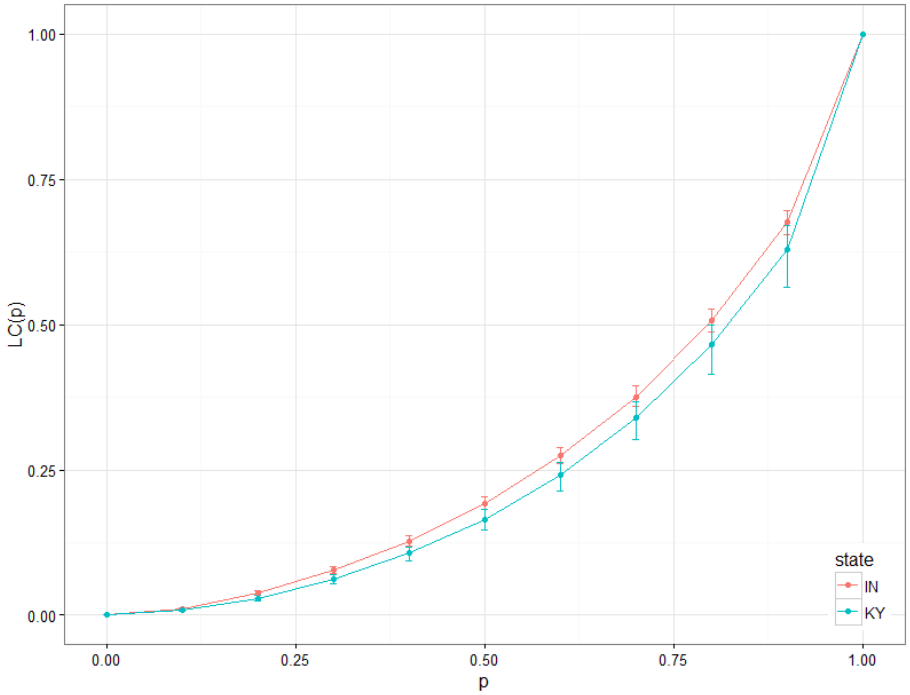

Figure 2.4: Lorenz Curves for States Indiana and Kentucky Based on Household Incomes from IPUMS-CPS, University of Minnesota, www.ipums.org. The error bar at each point is the 95% confidence interval by empirical likelihood method at each estimated point

degree of freedom under the null hypotheses. Empirical likelihood method for the trimmed mean inference provides comparable inferential results on the population mean estimate with the trimmed t and Winsorized t methods if the samples are from a symmetric distribution with longer tails than normal distribution. Empirical likelihood method on Lorenz Curves provides the inference on the selected points on Lorenz Curves. However, more work need to be done in order that our empirical likelihood method on Lorenz Curves is able to calculate the confidence bands of Lorenz Curves and provide inference on Gini index.

Chapter 3 Statistical Inference on the Partial Area Under ROC Curves by Empirical Likelihood Method

3.1 Introduction

The purpose of diagnostic tests is to confirm the presence of disease and to deny the possibility of the disease in healthy subjects. Ideally such tests correctly identify all patients with the disease (True Positive), and similarly correctly identify all patients who are disease free (True Negative). In other words, a perfect test is never positive in a patient who is disease free (False Positive) and is never negative in a patient who is in fact diseased (False Negative) (see Hajian-Tilaki (2013)). However, a perfect test is hardly found in reality. The accuracy of a diagnostic test with dichotomous outcome (positive/negative test results) can be measured by sensitivity and specificity, which are defined as the probabilities of the test correctly identifying the diseased and non-diseased subjects, respectively. The sensitivity and specificity can be computed across all the possible threshold values of the test results reported on continuous scale. The plot of the sensitivity or the true positive rate (TPR) versus 1-Specificity or the false positive rate (FPR) as the threshold value of the test results is varied is called receiver operating characteristic (ROC) curve (Hajian-Tilaki (2013) and Zweig and Campbell (1993)). The ROC curve was first developed during World War II by electrical engineers and radar engineers for detecting enemy objects and was soon found other uses in psychology, medicine, radiology, biometrics, and is increasingly

applied in machine learning and data mining research.

Let X and Y, with respective distribution functions F and G, be the results of a continuous-scale test for a non-diseased and a diseased subject, respectively. For a given cut-off point c, without loss of generality we assume that a test value greater than c is indicative of the positive test result. Sensitivity or true positive rate is defined as $TPR = p(Y > c) = 1 - G(c)$. 1-Specificity or false positive rate is defined as $FPR = p(X > c) = 1 - F(c)$. The ROC curve $\{1 - F(c), 1 - G(c)\}$ at $FPR = r$ can be express as $ROC(r) = 1 - G(F^{-1}(1 - r))$, where $r = 1 - F(c)$.

The area under a ROC curve (AUC) represents the overall accuracy of a diagnostic test, which can be interpreted as the probability that in a randomly selected pair of diseased and non-diseased subjects, the test value of the diseased subject is higher than that of the non-diseased subject (Hajian-Tilaki (2013) and Hanley and McNeil (1982)). A perfect test has AUC equal to 1.0, which has an ROC curve that passes through the upper left corner where the sensitivity or TPR is 1.0 and the false positive rate is 0; A test not better than a random guess has AUC equal to 0.5 with a 45^o diagonal line from the lower left corner to the upper right corner as the ROC curve. A test with an AUC value approaching 1.0 indicates a high sensitivity and specificity. The AUC of the results X and Y of the above non-diseased and diseased subjects can be represented by $AUC = \int (1 - G(x))dF(x)$ (Hanley and McNeil (1982)). However, AUC of a ROC as a measure of the overall performance of a diagnostic test may not be informative, or even misleading. For example, two diagnostic tests may have the equal AUC but not identical ROCs when the two ROC curves cross (For example ROC curves A and B in Figure 3.1). One test may be better than the other in the

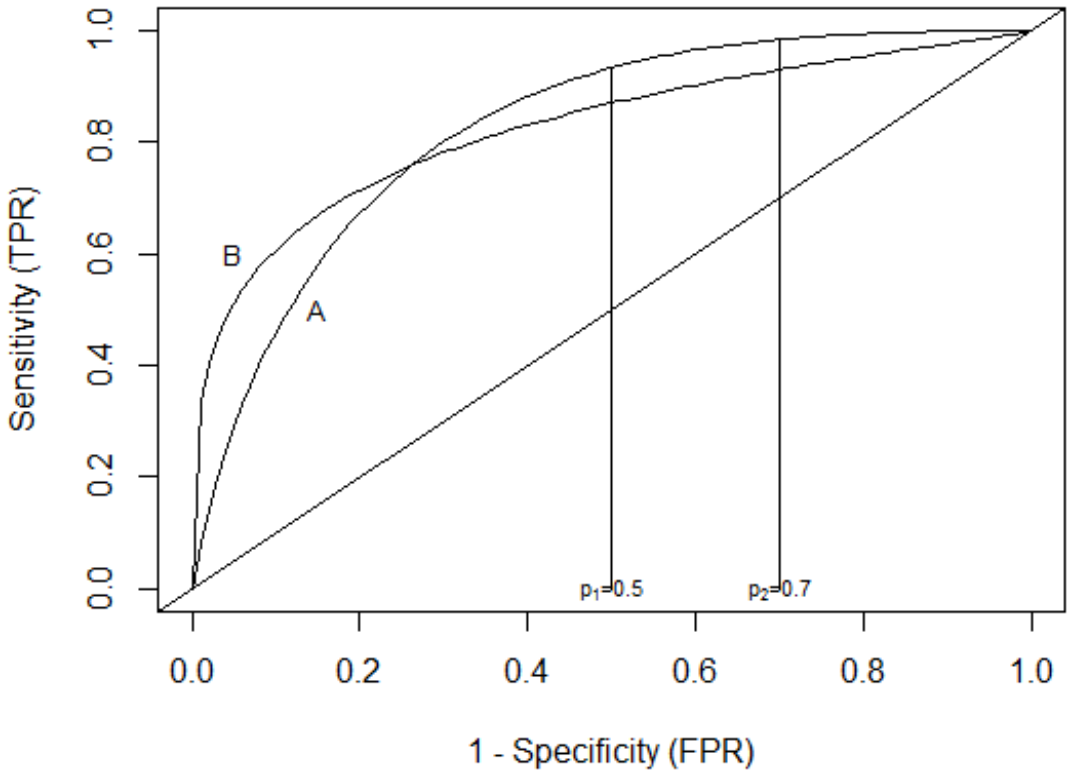

Figure 3.1: Illustration of ROC curves A and B with equal AUCs and unequal pAUCs between $p_1 = 0.5$ and $p_2 = 0.7$

high FPR range; while the other test may be better in the low FPR range. When screening for a high risk disease, good sensitivity is expected even if the FPR is high; however, if screening for a low risk disease with risky subsequent confirmatory tests and/or treatments, a high specificity or low FPR is required. To evaluate two diagnostic tests on a portion of ROC curves, the partial AUC is desirable, which is defined as the area under a ROC curve between two FPRs (see McClish (1989)). Figure 3.1 shows partial AUCs under ROC curves A and B between FPR = 0.5 and FPR = 0.7.

The partial AUC of the results X and Y of the above non-diseased and diseased subjects at $FPR = (p_1, p_2)$, $0 < p_1 < p_2 < 1$, is calculated as

$$pAUC = \int_{\xi(p_2)}^{\xi(p_1)} (1 - G(x))dF(x) \tag{3.1}$$

where $\xi(p_1) = F^{-1}(1 - p_1) = \inf\{x : F(x) \geq 1 - p_1\}$; $\xi(p_2) = F^{-1}(1 - p_2) = \inf\{x : F(x) \geq 1 - p_2\}$.

If the partial AUC is at $FPR = [0, p)$, $0 < p < 1$, then

$$pAUC = \int_{\xi(p)}^{\infty} (1 - G(x))dF(x) \tag{3.1.a}$$

where $\xi(p) = F^{-1}(1 - p) = \inf\{x : F(x) \geq 1 - p\}$.

If the partial AUC is at $FPR = (p, 1]$, $0 < p < 1$, then

$$pAUC = \int_{-\infty}^{\xi(p)} (1 - G(x))dF(x) \tag{3.1.b}$$

where $\xi(p) = F^{-1}(1 - p) = \inf\{x : F(x) \geq 1 - p\}$.

The partial AUC of ROC curves has been studied by different researchers. McClish (1989) first proposed the pAUC assuming binormal data. Y. Jiang (1996) derived the mathematical formation of the pAUC from the conventional binormal model only in a high-sensitivity region. Dodd and Pepe (2003) proposed the non-parametric

estimator for the pAUC.

$$\widehat{pAUC} = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} I(y_j > x_i)I(x_i \in (t_1,t_2)) \tag{3.2}$$

where $X_1, X_2, \cdots, X_m$ are the test results of non-diseased samples and $Y_1, Y_2, \cdots, Y_n$ are the test results of the diseased samples. If the quantiles (t_1,t_2) will not be known, Dodd and Pepe (2003) suggested that empirical quantiles estimates were substituted. $t_1 = \hat{F}^{-1}(1-p_2) = \sup\{t : \hat{F}(t) < 1-p_2\}$; $t_2 = \hat{F}^{-1}(1-p_1) = \sup\{t : \hat{F}(t) < 1-p_1\}$. If the empirical quantile value does not coincide precisely with the desired value, as may happen with small sample sizes, they use linearly interpolated values. They showed the estimator is consistent and asymptotically normal and recommended using the bootstrap to obtain variance estimates. Qin et al. (2011) applied a pseudo empirical likelihood ratio method to the above pAUC estimator. Instead of using the two samples of partial AUC (both diseased and non-diseased samples), they expressed the empirical likelihood test of partial AUC using only the non-diseased sample. Additionally, they directly used sample quantiles of non-diseased sample in the empirical likelihood test. They concluded that the limiting distribution of the test statistic was a scaled chi-square under the null hypothesis. They proposed a very complex bootstrap procedure to estimate the scale constant. Jihnhee Yu (2011) formulated a generalized empirical likelihood test for AUC utilizing both samples, where they incorporated the variance of AUC estimate to the empirical likelihood test statistic. Yang et al. (2016) recently published their study on pAUC at [0, p) by using normal approximation method, jackknife method and jackknife empirical likelihood

(JEL) method. Yang et al. (2016) proposed very complicate variance estimates for normal approximation method and the jackknife method and they built their JEL method upon the asymptotic normality and variance consistency of jackknife pseudo-samples.

In our study, we apply a different empirical likelihood ratio method to the inference of the above pAUC estimator. The empirical likelihood ratio method has been applied in many situations (Owen (1988)). Compared to parametric method, empirical likelihood ratio method is not affected by the original distributions of the data. Thus, empirical likelihood ratio method does not need to be based on any assumptions of the original distributions. We do not need to formulate the variance since the limiting distribution of our empirical likelihood ratio test of the pAUC estimate under the null hypothesis is a regular chi-square.

The rest of this chapter is organized as follows: in Section 3.2 we define the empirical likelihood for the pAUC and show that the limiting distribution of the empirical likelihood ratio test statistics for the pAUC is a chi-square distribution under the null hypothesis. In Section 3.3, we show Chi-square QQ plots of the empirical likelihood ratio test of pAUC from different original distributions. In Section 3.4, we compare the coverage probability and average length of 95% confidence intervals of pAUC from different inference methods. In Section 3.5, we finalize this chapter with discussions and conclusions. The computational algorithm and R codes applying the algorithm will be included in Chapter 4. The simulation R codes are listed in the appendix.

3.2 The Empirical Likelihood Ratio Test for pAUC and Its Limiting Distribution

Suppose we have two independent samples; $x_i, x_2, \cdots, x_m$ from the test results of non-diseased subjects with distribution F; $y_i, y_2, \cdots, y_n$ from the test results of diseased subjects with distribution G.

Hypothesis Test for pAUC at FPR = $(\mathbf{p_1}, \mathbf{p_2})$, $\mathbf{0 < p_1 < p_2 < 1}$

Based on the definition of pAUC (equation (3.1)) in Section 3.1, we will first simultaneously test

$$H_{00}: \begin{cases} F^{-1}(1-p_2) = t_1 \\ F^{-1}(1-p_1) = t_2 \\ \int_{t_1}^{t_2} (1-G(x))dF(x) = pAUC \end{cases} \quad \text{, for some } t_1 < t_2 \tag{3.3}$$

The above hypotheses are equivalent to

$$H_{00}: \begin{cases} 1-F(t_1) = p_2 \\ 1-F(t_2) = p_1 \\ \int_{t_1}^{t_2} (1-G(x))dF(x) = pAUC \end{cases}$$

For the discrete X and Y, we use $g_1(X)$, $g_2(X)$, and $g_3(X,Y)$ to generalize the

hypotheses.

$$\begin{cases} g_1(X, t_1) = g_1(X) = I(X > t_1) \\ g_2(X, t_2) = g_2(X) = I(X > t_2) \\ g_3(X, Y, t_1, t_2) = g_3(X, Y) = I(Y > X)I(t_1 \leq X \leq t_2) \end{cases} \tag{3.4}$$

Let $u_1, u_2, \cdots, u_m$ be the probabilities at $X_1, X_2, \cdots, X_m$, respectively, from distribution function F^* ($u_i = F^*(X_i) - F^*(X_i^-)$); $u_i > 0$ and $\sum_{i=1}^m u_i = 1$. Let $v_1, v_2, \cdots, v_n$ be the probabilities at $Y_1, Y_2, \cdots, Y_n$, respectively, from distribution function G^* ($v_j = G^*(y_j) - G^*(y_j^-)$); $v_j > 0$ and $\sum_{j=1}^n v_j = 1$. The above hypotheses can be written as

$$\begin{cases} \sum_{i=1}^m (g_1(x_i) - p_2)u_i = 0 \\ \sum_{i=1}^m (g_2(x_i) - p_1)u_i = 0 \\ \sum_{i=1}^m \sum_{j=1}^n (g_3(x_i, y_j) - pAUC)u_i v_j = 0 \end{cases}$$

We can express the simultaneous equalities of the hypotheses using vector notation as follows:

$$\sum_{i=1}^m \sum_{j=1}^n (\mathbf{g}(x_i, y_j) - \theta)u_i v_j = \mathbf{0} \tag{3.5}$$

where $(\mathbf{g}(X, Y))^T = (g_1(X),\ g_2(X),\ g_3(X, Y))$, $\theta^T = (p_2,\ p_1,\ pAUC)$, and $\mathbf{0}^T = (0,\ 0,\ 0)$. $(t_1,\ t_2,\ pAUC)$ are the true parameters and t_1 and t_2 are contained in the function $\mathbf{g}(X, Y)$.

The empirical likelihood of the non-diseased sample X and diseased sample Y

with the above constraints is expressed as

$$\begin{aligned}\mathbf{L}(t_1,\ t_2,\ pAUC) = \sup_{u_i,v_j}\{\prod_{i=1}^{m}\prod_{j=1}^{n} u_i v_j\colon\ u_i > 0,\ v_j > 0,\\ \sum_{i=1}^{m} u_i = 1,\ \sum_{j=1}^{n} v_j = 1,\ \sum_{i=1}^{m}\sum_{j=1}^{n}(\mathbf{g}(x_i, y_j) - \theta)u_i v_j = \mathbf{0}\}\end{aligned} \tag{3.6}$$

It is well known that the unconstrained empirical likelihood is maximized at $u_i = \frac{1}{m}$, $i = 1, 2, \cdots, m$ and $v_j = \frac{1}{n}$, $j = 1, 2, \cdots, n$. Thus the logarithm of the empirical likelihood ratio function of the above hypotheses is then written as

$$\begin{aligned}\log R(t_1,\ t_2,\ pAUC) = \log \frac{L(\theta)}{\prod_{i=1}^{m}\frac{1}{m}\prod_{j=1}^{n}\frac{1}{n}} = \sup_{u_i,v_j}\{\sum_{i=1}^{m}\log m u_i + \sum_{j=1}^{n}\log n v_j\colon\\ u_i > 0,\ v_j > 0,\ \sum_{i=1}^{m} u_i = 1,\ \sum_{j=1}^{n} v_j = 1,\ \sum_{i=1}^{m}\sum_{j=1}^{n}(\mathbf{g}(x_i, y_j) - \theta)u_i v_j = \mathbf{0}\}\end{aligned} \tag{3.7}$$

To calculate the sup in (3.7), we use Lagrangian Multiplier as usual (see Owen (1988) and Zhou (2016)). The Lagrangian function for constrained logarithm of the empirical likelihood ratio function is

$$\begin{aligned}G(u_i, v_j) = \sum_{i=1}^{m}\log u_i + \sum_{j=1}^{n}\log v_j + \gamma\left(\sum_{i=1}^{m} u_i - 1\right) +\\ \eta\left(\sum_{j=1}^{n} v_j - 1\right) - \lambda^T\sum_{i=1}^{m}\sum_{j=1}^{n}(\mathbf{g}(x_i, y_j) - \theta)u_i v_j\end{aligned}$$

By Lagrangian multiplier method, the maximum of the constrained logarithm of

the empirical likelihood ratio function (3.7) occurs at

$$\begin{aligned} u_i &= \frac{1}{m + \lambda^T \sum_{j=1}^{n} (\mathbf{g}(x_i, y_j) - \theta) v_j} \\ v_j &= \frac{1}{n + \lambda^T \sum_{i=1}^{m} (\mathbf{g}(x_i, y_j) - \theta) u_i} \end{aligned} \tag{3.8}$$

These two equations do not immediately provide a solution for the probabilities u_i and v_j, but as we shall see in Chapter 4, they lead to an algorithm that can be used to find the solutions.

We now introduce some more notation.

Let

$$A(X, t_1, t_2) = \mathbb{E}[I[Y > X] I[t_1 < X < t_2] | X] = I[t_1 < X < t_2](1 - G(X))$$

and

$$\begin{aligned} B(Y, t_1, t_2) &= \mathbb{E}[I[Y > X] I[t_1 < X < t_2] | Y] \\ &= (F(t_2) - F(t_1)) I[t_2 < Y] + (F(Y) - F(t_1)) I[t_1 < Y < t_2] \end{aligned}$$

Theorem 3.1 Suppose that X_i and Y_j are independent random variables with continuous distribution functions F and G.

Furthermore suppose the following conditions hold

(i) $0 < p_1 < p_2 < 1$,

(ii) $\infty > VarA(X) > 0$ and $\infty > VarB(Y) > 0$,

(iii) the density function f of F is positive and continuous at t_1 and t_2.

If $0 < \rho = \lim_{m,n\to\infty} m/n < \infty$, then the limiting distribution of $-2\log R(\theta_0)$ in (3.7) is a chi-square with degree of freedom 3 when the null hypotheses (3.3) is true, that is, when $t_1 = F^{-1}(1-p_2)$, $t_2 = F^{-1}(1-p_1)$ and $pAUC = pAUC_0(p_1, p_2)$.
In fact we have

$$-2\log R(\mu_0) = (n+m)(U-\theta_0)\Sigma^{-1}(U-\theta_0)^\top + o_p(1) \tag{3.9}$$

where

$$U = \left(\frac{1}{m}\sum_{i=1}^{m} g_1(X_i, t_1),\ \frac{1}{m}\sum_{i=1}^{m} g_2(X_i, t_2),\ \frac{1}{m}\sum_{i=1}^{m} A(X_i, t_1, t_2) + \frac{1}{n}\sum_{j=1}^{n} B(Y_j, t_1, t_2) \right),$$

$\theta_0 = (1-p_2, 1-p_1, pAUC)$, and $\mu_0 = (t_{10}, t_{20}, pAUC)$.

The above condition (i) assures that the pAUC is the internal portion of the ROC curve between p_1 and p_2. Condition (ii) ascertains that the variance of pAUC estimate is finite and larger than zero. Here the variance structure is the consistent estimate of the variance for the U-statistic provided by Sen (1967). Condition (iii) indicates that t_1 and t_2 corresponding to $1-p_2$ and $1-p_1$ quantiles of the non-diseased sample exist.

For the proof of Theorem 3.1, please refer to Owen (2001), where Owen proved that the empirical likelihood test on two independent samples with one constrain estimation equation has an asymptotic $\chi^2_{(1)}$ and Owen also mentioned that the empirical likelihood test on two independent samples with k constrains has an asymptotic $\chi^2_{(k)}$.

It follows from Theorem 3.1 that for any $0 < \alpha < 1$ an empirical likelihood confidence region for $\mu = (t_1, t_2, pAUC(p_1, p_2))$ with an asymptotic coverage probability $1 - \alpha$ is given by $\{\mu | -2 \log R(\mu) < c_{1-\alpha}\}$ where $c_{1-\alpha}$ is defined as $P(\chi^2_3 > c_{1-\alpha}) = \alpha$.

Hypothesis Test for pAUC at FPR = (0, p) or FPR = (p, 1), 0 < p < 1

For the partial AUC at [0, p) or (p, 1] as calculated in equations (3.1.a) or (3.1.b) the hypothesis is respectively formulated as

$$H_{00} : \begin{cases} 1 - F(t) = p \\ \int_t^{\infty} (1 - G(x)) dF(x) = pAUC \end{cases} \tag{3.3.a}$$

or

$$H_{00} : \begin{cases} 1 - F(t) = p \\ \int_{-\infty}^{t} (1 - G(x)) dF(x) = pAUC \end{cases} \tag{3.3.b}$$

The discrete version of the above hypothesis on the parameters are

$$\begin{cases} g(X, t) = g(X) = I(X > t) \\ g_4(X, Y, t) = g_4(X, Y) = I(Y > X) I(X \geq t) \end{cases}$$

or

$$\begin{cases} g(X, t) = g(X) = I(X > t) \\ g_5(X, Y, t) = g_5(X, Y) = I(Y > X) I(X \leq t) \end{cases}$$

The simultaneous equalities of the hypotheses using vector notation are expressed

as

$$\sum_{i=1}^{m}\sum_{j=1}^{n}(\mathbf{g}(x_i, y_j) - \theta)u_i v_j = \mathbf{0}$$

where $(\mathbf{g}(X,Y))^T = (g(X),\ g_4(X,Y))$ or $(\mathbf{g}(X,Y))^T = (g(X),\ g_5(X,Y))$, $\theta^T = (p,\ pAUC)$, and $\mathbf{0}^T = (0,\ 0)$. $(t,\ pAUC)$ are the true parameters and t is contained in the function $\mathbf{g}(X,Y)$.

With the above $\mathbf{g}(X,Y)$ and θ, the constrained logarithm ($\log R(t,\ pAUC)$) of the empirical likelihood ratio function of hypothesis (3.3.a) or (3.3.b) has the same form as the right hand side of function (3.7) and the equations of u_i, v_j in (3.8) maximize this constrained logarithm of the empirical likelihood ratio function of hypothesis (3.3.a) or (3.3.b).

Let $A_1(X,t) = I[X > t](1 - G(X))$ and $B_1(Y,t) = (F(Y) - F(t))I[Y > t]$ or Let $A_2(X,t) = I[X < t](1 - G(X))$ and $B_2(Y,t) = F(t)I[Y > t] + F(Y)I[Y \leq t]$ for hypothesis (3.3.a) or (3.3.b), respectively.

Theorem 3.1.a Suppose that X_i and Y_j are independent random variables with continuous distribution functions F and G.

Furthermore suppose the following conditions hold

(i) $0 < p < 1$,

(ii) $\infty > VarA_1(X) > 0$ and $\infty > VarB_1(Y) > 0$,

(iii) the density function f of F is positive and continuous at t.

If $0 < \rho = \lim_{m,n\to\infty} m/n < \infty$, then the limiting distribution of $-2\log R(\theta_0)$ in (3.7) is a chi-square with degree of freedom 2 when the null hypotheses (3.3.a) is true, that is, when $t = F^{-1}(1-p)$ and $pAUC = pAUC_0[0,p)$.

Theorem 3.1.b Suppose that X_i and Y_j are independent random variables with continuous distribution functions F and G.

Furthermore suppose the following conditions hold

(i) $0 < p < 1$,

(ii) $\infty > VarA_2(X) > 0$ and $\infty > VarB_2(Y) > 0$,

(iii) the density function f of F is positive and continuous at t.

If $0 < \rho = \lim_{m,n\to\infty} m/n < \infty$, then the limiting distribution of $-2\log R(\theta_0)$ in (3.7) is a chi-square with degree of freedom 2 when the null hypotheses (3.3.b) is true, that is, when $t = F^{-1}(1-p)$ and $pAUC = pAUC_0(p, 1]$.

From Theorem 3.1.a and Theorem 3.1.b, we still have expression (3.9)

$$-2\log R(\mu_0) = (n+m)(U-\theta_0)\Sigma^{-1}(U-\theta_0)^\top + o_p(1) \tag{3.9}$$

where

$$U = \left(\frac{1}{m}\sum_{i=1}^{m} g(X_i, t),\ \frac{1}{m}\sum_{i=1}^{m} A_1(X_i, t) + \frac{1}{n}\sum_{j=1}^{n} B_1(Y_j, t)\right),$$

$\theta_0 = (p, pAUC[0, p))$, and $\mu_0 = (t, pAUC[0, p))$ for Theorem 3.1.a.

$$U = \left(\frac{1}{m}\sum_{i=1}^{m} g(X_i, t),\ \frac{1}{m}\sum_{i=1}^{m} A_2(X_i, t) + \frac{1}{n}\sum_{j=1}^{n} B_2(Y_j, t)\right),$$

$\theta_0 = (p, pAUC(p, 1])$, and $\mu_0 = (t, pAUC(p, 1])$ for Theorem 3.1.b.

It follows from Theorem 3.1.a or Theorem 3.1.b that for any $0 < \alpha < 1$ an empirical likelihood confidence region for $\mu = (t,\ pAUC_0[0,\ p))$ or $\mu = (t,\ pAUC_0(p,\ 1])$ with an asymptotic coverage probability $1-\alpha$ is given by $\{\mu | -2\log R(\mu) < c_{1-\alpha}\}$

where $c_{1-\alpha}$ is defined as $P(\chi_2^2 > c_{1-\alpha}) = \alpha$.

Profile Empirical Likelihood Function

In reality, we are just interested in the inference on $pAUC$ alone, which can be accomplished by profile out the nuisance parameter t_1 and t_2 (or t) from the above empirical likelihood ratio functions (3.7). t_1 and t_2 (or t) are contained in the function $g(\cdot)$ in (3.7). In other words, t_1 and t_2 (or t) become the nuisance parameters and need to be profiled out.

Profile likelihood statistic has been used in parametric likelihood ratio test (See the review in Chapter 1).

We apply the profile likelihood statistic method to (3.7), and we have

$$\begin{aligned} -2\log R(pAUC) &= -2\max_{\mathbf{t}} \log R(\mathbf{t}, pAUC) \\ &= \min_{\mathbf{t}}(-2\log R(\mathbf{t}, pAUC)) \\ &= -2\log R(\hat{t}, pAUC) \end{aligned} \tag{3.10}$$

Theorem 3.2 Under the same condition as Theorem 3.1, the limiting distribution of the above defined profile empirical likelihood ratio test $-2\log R(pAUC)$ is a chi-square with one degree of freedom when the null hypotheses $H_0 : pAUC = pAUC_0$ is true.

Proof: We minimize expression (3.9) with respect to the two nuisance parameters t_1 and t_2 or one nuisance parameter t and use the Lemma 2.2.1 in Chapter 2 to arrive that the likelihood ratio test (3.10) is chi square with one degree of freedom under the null hypothesis.

It follows from Theorem 3.2 that for any $0 < \alpha < 1$ an empirical likelihood confidence interval for $\theta = pAUC_0(p_1, p_2)$ or $\theta = pAUC_0[0, p)$ or $\theta = (pAUC_0(p, 1]$ with an asymptotic coverage probability $1 - \alpha$ is given by $\{\theta| - 2\log R(\theta) < c_{1-\alpha}\}$ where $c_{1-\alpha}$ is defined as $P(\chi_1^2 > c_{1-\alpha}) = \alpha$.

3.3 Simulation: Chi square QQ plots

QQ plots in Figure 3.2 show that the test statistics for hypotheses in equation (3.3) is a chi-square distribution with three degrees of freedom when the null hypotheses are true; the test statistics for hypotheses in equations (3.3.a) or (3.3.b) is a chi-square distribution with two degrees of freedom when the null hypotheses are true. The smoothing parameter used in these simulations is $m^{-1/2}$.

Figure 3.3 shows the QQ plots of the profiled test statistics of the hypotheses tests in Figure 3.2, which indicates that the test statistics of the profiled empirical likelihood ratio test has a chi-square distribution with one degree of freedom when the null hypothesis is true.

3.4 Simulation: Confidence Intervals and Coverage Probabilities

Qin et al. (2011) did two simulation studies of pAUC inference. In the first simulation study, they chose standard normal distribution for the non-diseased population and $N(2, 2)$ for the diseased population. In the second simulation study, they chose standard exponential distribution (with rate=1) for the non-diseased population and a exponential distribution with rate=0.25 for the diseased population. In both studies, they generate 1,000 random samples of size m from the non-diseased population and

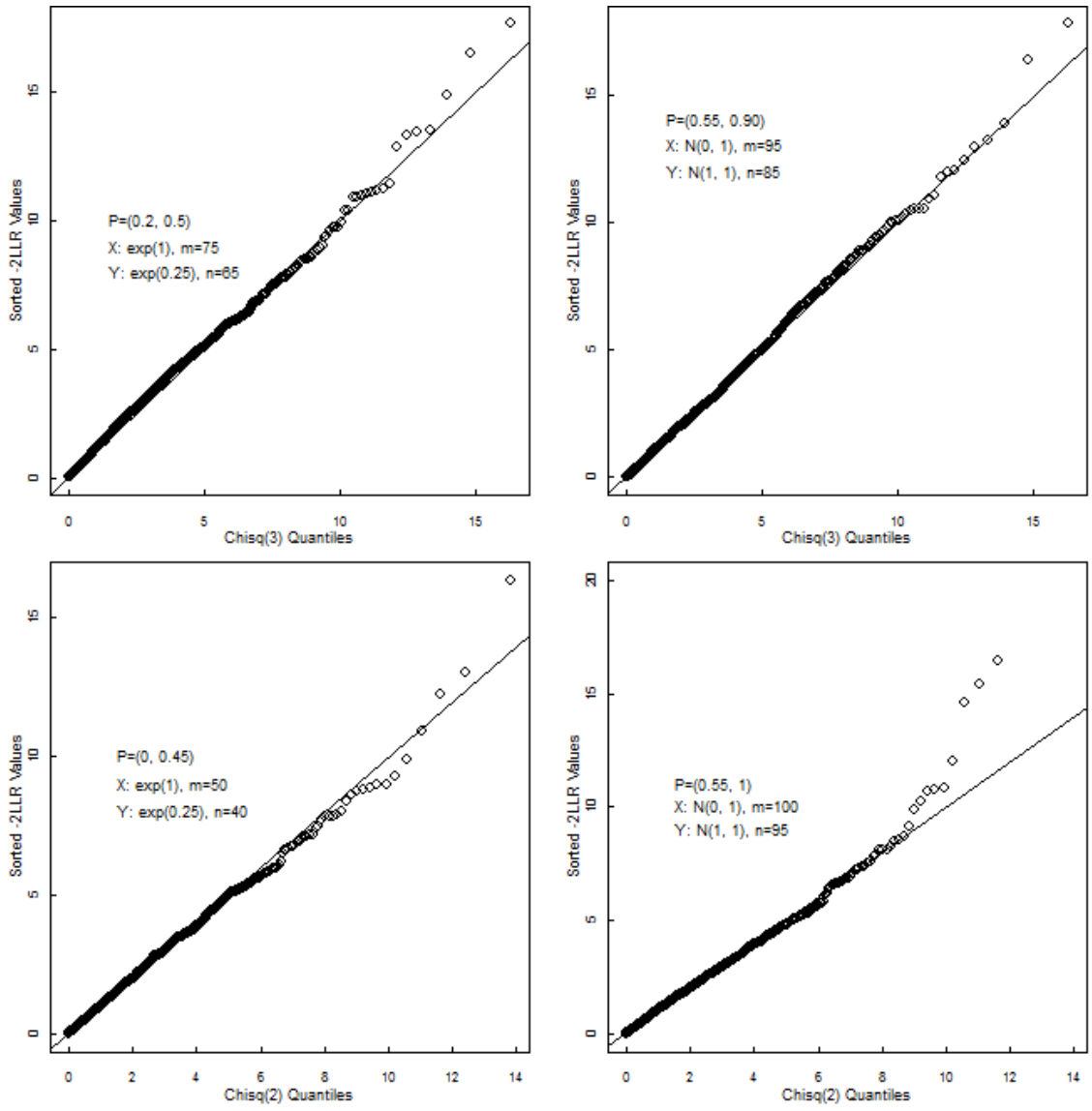

Figure 3.2: Chi-square QQ plots for empirical likelihood tests on hypotheses in equations (3.3), (3.3.a) and (3.3.b)

of size n from the diseased population at several combinations of sample sizes (m, n). Based on the two simulation studies, Qin et al. (2011) recommended four methods (NA-QJZ, BII, HBELI and HBELII) for the inference of pAUC after they compared the performance of 95% confidence intervals of several different inference methods for the partial AUC at $(p_1, p_2) = (0, 0.1), (0, 0.7), (0.05, 0.5)$. Here NA-QJZ is a normal

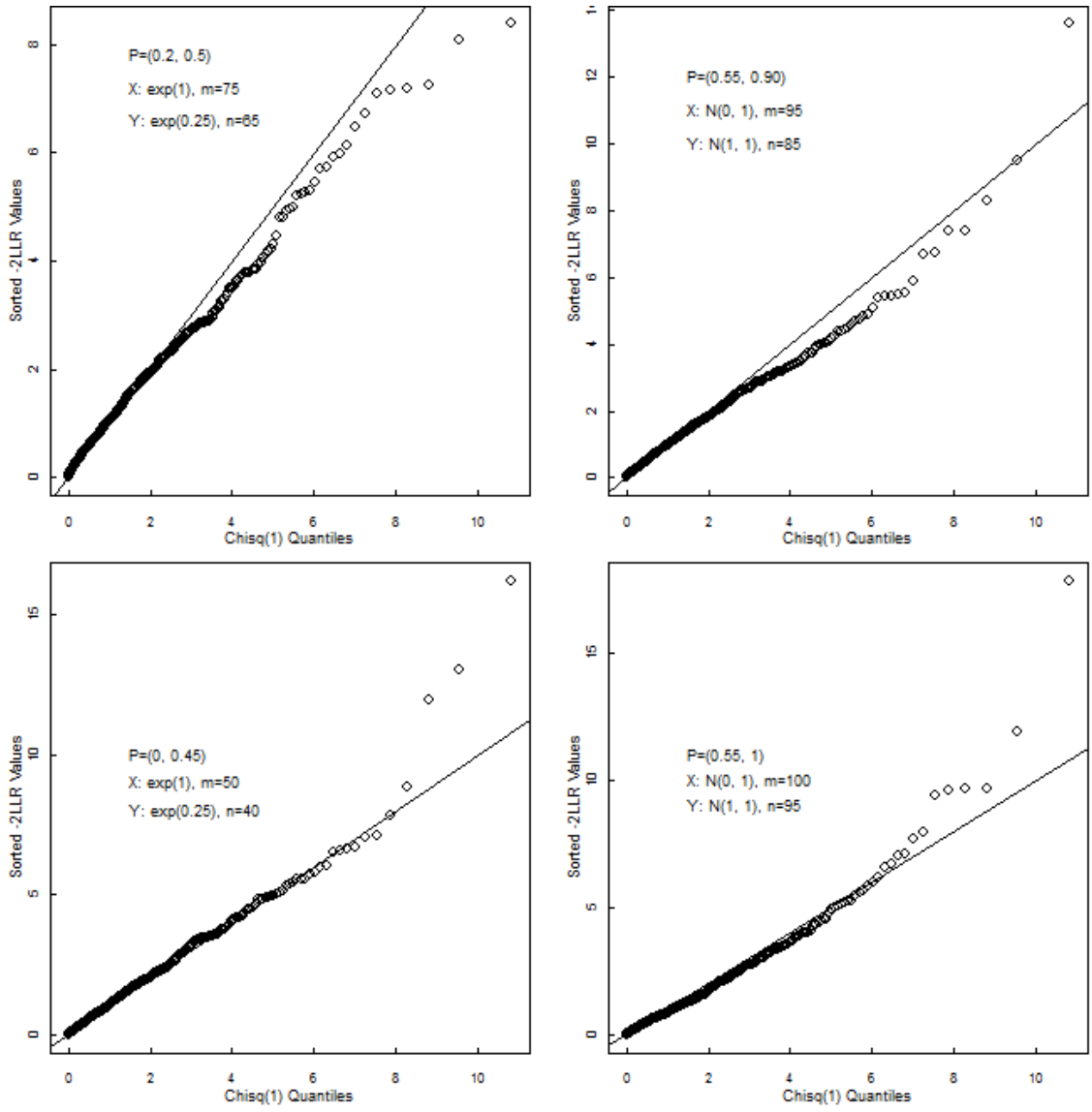

Figure 3.3: Chi-square QQ plots for profiled empirical likelihood tests

approximation-based confidence interval; BII is a bootstrap-based interval; HBELI and HBELII are hybrid bootstrap and empirical likelihood (HBEL) confidence intervals. HBELI applied bootstrap to estimate the variance of pAUC estimate. HBELII is a bootstrap interval based on the empirical likelihood test statistics they formulated. To compare with the methods Qin et al. (2011) recommended, we perform the same simulation studies using the ELseesaw method. Tables 3.1 and 3.2 show the

comparisons of ELseesaw method with the four methods (NA-QJZ, BII, HBELI and HBELII). ELseesaw method achieves the shortest confidence intervals in all the simulations of exponential samples of different sample sizes and different partial AUC at (p_1, p_2). ELseesaw method obtains comparable confidence intervals with NA-QJZ for the normal samples, which are shorter than the confidence intervals from BII, HBELI and HBELII. In most cases, coverage probabilities of ELseesaw method are accurate except at $(m, n) = (30, 30)$ and $(p_1, p_2) = (0, 0.1)$ the coverage probabilities are lower than 95% for both normal samples and exponential samples. The smoothing parameter used in these simulations is m^{-1}.

3.5 Discussion and Conclusion

Tables 3.1 and 3.2 show that Elseesaw method generates relatively more accurate coverage probabilities and shorter confidence intervals at all combinations of different portions of pAUC and sample sizes listed on Tables 3.1 and 3.2 except pAUC of (0, 0.1) at sample sizes (30, 30) compared to the methods suggested by Qin et al. (2011). The computation speed of ELseesaw method is adequate. On average it takes a laptop computer with Intel(R) Core(TM) i5-4310U CPU at 2.00GHz and 3.88 GB usable memory (RAM) about 6 minutes to calculate one confidence interval for pAUC at (0.05, 0.5) for normal samples with sample sizes (100, 100), which is the most time-consuming scenario on Table 3.1. The actual coverage probabilities from ELseesaw method for pAUC of (0, 0.1) at sample sizes (30, 30) are smaller than the nominal coverage probability for both normal samples and exponential samples. NA-QJZ method obtains comparable coverage probabilities with ELseesaw method

Table 3.1: Coverage Probability and Average Length of nominal 95% Confidence Intervals of Partial AUC of Normal Samples

		Coverage Probability			Average Length		
(m, n)	Method	(0, 0.1)	(0, 0.7)	(0.05, 0.5)	(0, 0.1)	(0, 0.7)	(0.05, 0.5)
(30, 30)	NA-QJZ	0.929	0.927	0.951	0.042	0.194	0.135
	BII	0.998	0.936	0.954	0.080	0.215	0.170
	HBELI	0.998	0.957	0.977	0.070	0.213	0.163
	HBELII	0.998	0.969	0.987	0.071	0.220	0.180
	ELseesaw	0.922	0.948	0.962	0.040	0.187	0.135
(50, 50)	NA-QJZ	0.913	0.938	0.945	0.033	0.149	0.106
	BII	0.990	0.949	0.968	0.052	0.160	0.124
	HBELI	0.996	0.957	0.967	0.048	0.155	0.120
	HBELII	0.994	0.954	0.975	0.049	0.157	0.125
	ELseesaw	0.973	0.957	0.973	0.035	0.146	0.108
(100, 100)	NA-QJZ	0.947	0.947	0.941	0.023	0.105	0.076
	BII	0.976	0.948	0.928	0.031	0.110	0.082
	HBELI	0.983	0.942	0.952	0.029	0.106	0.080
	HBELII	0.986	0.950	0.950	0.030	0.106	0.082
	ELseesaw	0.973	0.952	0.958	0.025	0.105	0.076
(50, 30)	NA-QJZ	0.936	0.927	0.927	0.039	0.186	0.131
	BII	0.982	0.929	0.918	0.057	0.193	0.144
	HBELI	0.986	0.925	0.961	0.052	0.184	0.139
	HBELII	0.990	0.937	0.962	0.056	0.193	0.149
	ELseesaw	0.965	0.947	0.956	0.039	0.180	0.129
(80, 50)	NA-QJZ	0.939	0.950	0.943	0.030	0.144	0.104
	BII	0.978	0.945	0.932	0.039	0.146	0.111
	HBELI	0.990	0.947	0.953	0.038	0.147	0.108
	HBELII	0.985	0.940	0.956	0.038	0.147	0.110
	ELseesaw	0.966	0.945	0.953	0.031	0.141	0.102

at the same conditions. While methods BII, HBELI and HBELII gain higher coverage probabilities than the nominal coverage probability at these conditions. These three methods applied bootstrap in the algorithms and the sample sizes were augmented from bootstrap, which might be the reason why these three methods acquired high coverage probability for pAUC of (0, 0.1) at sample sizes (30, 30).

In conclusion, the ELseesaw method, an empirical likelihood ratio method for

Table 3.2: Coverage Probability and Average Length of nominal 95% Confidence Intervals of Partial AUC of Exponential Samples

		Coverage Probability			Average Length		
(m, n)	Method	(0, 0.1)	(0, 0.7)	(0.05, 0.5)	(0, 0.1)	(0, 0.7)	(0.05, 0.5)
(30, 30)	NA-QJZ	0.906	0.905	0.945	0.046	0.200	0.148
	BII	0.990	0.936	0.968	0.078	0.227	0.179
	HBELI	0.997	0.954	0.981	0.065	0.218	0.169
	HBELII	0.999	0.969	0.988	0.075	0.223	0.181
	ELseesaw	0.902	0.949	0.960	0.041	0.194	0.142
(50, 50)	NA-QJZ	0.919	0.939	0.944	0.035	0.156	0.115
	BII	0.970	0.936	0.944	0.051	0.167	0.129
	HBELI	0.977	0.958	0.964	0.045	0.164	0.125
	HBELII	0.995	0.949	0.970	0.047	0.165	0.130
	ELseesaw	0.955	0.946	0.956	0.035	0.153	0.115
(100, 100)	NA-QJZ	0.947	0.942	0.938	0.025	0.110	0.082
	BII	0.970	0.944	0.946	0.030	0.114	0.087
	HBELI	0.970	0.949	0.958	0.029	0.114	0.086
	HBELII	0.970	0.962	0.972	0.030	0.111	0.085
	ELseesaw	0.964	0.941	0.951	0.025	0.109	0.081
(50, 30)	NA-QJZ	0.924	0.932	0.943	0.041	0.188	0.138
	BII	0.985	0.936	0.950	0.056	0.198	0.150
	HBELI	0.977	0.926	0.956	0.050	0.190	0.146
	HBELII	0.990	0.950	0.966	0.060	0.195	0.152
	ELseesaw	0.949	0.953	0.957	0.040	0.184	0.136
(80, 50)	NA-QJZ	0.926	0.941	0.936	0.032	0.148	0.109
	BII	0.954	0.960	0.934	0.039	0.152	0.117
	HBELI	0.967	0.950	0.952	0.036	0.149	0.113
	HBELII	0.982	0.932	0.954	0.039	0.143	0.115
	ELseesaw	0.950	0.952	0.962	0.031	0.145	0.107

pAUC is based on solid statistical theories and is practically applicable. The advantages of this method include (1) it doesn't need any assumptions of the original distributions. (2) it doesn't need a formula for the variance. The limiting distribution of our empirical likelihood ratio method of the pAUC estimate under the null hypothesis is a regular chi-square with one degree of freedom. (3) Coverage probabilities are very close to the nominal. (4) Shorter confidence intervals than methods BII, HBELI

and HBELII and comparable confidence intervals to NA-QJZ method. Eventually we will publish an R package of our ELseesaw method as a tool to calculate pAUC.

Chapter 4 Computational Algorithm and R Package

4.1 Smoothing the Indicator Functions in (2.8), (2.21), (2.26) and (3.7)

Chen and Hall (1993) has shown that the coverage accuracy of empirical likelihood confidence intervals for quantiles can be improved by smoothing the quantile functions. The $\mathbf{g}(\cdot)$ functions in the test statistics for μ_T in (2.8) and pAUC in (3.7) comprises several indicator functions (see (2.6) and (3.4)). These indicator functions are equivalent to the empirical quantile function. Thus, we use the following function to smooth the indicator function $I(x \leq x^*)$ in our final definition of $\mathbf{g}(\cdot)$ functions and also use it in our algorithm of calculating the statistics of μ_T, GLC_p, LC_p, and pAUC:

$$I_\epsilon(x, x^*) = \begin{cases} 1 & \text{if } x \leq x^* - \epsilon \\ 0.5 - \frac{3(x-x^*)}{4\epsilon} + \frac{(x-x^*)^3}{4\epsilon^3} & \text{if } x^* - \epsilon \leq x \leq x^* + \epsilon \\ 0 & \text{if } x \geq x^* + \epsilon \end{cases}$$

for any $\epsilon > 0$. The above is a second-order kernel smoothing function with ϵ as the smoothing parameter. Figure 4.1 shows an example of the smoothing function (4.1) at $x^* = 0$ and $\epsilon = 0.6$.

Chen and Hall (1993) suggested that the smoothing parameter in the range $n^{-1/2}$, $n^{-3/4}$ generally provided good coverage accuracy for empirical likelihood confidence intervals of the quantiles. Chen and Hall (1993) also mentioned that less smoothing

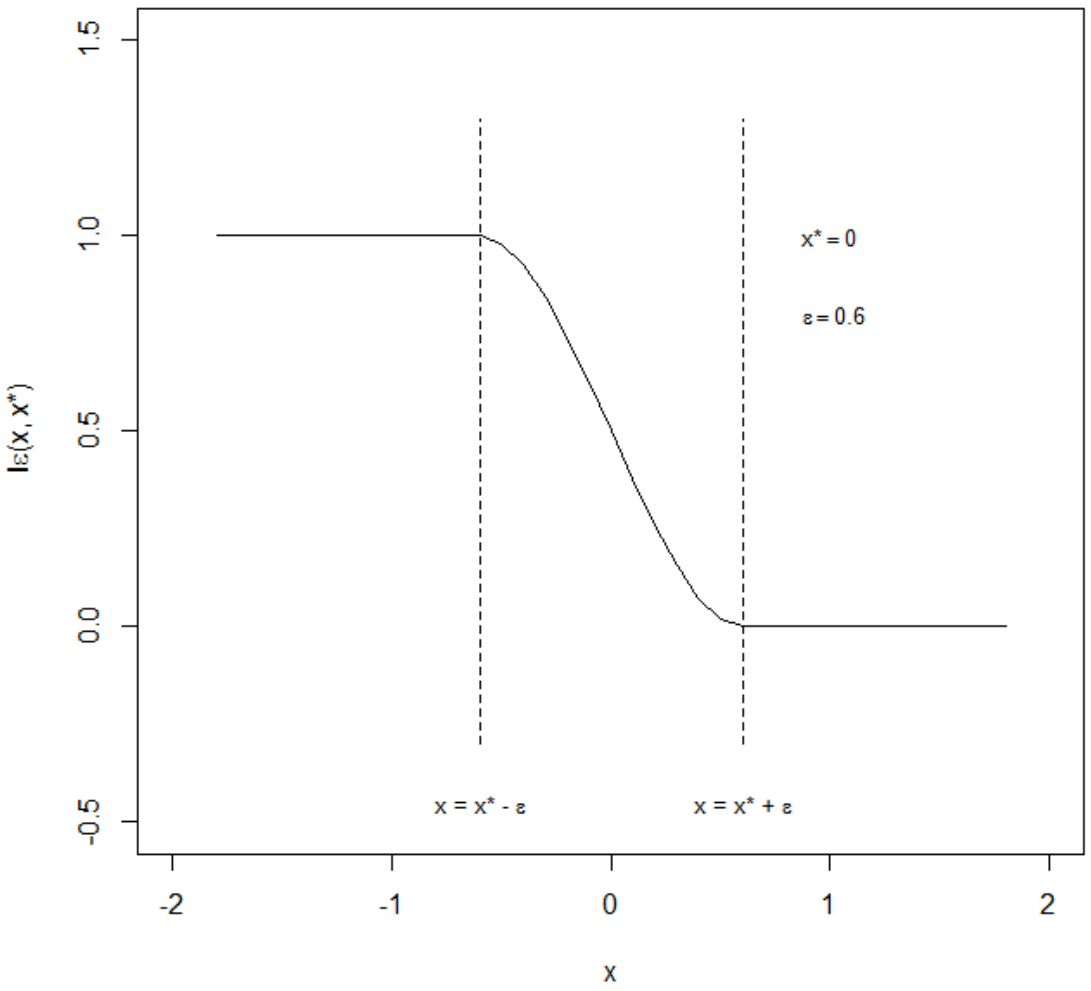

Figure 4.1: An example of the smoothing function (4.1) at $x^* = 0$ and $\epsilon = 0.6$

than this range is desired when the underlying distribution is heavily skewed. We try different smoothing parameters in this study, which are listed in each simulation.

4.2 Algorithm for Calculating the Empirical Likelihood Functions in Chapter 2 and Chapter 3

The empirical likelihood functions (2.8), (2.21), and (2.26) in Chapter 2 are based on one sample, which can be calculated by the function 'el.test' in the R package 'emplik' developed by Zhou and Yang (2014). Thus, we directly call function 'el.test' to compute the empirical likelihood functions (2.8), (2.21), and (2.26).

The empirical likelihood ratio function (3.7) in Chapter 3 involves two samples. To calculate the the empirical likelihood ratio function (3.7), we need to solve (3.8) and (3.5) simultaneously to get the $\mathbf{u}$ and $\mathbf{v}$ that maximize the logarithm of the empirical likelihood ratio function (3.7). However, numerically solving (3.5) and (3.8) is not easy since $\mathbf{u}$ and $\mathbf{v}$ are cross-related. None of the traditional optimization algorithms such as Newton Raphson and quadratic programming methods worked out of the box here. When sample sizes are relatively small, for example, $m = 50$ and $n = 40$, Newton Raphson method failed because of the computational singularity of the derivative matrix with respect to λ of the function in (3.5) after substituting (3.8) to it for quite amount of samples. Quadratic programming method often ran into not positive definite leading minor of the derivative matrix. Here we originate a new method "ELseesaw", which simplifies the minimization of the empirical likelihood ratio test of two samples to the minimization of one sample empirical likelihood ratio test in four steps.

1. Minimize the one sample (X_i sample) empirical likelihood ratio $-2\log R(\theta_0')$ over u_i, when we fix $v_j = 1/n$, and ${\theta_0'}^T = (t_1, t_2, pAUC')$ and $pAUC'$ is a

temporary point in the vicinity of $pAUC_0$ and $\widehat{pAUC}$.

2. Minimize the one sample (for Y_j sample) $-2\log R(\theta_0)$ over v_j at u_i gotten from step 1. Here $\theta_0 = pAUC_0$.

3. Sum the minimum $-2\log R$ calculated from steps 1 and 2. This summation is a function of $pAUC'$.

4. Apply Golden Section Search Optimization method in the vicinity of $pAUC_0$ and $\widehat{pAUC}$ to find $pAUC' = pAUC^*$ so that $pAUC^*$ minimizes the summation of step 3.

We use the function 'el.test' in the R package of 'emplik' by Zhou and Yang (2014) to minimize the test statistics of the one sample empirical likelihood ratio test in steps 1 and 2.

Figure 4.2 shows that such point $pAUC^*$ that minimizes the summation of the empirical likelihood ratio test statistics of step 1 and step 2 exists. In fact, u_i and v_j obtained from the above steps at $pAUC^*$ minimize $-2\log R(\theta)$ and satisfy all the constrains in the empirical likelihood function (3.7). The summation of step 3 at $pAUC^*$ is the minimum of $-2\log R(\theta)$.

4.3 The Profile likelihood

The empirical likelihood functions (2.8), (2.21), (2.26), and (3.7) contain two or three parameters. In reality we are just interested in the inference on one parameter such as trimmed mean or general Lorenz curve for the one sample scenario or pAUC for the

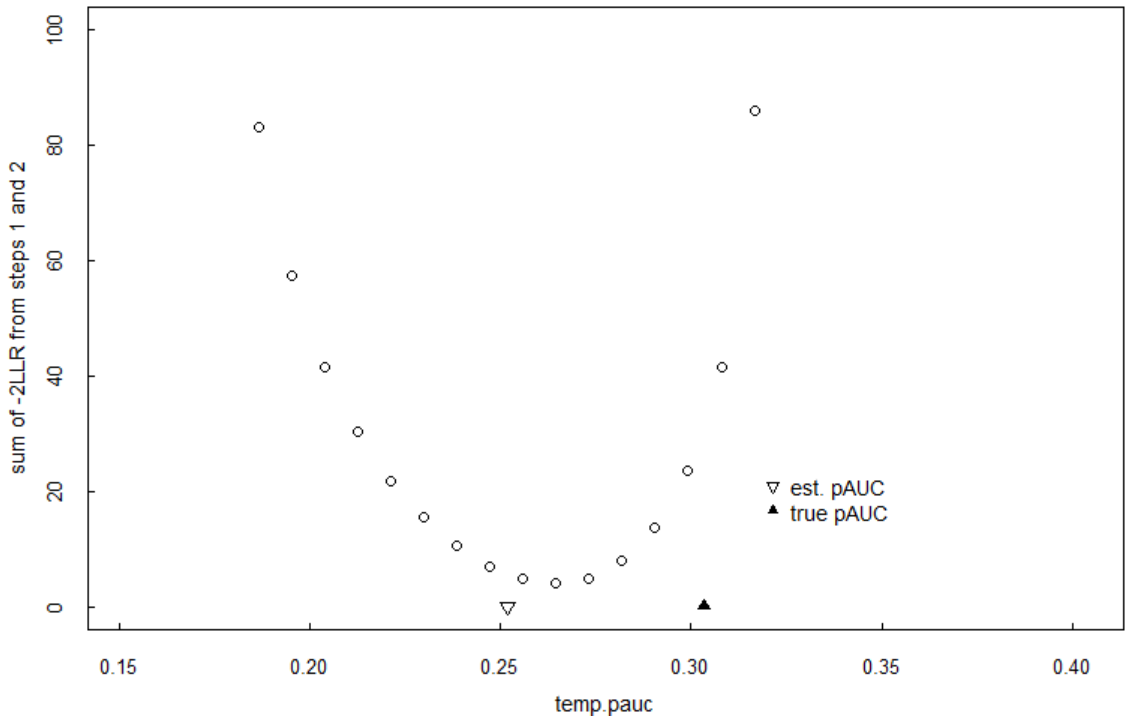

Figure 4.2: Sum of -2LLR as a function of temporary pAUC

$X \sim N(0, 1)$, $m = 50$; $Y \sim N(1, 1)$, $n = 45$; $\mathbf{p} = (0.2, 0.6)$

two sample scenario, which can be achieved by profile empirical likelihood method as discussed in Chapter 2 and Chapter 3. It can be computationally challenging to optimize a likelihood over some nuisance parameters in the empirical setting. We note that the test statistics is a piecewise constant function of t_1 and t_2 for the trimmed mean or pAUC at (p_1, p_2) or is a piecewise constant function of t for general Lorenz Curve or pAUC at $[0, p)$ or $(p, 1]$, which makes the minimization relatively easy.

For simplicity, we describe the situation that profiles out one nuisance parameter such as general Lorenz curve and pAUC at $[0, p)$ or $(p, 1]$.

To minimize the test statistics over one quantile, we follow the following steps:

1. Search t over three consecutive t_i to find the minimum of the test statistics and

the t_i at which the minimum occurred.

2. Center at this t found at Step 1 where the minimum test statistics occurred adding new consecutive t value and compare the test statistics at new consecutive t value with the minimum test statistics from Step 1.

3. If the t at which the minimum test statistics occurred changes, repeat step 2 until the t at which the minimum test statistics occurred is stable, when the nuisance parameter t is profiled out and the minimum test statistics is the profiled empirical likelihood test.

If we start the above search at some strategical t values, such as the sample p quantile for the general Lorenz Curve or the sample $1-p$ quantile for pAUC at [0, p) or (p, 1], we can quickly find the profiled empirical likelihood test for hypothesis test on GLC_p, LC_p or pAUC at [0, p) or (p, 1].

For profiling the empirical likelihood test on μ_T or $pAUC(p_1, p_2)$, we need to do the above search at t_1 and t_2 simultaneously and find the minimum among all the combinations of t_1 and t_2, which renders the computation of profiling empirical likelihood test on μ_T or $pAUC(p_1, p_2)$ three times as much as the computation for profiling empirical likelihood test on GLC_p, LC_p or pAUC at [0, p) or (p, 1].

Figure 4.3 shows that 2LLR of test on μ_T at ($p_1 = 0.2$, $p_2 = 0.8$) as a function of t_1 and t_2. The sample is from N(0, 1) and sample size is 200. Figure 4.3 shows that the minimum of -2LLR (the red dot) exists among the different combinations of quantiles t_1 and t_2.

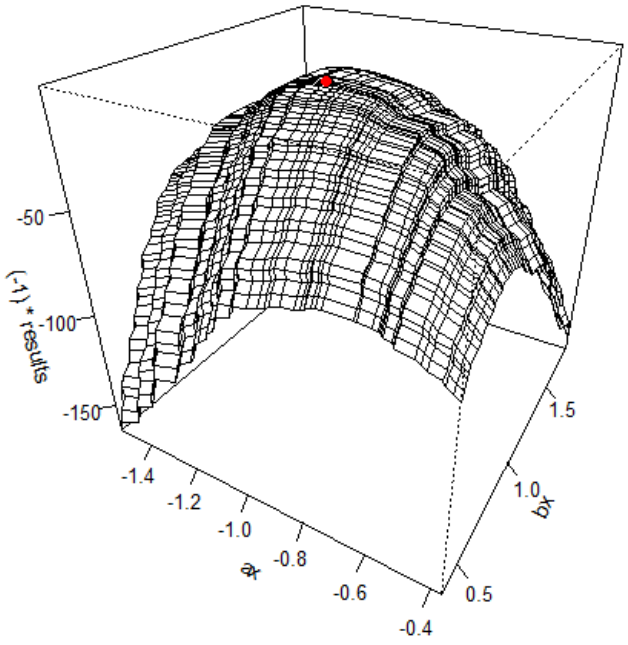

Figure 4.3: Negative test statistics (-results) for μ_T as a function of t_1 (ax) and t_2 (bx)

4.4 R Package 'pAUC'

Based on the above-described algorithms, we develop an R package 'pAUC' to compute the empirical likelihood test statistics, p value for hypothesis on pAUC/AUC/-trimmed mean/truncated mean/Lorenz Curve and to calculate the estimate and confidence interval of pAUC/AUC/trimmed mean/truncated mean/Lorenz Curve. In this section, we provide the documentations for 14 external functions and a list of annotated code for both external functions and internal functions.

External Functions

The names of the external functions are as follows:

```
eltest
```

```
neighb
eltest.p
neighb.p
eltest.p.lc
neighb.p.lc
est.tr.mean
eltest2step
neighb.xy
eltest2step.p12
neighb.xy3p12
eltest2step.all
est.pAUC
find.UL
```

We will individually introduce these external functions here.
eltest

Compute the empirical likelihood ratio test of three parameters: percentiles at p1 and p2, and trimmed mean at (p1, p2)

Description:

This function calls the function 'el.test' in R Package 'emplik'

Usage:

eltest(ab = vector("numeric", 2), x.sample, mu, eps)

Arguments:

ab	a vector of two elements for the percentiles at p1 and p2
x.sample	a random sample
mu	the null hypothesis vector with three elements (p1, p2, μ_T)
eps	a smoothing parameter

Details:

If mu is far away from the true value vector, "-2LLR" may be positive infinite.

Value:

"-2LLR"	the -2 log likelihood ratio; approximate chi square distribution with a degree of freedom 3 under H_0

Reference:

Zhou and Yang (2014)

Example:

```
set.seed(123)
nsize <- 50
p1<- 0.21
p2<- 0.79
q1 <- qnorm(p1)
q2 <- qnorm(p2)
rp_trim <- function(x){
  y <- x*dnorm(x)
  y
}
omu_trim <- integrate(rp_trim, lower=q1, upper=q2)$value
mu_trim <- c(p1, p2, omu_trim)
x <- rnorm(nsize)
eltest(c(q1, q2), x, mu=mu_trim, eps=1/nsize)
```

Code

```
#a function to call el.test in emplik to compute -2LLR for trimmed
    mean at (p1, p2)
#the internal smoothing function 'myfun5' is called as well
eltest <- function(ab = vector("numeric",2), x.sample, mu, eps )
{
  axb <- matrix(c(myfun5(x.sample, ab[1], eps), myfun5(x.sample, ab
      [2], eps), x.sample*((1- myfun5(x.sample, ab[1], eps)) * myfun5
      (x.sample, ab[2], eps))), ncol=3)
  all <- el.test(axb, mu)
  list('-2LLR'=all$`-2LLR`)
}
```

neighb

Compute the empirical likelihood ratio test for trimmed mean at (p1, p2), $0 < p1 < p2 < 1$

Description:

This function apples the profile algorithm described in Section 4.3

Usage:

neighb(sp12 = vector("numeric", 2), x, true, eps)

Arguments:

sp12	a vector of two elements for p1 and p2
x	a random sample
true	the trimmed mean to be tested
eps	a smoothing parameter

Details:

The empirical likelihood ratio test for trimmed mean at (p1, p2) is executed in two steps. First, test the hypotheses with three parameters, two of which are the nuisance parameters p1 and p2. Second, profile out the nuisance parameters p1 and p2. This function performs the second step - profiling out the nuisance parameters. This function searches the minimum of the empirical likelihood ratio test statistics among combinations of p1-th quantile and p2-th quantile with p1-th quantile and p2-th quantile moving along the consecutive sample points separately. Since the test statistics is a piecewise constant function of the quantiles for p1 and p2, the minimum is easily found if searching starts at some strategical quantiles such as the sample quantiles at p1 and p2.

Value:

"-2LLR"	the -2 loglikelihood ratio; approximate chi square distribution with one degree of freedom under H_0
Pval	the observed P-value by chi square approximation

Example:

```
set.seed(123)
nsize <- 50
p1<- 0.21
p2<- 0.79
q1 <- qnorm(p1)
q2 <- qnorm(p2)
rp <- function(x){
  y <- x*dnorm(x)
  y
}
omu <- integrate(rp, lower=q1, upper=q2)$value
sp12 <- c(p1, p2)
x <- rnorm(nsize)
neighb(sp12=sp12, x=x, true=omu, eps=1/nsize)
```

Code

```
# a function to find the min of -2LLR among different cutting points
    of x sample at two sides

neighb <- function (sp12 = vector("numeric", 2), x, true, eps)
{
    if (sp12[1] >= sp12[2])
```

```
    stop("the first sample quantile has to be smaller than the
       second")
n <- length(x)
sortx_w <- myWdataclean2(z = x)
sortedx <- sortx_w$value
cx <- cumsum(sortx_w$weight/n)
idex1 <- ifelse(sp12[1] == 0, 1, max(which(cx <= sp12[1])))
idex2 <- max(which(cx <= sp12[2]))
mu <- c(sp12, true)
abs <- data.frame(x1 = idex1, x2 = idex2)
res <- eltest(c(sortedx[idex1], sortedx[idex2]), x, mu, eps)
best <- res$"-2LLR"
neg2.llr <- as.vector(best)
ite <- TRUE
while (ite == TRUE) {
    neib1 <- c((idex1 - 1), idex2)
    if (idex1 - 1 > 0) {
        if ((!any(abs[, 1] == neib1[1])) || (!any(which(neib1[1]
            ==
            abs[, 1]) %in% which(neib1[2] == abs[, 2])))) {
            abs <- rbind(abs, neib1)
            res <- eltest(ab = c(sortedx[idex1 - 1], sortedx[
               idex2]),
              x, mu, eps)
            neg2.llr <- c(neg2.llr, res$"-2LLR")
        }
    }
    neib2 <- c(idex1, (idex2 - 1))
    if (idex2 - 1 > idex1) {
        if ((!any(abs[, 2] == neib2[2])) || (!any(which(neib2[1]
            ==
            abs[, 1]) %in% which(neib2[2] == abs[, 2])))) {
            abs <- rbind(abs, neib2)
            res <- eltest(ab = c(sortedx[idex1], sortedx[idex2 -
              1]), x, mu, eps)
            neg2.llr <- c(neg2.llr, res$"-2LLR")
        }
    }
    neib3 <- c((idex1 + 1), idex2)
    if (idex1 + 1 < idex2) {
        if ((!any(abs[, 1] == neib3[1])) || (!any(which(neib3[1]
            ==
            abs[, 1]) %in% which(neib3[2] == abs[, 2])))) {
            abs <- rbind(abs, neib3)
            res <- eltest(ab = c(sortedx[idex1 + 1], sortedx[
               idex2]),
              x, mu, eps)
            neg2.llr <- c(neg2.llr, res$"-2LLR")
        }
    }
    neib4 <- c(idex1, (idex2 + 1))
    if (idex2 + 1 < n) {
        if ((!any(abs[, 2] == neib4[2])) || (!any(which(neib4[1]
            ==
```

```
        abs[, 1]) %in% which(neib4[2] == abs[, 2])))) {
        abs <- rbind(abs, neib4)
        res <- eltest(ab = c(sortedx[idex1], sortedx[idex2 +
          1]), x, mu, eps)
        neg2.llr <- c(neg2.llr, res$"-2LLR")
    }
}
neib5 <- c((idex1 - 1), (idex2 - 1))
if ((idex1 - 1 > 0) & (idex2 - 1 > idex1)) {
    if (!any(which(neib5[1] == abs[, 1]) %in% which(neib5[2]
        ==
        abs[, 2]))) {
        abs <- rbind(abs, neib5)
        res <- eltest(ab = c(sortedx[idex1 - 1], sortedx[
            idex2 -
          1]), x, mu, eps)
        neg2.llr <- c(neg2.llr, res$"-2LLR")
    }
}
neib6 <- c((idex1 - 1), (idex2 + 1))
if ((idex1 - 1 > 0) & (idex2 + 1 < n)) {
    if (!any(which(neib6[1] == abs[, 1]) %in% which(neib6[2]
        ==
        abs[, 2]))) {
        abs <- rbind(abs, neib6)
        res <- eltest(ab = c(sortedx[idex1 - 1], sortedx[
            idex2 +
          1]), x, mu, eps)
        neg2.llr <- c(neg2.llr, res$"-2LLR")
    }
}
neib7 <- c((idex1 + 1), idex2 - 1)
if (idex1 + 1 < idex2 - 1) {
    if (!any(which(neib7[1] == abs[, 1]) %in% which(neib7[2]
        ==
        abs[, 2]))) {
        abs <- rbind(abs, neib7)
        res <- eltest(ab = c(sortedx[idex1 + 1], sortedx[
            idex2 -
          1]), x, mu, eps)
        neg2.llr <- c(neg2.llr, res$"-2LLR")
    }
}
neib8 <- c((idex1 + 1), (idex2 + 1))
if ((idex1 + 1 < idex2) & (idex2 + 1 < n)) {
    if (!any(which(neib8[1] == abs[, 1]) %in% which(neib8[2]
        ==
        abs[, 2]))) {
        abs <- rbind(abs, neib8)
        res <- eltest(ab = c(sortedx[idex1 + 1], sortedx[
            idex2 +
          1]), x, mu, eps)
        neg2.llr <- c(neg2.llr, res$"-2LLR")
    }
```

```
        }
        if (min(neg2.llr) == best) {
            return(list(`-2LLR` = best, Pval = 1 - pchisq(best,
                df = 1)))
            ite <- FALSE
        }
        else {
            best = min(neg2.llr)
            idex1 <- abs[which(neg2.llr == best), 1]
            idex2 <- abs[which(neg2.llr == best), 2]
            ite <- TRUE
        }
    }
}
```

eltest.p

Compute the empirical likelihood ratio test of two parameters: percentile p, and truncated mean at [0, p)

Description:

This function calls the function 'el.test' in R Package 'emplik'

Usage:

eltest.p(cutx1, x.sample, mu, eps)

Arguments:

cutx1	the p-th percentile
x.sample	a random sample
mu	the null hypothesis vector with two elements (p, μ_T)
eps	a smoothing parameter

Details:

If mu is far away from the true value vector, "-2LLR" may be positive infinite.

Value:

"-2LLR"	the -2 loglikelihood ratio; approximate chi square distribution with a degree of freedom 2 under H_0

Reference:

Zhou and Yang (2014)

Example:

```
nsize <- 50
p<- 0.89
rate <- 0.25
q <- qexp(p, rate)
rp_glc <- function(x){
  y <- x*dexp(x, rate=0.25)
  y
}
omu_glc <- integrate(rp_glc, lower= 0, upper=q)$value
mu_glc <- c(p, omu_glc)
x <- rexp(n=nsize, rate=rate)
eltest.p(q, x, mu=mu_glc, eps=1/nsize)
```

Code

```
#a function to call el.test in emplik to compute -2LLR for trimmed
   mean at (0, p)
eltest.p <- function(idex1, x.sample, mu, eps )
{
  axb <- matrix(c(myfun5(x.sample, idex1, eps), x.sample*as.numeric(
     myfun5(x.sample, idex1, eps))), ncol=2)
  all <- el.test(axb, mu)
  list('-2LLR'=all$`-2LLR`)
}
```

neighb.p

Compute the empirical likelihood ratio test for truncated mean at [0, p), $0 < p < 1$

Description:

This function apples the profile algorithm described in Section 4.3

Usage:

neighb.p(p, x, true, eps)

Arguments:

p	a cutting percentage p
x	a random sample
true	the truncated mean to be tested
eps	a smoothing parameter

Details:

The empirical likelihood ratio test for trimmed mean at [0, p) is executed in two steps. First, test the hypotheses with two parameters, one of which is the nuisance parameters p. Second, profile out the nuisance parameter p. This function performs the second step - profiling out the nuisance parameter. This function searches the minimum of the empirical likelihood ratio test statistics along the consecutive sample points. Since the test statistics is a piecewise constant function of the quantile for p, the minimum is easily found if the searching starts at the sample p-th quantile.

Value:

"-2LLR"	the -2 loglikelihood ratio; approximate chi square distribution with one degree of freedom under H_0
Pval	the observed P-value by chi square approximation

Example:

```
nsize <- 50
p<- 0.89
q <- qnorm(p)
rp <- function(x){
  y <- x*dnorm(x)
  y
}
omu <- integrate(rp, lower= -Inf, upper=q)$value
x <- rnorm(nsize)
neighb.p(p, x=x, true=omu, eps=1/nsize)
```

Code

```
# a function to find the min of -2LLR among different cutting points
    of x sample at one side
neighb.p <- function(p, x, true, eps){
  n <- length(x)
  sortx_w <- myWdataclean2(z=x)
  sortedx <- sortx_w$value
  cx<- cumsum(sortx_w$weight/n)
  idex1 <- max(which(cx <= p ))
  mu <- c(p, true)
  abs <- as.vector(idex1)
  res <- eltest.p(idex1, x, mu, eps)
  best <- res$'-2LLR'
  neg2.llr <- as.vector(best)
  ite <- TRUE
  while (ite==TRUE) {
    neib1 <- idex1-1
    if ((neib1 > 0) & (!any(abs==neib1))) {
        abs <- c(abs, neib1)
        res <- eltest.p(sortedx[idex1-1], x, mu, eps)
        neg2.llr <- c(neg2.llr, res$'-2LLR')
      }
```

```
    neib2 <- idex1+1
    if ((neib2 <= n) & (!any(abs==neib2))){
        abs <- c(abs, neib2)
        res <- eltest.p(sortedx[idex1 + 1], x, mu, eps)
        neg2.llr <- c(neg2.llr, res$'-2LLR')
      }
    if (min(neg2.llr)==best){
      return(list('-2LLR'=best, Pval = 1 - pchisq(best, df=1)))
      ite <- FALSE
    }
    else {
      best=min(neg2.llr)
      idex1 <- abs[which(neg2.llr==best)]
      ite <- TRUE
    }
  }
}
```

eltest.p.lc

Compute the empirical likelihood ratio test of two parameters: percentile p, and Lorenz Curve at [0, p)

Description:

This function calls the function 'el.test' in R Package 'emplik'

Usage:

eltest.p(cutx1, x.sample, mu, eps)

Arguments:

cutx1	the p-th percentile
x.sample	a random sample
mu	the null hypothesis vector with two elements (p, LC_p)
eps	a smoothing parameter

Details:

If mu is far away from the true value vector, "-2LLR" may be positive infinite.

Value:

"-2LLR"	the -2 loglikelihood ratio; approximate chi square distribution with a degree of freedom 2 under H_0

Reference:

Zhou and Yang (2014)

Example:

```
nsize <- 500
p<- 0.89
rate <- 0.25
q <- qexp(p, rate)
rp_glc <- function(x){
  y <- x*dexp(x, rate=0.25)
  y
}
omu_glc <- integrate(rp_glc, lower= 0, upper=q)$value
mu_lc <- c(p, omu_glc*rate)
x <- rexp(n=nsize, rate=rate)
eltest.p.lc(q, x, mu=mu_lc, eps=1/nsize)
```

Code

```
#a function to call el.test in emplik to compute -2LLR for Lorenz
    Curve at (0, p)
eltest.p.lc <- function (cutx1, x.sample, mu, eps)
{
     axb <- matrix(c(myfun5(x.sample, cutx1, eps), (x.sample *
         as.numeric(myfun5(x.sample, cutx1, eps)) - x.sample *
         mu[2])), ncol = 2)
     all <- el.test(axb, c(mu[1], 0))
     list(`-2LLR` = all$`-2LLR`)
}
```

neighb.p.lc

Compute the empirical likelihood ratio test for Lorenz Curve at [0, p), $0 < p < 1$

Description:

This function apples the profile algorithm described in Section 4.3

Usage:

neighb.p.lc(p, x, true, eps)

Arguments:

p	a cutting percentage p
x	a random sample
true	the Lorenz Curve to be tested
eps	a smoothing parameter

Details:

The empirical likelihood ratio test for Lorenz Curve at [0, p) is executed in two steps. First, test the hypotheses with two parameters, one of which is the nuisance parameters p. Second, profile out the nuisance parameter p. This function performs the second step - profiling out the nuisance parameter. This function searches the minimum of the empirical likelihood ratio test statistics along the consecutive sample points. Since the test statistics is a piecewise constant function of the quantile for p, the minimum is easily found if the searching starts at the sample p-th quantile.

Value:

"-2LLR"	the -2 loglikelihood ratio; approximate chi square distribution with one degree of freedom under H_0
Pval	the observed P-value by chi square approximation

Example:

```
set.seed(123)
nsize <- 50
p<- 0.89
rate <- 0.25
q <- qexp(p, rate)
rp_glc <- function(x){
  y <- x*dexp(x, rate=0.25)
  y
}
omu_glc <- integrate(rp_glc, lower= 0, upper=q)$value
mu_lc <- omu_glc*rate
x <- rexp(n=nsize, rate=rate)
neighb.p.lc(p, x, mu_lc, eps=1/nsize)
```

Code

```
# a function to find the min of -2LLR among different cutting points
    of x sample at one side
neighb.p.lc <- function (p, x, true, eps)
{
    n <- length(x)
    sortx_w <- myWdataclean2(z = x)
    sortedx <- sortx_w$value
    cx <- cumsum(sortx_w$weight/n)
    idex1 <- max(which(cx <= p))
    mu <- c(p, true)
    abs <- as.vector(idex1)
    res <- eltest.p.lc(sortedx[idex1], x, mu, eps)
    best <- res$"-2LLR"
    neg2.llr <- as.vector(best)
    ite <- TRUE
    while (ite == TRUE) {
```

```
        neib1 <- idex1 - 1
        if ((neib1 > 0) & (!any(abs == neib1))) {
            abs <- c(abs, neib1)
            res <- eltest.p.lc(sortedx[idex1 - 1], x, mu, eps)
            neg2.llr <- c(neg2.llr, res$"-2LLR")
        }
        neib2 <- idex1 + 1
        if ((neib2 <= n) & (!any(abs == neib2))) {
            abs <- c(abs, neib2)
            res <- eltest.p.lc(sortedx[idex1 + 1], x, mu, eps)
            neg2.llr <- c(neg2.llr, res$"-2LLR")
        }
        if (min(neg2.llr) == best) {
            return(list(`-2LLR` = best, Pval = 1 - pchisq(best,
                df = 1)))
            ite <- FALSE
        }
        else {
            best = min(neg2.llr)
            idex1 <- abs[which(neg2.llr == best)]
            ite <- TRUE
        }
    }
}
```

est.tr.mean

Estimate trimmed/truncated means

Description:

This function estimates the trimmed means or truncated means depended on the parameter p

Usage:

est.tr.mean(x.sample, p, ...)

Arguments:

x.sample	a random sample
p	for a trimmed mean $p = (p1, p2)$, $0 < p1 < p2 < 1$; for a truncated mean p is a scalar, $0 \leq p \leq 1$
...	there is an additional argument 'eps' for the smoothing parameter

Details:

For trimmed means at (p_1, p_2), $\hat{\mu}_T = \frac{1}{n}\sum_{i=1}^{n} x_i I[t_1 \leq x_i \leq t_2]$ where $t_1 = \hat{F}^{-1}(p_1) = \sup\{t : \hat{F}(t) \leq p_1\}$; $t_2 = \hat{F}^{-1}(p_2) = \sup\{t : \hat{F}(t) \leq p_2\}$; If the population has a sym-

metric distribution and trims are symmetric (i.e., $p_1 = 1 - p_2$) as well, $\hat{\mu} = \frac{\hat{\mu}_T}{p_2 - p_1}$, where $\hat{\mu}$ is the estimate of population mean. For truncated means at [0, p), $\hat{\mu}_{Tn} = \frac{1}{n}\sum_{i=1}^{n} x_i I[x_i \leq t]$ where $t = \hat{F}^{-1}(p) = \sup\{t : \hat{F}(t) \leq p\}$;

Value:

the returned value is the estimate of trimmed/truncated mean

Example:

```
## An example of trimmed mean at (0.21, 0.79)
nsize <- 50
p1 <- 0.21
p2 <- 0.79
eps <- 1/nsize
x <- rnorm(nsize)
est.tr.mean(x, p=c(p1, p2), eps)

## An example of truncated mean at (0, 0.79)
nsize <- 50
p <- 0.89
eps <- 1/nsize
x <- rnorm(nsize)
est.tr.mean(x, p, eps)
```

Code

```
est.tr.mean <- function(x.sample, p, ...){
  sortedx <- x.sample[order(x.sample)]
  nx <- length(x.sample)
  cx<-cumsum(rep(1/nx, nx))
  if(length(p)==2){
      lowx <- sortedx[max(which(cx <= p[1]))]
      highx <- sortedx[max(which(cx <= p[2] ))]
      est <- mean(as.vector(x.sample*((1-myfun5(x.sample, lowx, ...)
         ) * myfun5(x.sample, highx, ...))))
  }
    else if (length(p)==1){
       highx <- sortedx[max(which(cx <= p ))]
       est <- mean(as.vector(x.sample*myfun5(x.sample, highx, ...)))
  }
  return(est)
}
```

eltest2step

Empirical likelihood test for percentiles at $(1 - p2)$ *and* $(1 - p1)$*, and pAUC at* $(p1, p2)$*. P1 and p2 are false positive rates,* $p1 < p2$

Description:

Compute the empirical likelihood ratio test with three parameters: percentiles at (1-p2) and (1-p1), and pAUC fixed at $pAUC_0$

Usage:

eltest2step(x, ab= vector("numeric",2), y, p1, p2, truepauc, tol=.001, eps)

Arguments:

x	a sample of test results from non-diseased population
ab	a vector of two percentiles at $(1-p2)$ and $(1-p1)$, $0 < p1 < p2 < 1$
y	a sample of test results from diseased population
p1	the smaller false positive rate to be included in pAUC
p2	the larger false positive rate to be included in pAUC
truepauc	the pAUC to be tested
tol	the maximum distance between the current guess and the current range on Golden Section Search Optimization method
eps	a smoothing parameter

Details:

The function apples the 'ELseesaw' algorithm described in Section 4.2 to minimize the empirical likelihood ratio test. "-2LLR" may be positive infinite if the tested values of the parameters are far away from the true values.

Value:

"-2LLR"	the -2 loglikelihood ratio; approximate chi-square distribution with three degrees of freedom under the null hypotheses

Example:

```
mux <- 0
stddx <- 1
muy <- 1
stddy <- 1
p1 <- 0.05
p2 <- 0.5
q1 <- qnorm(p=1-p2, mean=mux, sd=stddx)
q2 <- qnorm(p=1-p1, mean=mux, sd=stddx)
rp<-function(p, mux, stddx, muy, stddy)
  {
    y<-1-pnorm(qnorm(1-p, mux, stddx), muy, stddy)
    y
  }
```

```
###true pAUC at (p1, p2)
truepauc<-integrate(rp, mux=mux, stddx=stddx, muy=muy, stddy=stddy,
   lower=p1,upper=p2)$value
nx <- 40
ny <- 30
x <- rnorm(nx, mux, stddx)
y <- rnorm(ny, muy, stddy)
eltest2step(ab=c(q1, q2), x=x, y=y, truepauc=truepauc, p1=p1, p2=p2,
    tol=0.001, eps=1/nx)
```

Code

```
eltest2step <- function(x, ab= vector("numeric",2), y, p1, p2,
   truepauc, tol=.001, eps){
  nx <- length(x)
  ny <- length(y)
  est.pauc <- rep(1/nx, nx)%*% mean.pauc(x, ab[1], ab[2], eps, y)%*%
      rep(1/ny, ny)
  search.low <- ifelse((min(est.pauc, truepauc)-0.1) >0, (min(est.
     pauc, truepauc)-0.1), 0)
  search.high <- ifelse((max(est.pauc, truepauc)+0.1) <1 , (max(est.
     pauc, truepauc)+0.1), 1)
  grat <- (sqrt(5)-1)/2
  d <- grat * (search.high - search.low)
  x2 <- search.low + d
  x1 <- search.high - d
  res1 <- eltest2u(ab, x.sample=x, mu=c(1-p2, 1-p1, x1), y, vvec=rep
     (1/ny, ny), eps)
  res2 <- eltest2v(ab, x.sample=x, mu=truepauc, y, res1$uvec/nx, eps
     )
  fun.x1 <-  res1$"-2LLR" + res2$"-2LLR"
  res1 <- eltest2u(ab, x.sample=x, mu=c(1-p2, 1-p1, x2), y, vvec=rep
     (1/ny, ny), eps)
  res2 <- eltest2v(ab, x.sample=x, mu=truepauc, y, res1$uvec/nx, eps
     )
  fun.x2 <-  res1$"-2LLR" + res2$"-2LLR"
  err <- 100
  numit <- 0
  while (err > tol) {
    numit <- numit + 1
    if (fun.x1 < fun.x2){
      xopt <- x1
      err <- (1 - grat) * abs((search.high - search.low)/xopt)
      if (err > tol) {
        search.high <- x2
        x2 <- x1
        fun.x2 <- fun.x1
        d <- grat * (search.high - search.low)
        x1 <- search.high - d
        res1 <- eltest2u(ab, x.sample=x, mu=c(1-p2, 1-p1, x1), y,
           vvec=rep(1/ny, ny), eps)
        res2 <- eltest2v(ab, x.sample=x, mu=truepauc, y, res1$uvec/
           nx, eps)
```

```
        fun.x1 <-  res1$"-2LLR" + res2$"-2LLR"
      }
    }
    else {
      xopt <- x2
      err <- (1 - grat) * abs((search.high - search.low)/xopt)
      if (err > tol) {
        search.low <- x1
        x1 <- x2
        fun.x1 <- fun.x2
        d <- grat * (search.high - search.low)
        x2 <- search.low + d
        res1 <- eltest2u(ab, x.sample=x, mu=c(1-p2, 1-p1, x2), y,
           vvec=rep(1/ny, ny), eps)
        res2 <- eltest2v(ab, x.sample=x, mu=truepauc, y, res1$uvec/
           nx, eps)
        fun.x2 <-  res1$"-2LLR" + res2$"-2LLR"
      }
    }
  }
  res1 <- eltest2u(ab, x.sample=x, mu=c(1-p2, 1-p1, xopt), y, vvec=
     rep(1/ny, ny), eps)
  res2 <- eltest2v(ab, x.sample=x, mu=truepauc, y, res1$uvec/nx, eps
     )
  return("-2LLR"= res1$"-2LLR" + res2$"-2LLR")
}
```

neighb.xy

Empirical likelihood test for pAUC at (p1, p2), $0 < p1 < p2 < 1$

Description:

This function apples the profile algorithm described in Section 4.3

Usage:

neighb.xy(sp12 = vector("numeric", 2), x, y, true, tol = 0.001, eps)

Arguments:

sp12	a vector of two False Positive Rates between which the pAUC is tested, sp12=(p1, p2)
x	a sample of test results from non-diseased population
y	a sample of test results from diseased population
true	the pAUC to be tested
tol	the maximum distance between the current guess and the current range on Golden Section Search Optimization method
eps	a smoothing parameter

Details:

The empirical likelihood ratio test for pAUC at (p1, p2) is executed in two steps. First, test the hypotheses with three parameters, two of which are the nuisance parameters p1 and p2. Second, profile out the nuisance parameters p1 and p2. This function performs the second step - profiling out the nuisance parameters. This function searches the minimum of the empirical likelihood ratio test statistics among combinations of (1 - p1)-th quantile and (1 - p2)-th quantile with (1 - p1)-th quantile and (1 - p2)-th quantile moving along the consecutive non-diseased sample points separately. Since the test statistics is a piecewise constant function of the quantiles for (1 - p1) and (1 - p2), the minimum is easily found if searching starts at some strategical quantiles such as the sample quantiles at (1 - p1) and (1 - p2).

Value:

"-2LLR"	the -2 log likelihood ratio; approximate chi square distribution with one degree of freedom under H_0
Val	the observed P-value by chi square approximation

Example:

```
mux <- 0
stddx <- 1
muy <- 1
stddy <- 1
p1 <- 0.05
p2 <- 0.5
rp<-function(p, mux, stddx, muy, stddy)
  {
    y<-1-pnorm(qnorm(1-p, mux, stddx), muy, stddy)
    y
  }
##ture pAUC at (p1, p2)
truepauc<-integrate(rp, mux=mux, stddx=stddx, muy=muy, stddy=stddy,
   lower=p1,upper=p2)$value
nx <- 40
ny <- 30
x <- rnorm(nx, mux, stddx)
y <- rnorm(ny, muy, stddy)
neighb.xy(sp12=c(p1, p2), x=x, y=y, true=truepauc, tol=0.001, eps=1/
   nx)
```

Code

```
neighb.xy <- function (sp12 = vector("numeric", 2), x, y, true, tol
   = 0.001,
    eps)
```

```
{
    nx <- length(x)
    ny <- length(y)
    if (sp12[1] >= sp12[2])
        stop("the first sample quantile has to be smaller than the
            second")
    sortx_w <- myWdataclean2(z = x)
    sortedx <- sortx_w$value
    cx <- cumsum(sortx_w$weight/nx)
    idex1 <- ifelse(sp12[2] < 1, max(which(cx <= 1 - sp12[2])), 1)
    idex2 <- max(which(cx <= 1 - sp12[1]))
    abs <- data.frame(x1 = idex1, x2 = idex2)
    best <- eltest2step(ab = c(sortedx[idex1], sortedx[idex2]),
        x = x, y = y, p1 = sp12[1], p2 = sp12[2], true, tol,
        eps = eps)
    neg2.llr <- data.frame(value = best)
    ite <- TRUE
    while (ite == TRUE) {
        neib1 <- c((idex1 - 1), idex2)
        if (idex1 - 1 > 0) {
            if ((!any(abs[, 1] == neib1[1])) || (!any(which(neib1[1]
                ==
                abs[, 1]) %in% which(neib1[2] == abs[, 2])))) {
                abs <- rbind(abs, neib1)
                neg2.llr <- rbind(neg2.llr, eltest2step(ab = c(
                    sortedx[idex1 -
                  1], sortedx[idex2]), x = x, y = y, p1 = sp12[1],
                  p2 = sp12[2], true, tol, eps = eps))
            }
        }
        neib2 <- c(idex1, (idex2 - 1))
        if (idex2 - 1 > idex1) {
            if ((!any(abs[, 2] == neib2[2])) || (!any(which(neib2[1]
                ==
                abs[, 1]) %in% which(neib2[2] == abs[, 2])))) {
                abs <- rbind(abs, neib2)
                neg2.llr <- rbind(neg2.llr, eltest2step(ab = c(
                    sortedx[idex1],
                  sortedx[idex2 - 1]), x = x, y = y, p1 = sp12[1],
                  p2 = sp12[2], true, tol, eps = eps))
            }
        }
        neib3 <- c((idex1 + 1), idex2)
        if (idex1 + 1 < idex2) {
            if ((!any(abs[, 1] == neib3[1])) || (!any(which(neib3[1]
                ==
                abs[, 1]) %in% which(neib3[2] == abs[, 2])))) {
                abs <- rbind(abs, neib3)
                neg2.llr <- rbind(neg2.llr, eltest2step(ab = c(
                    sortedx[idex1 +
                  1], sortedx[idex2]), x = x, y = y, p1 = sp12[1],
                  p2 = sp12[2], true, tol, eps = eps))
            }
        }
```

```
neib4 <- c(idex1, (idex2 + 1))
if (idex2 + 1 < nx) {
    if ((!any(abs[, 2] == neib4[2])) || (!any(which(neib4[1]
        ==
        abs[, 1]) %in% which(neib4[2] == abs[, 2])))) {
        abs <- rbind(abs, neib4)
        neg2.llr <- rbind(neg2.llr, eltest2step(ab = c(
            sortedx[idex1],
          sortedx[idex2 + 1]), x = x, y = y, p1 = sp12[1],
          p2 = sp12[2], true, tol, eps = eps))
    }
}
neib5 <- c((idex1 - 1), (idex2 - 1))
if ((idex1 - 1 > 0) & (idex2 - 1 > idex1)) {
    if (!any(which(neib5[1] == abs[, 1]) %in% which(neib5[2]
        ==
        abs[, 2]))) {
        abs <- rbind(abs, neib5)
        neg2.llr <- rbind(neg2.llr, eltest2step(ab = c(
            sortedx[idex1 -
          1], sortedx[idex2 - 1]), x = x, y = y, p1 = sp12
              [1],
          p2 = sp12[2], true, tol, eps = eps))
    }
}
neib6 <- c((idex1 - 1), (idex2 + 1))
if ((idex1 - 1 > 0) & (idex2 + 1 < nx)) {
    if (!any(which(neib6[1] == abs[, 1]) %in% which(neib6[2]
        ==
        abs[, 2]))) {
        abs <- rbind(abs, neib6)
        neg2.llr <- rbind(neg2.llr, eltest2step(ab = c(
            sortedx[idex1 -
          1], sortedx[idex2 + 1]), x = x, y = y, p1 = sp12
              [1],
          p2 = sp12[2], true, tol, eps = eps))
    }
}
neib7 <- c((idex1 + 1), idex2 - 1)
if (idex1 + 1 < idex2 - 1) {
    if (!any(which(neib7[1] == abs[, 1]) %in% which(neib7[2]
        ==
        abs[, 2]))) {
        abs <- rbind(abs, neib7)
        neg2.llr <- rbind(neg2.llr, eltest2step(ab = c(
            sortedx[idex1 +
          1], sortedx[idex2 - 1]), x = x, y = y, p1 = sp12
              [1],
          p2 = sp12[2], true, tol, eps = eps))
    }
}
neib8 <- c((idex1 + 1), (idex2 + 1))
if ((idex1 + 1 < idex2) & (idex2 + 1 < nx)) {
```

```
            if (!any(which(neib8[1] == abs[, 1]) %in% which(neib8[2]
                ==
                abs[, 2]))) {
                abs <- rbind(abs, neib8)
                neg2.llr <- rbind(neg2.llr, eltest2step(ab = c(
                   sortedx[idex1 +
                  1], sortedx[idex2 + 1]), x = x, y = y, p1 = sp12
                     [1],
                  p2 = sp12[2], true, tol, eps = eps))
            }
        }
        if (min(neg2.llr) == best) {
            idex1 <- abs[which(neg2.llr == best), 1]
            idex2 <- abs[which(neg2.llr == best), 2]
            return(list(`-2LLR` = best, Pval = 1 - pchisq(best,
                df = 1)))
            ite <- FALSE
        }
        else {
            best = min(neg2.llr)
            idex1 <- abs[which(neg2.llr == best), 1]
            idex2 <- abs[which(neg2.llr == best), 2]
            ite <- TRUE
        }
    }
}
```

eltest2step.p12

Empirical likelihood test for percentile at $(1-p)$ *and pAUC at* $(0,\ p)$ *or* $(p,\ 1)$. *P is the false positive rate,* $0 < p < 1$

Description:

Compute the empirical likelihood ratio test with two parameters: percentiles at (1-p) and pAUC fixed at $pAUC_0$

Usage:

eltest2step.p12(x, highx, y, vp, truepauc, tol=.001, eps)

Arguments:

x	a sample of test results from non-diseased population
highx	the percentile at $(1-p)$, $0 < p < 1$
y	a sample of test results from diseased population
vp	vector of (0, p) or (p, 1), p is the false positive rate to be included in pAUC
truepauc	the pAUC to be tested
tol	the maximum distance between the current guess and the current range on Golden Section Search Optimization method
eps	a smoothing parameter

Details:

The function apples the 'ELseesaw' algorithm described in Section 4.2 to minimize the empirical likelihood ratio test. "-2LLR" may be positive infinite if the tested values of the parameters are far away from the true values.

Value:

"-2LLR"	the -2 loglikelihood ratio; approximate chi-square distribution with two degrees of freedom under the null hypotheses

Example:

```
p <- 0.4
rate.x <- 1
rate.y <- 0.25
rp.exp2<-function(p, rate.x, rate.y)
  {
    y<- 1- pexp(qexp(1-p, rate=rate.x), rate=rate.y)
    y
  }
p1 <- 0
p2 <- p
###true pAUC at (0, p)
truepauc <- integrate(rp.exp2, rate.x, rate.y,lower=p1,upper=p2)$
   value
highx <- qexp(1-p, rate=rate.x)
nx <- 50
ny <- 40
x <- rexp(nx, rate.x)
y <- rexp(ny, rate.y)
eltest2step.p12(vp =c(p1, p2), x=x, y=y, highx=highx, truepauc=
   truepauc, tol=0.001, eps=1/nx)
```

Code

```
eltest2step.p12 <- function (x, highx, y, vp, truepauc, tol = 0.001,
    eps)
```

```
{
    if (vp[1] == 0 & vp[2] > 0 & vp[2] < 1) {
        p <- vp[2]
        mean.fun <- mean.paucp2
    }
    if (vp[2] == 1 & vp[1] > 0 & vp[1] < 1) {
        p <- vp[1]
        mean.fun <- mean.paucp1
    }
    nx <- length(x)
    ny <- length(y)
    est.pauc <- rep(1/nx, nx) %*% mean.fun(x, highx, eps, y) %*%
        rep(1/ny, ny)
    search.low <- ifelse((min(est.pauc, truepauc) - 0.1) > 0,
        (min(est.pauc, truepauc) - 0.1), 0)
    search.high <- ifelse((max(est.pauc, truepauc) + 0.1) < 1,
        (max(est.pauc, truepauc) + 0.1), 1)
    grat <- (sqrt(5) - 1)/2
    d <- grat * (search.high - search.low)
    x2 <- search.low + d
    x1 <- search.high - d
    res1 <- eltest2up12(mean.fun, highx, x.sample = x, mu = c(1 -
        p, x1), y, vvec = rep(1/ny, ny), eps)
    res2 <- eltest2vp12(mean.fun, highx, x.sample = x, mu = truepauc
        ,
        y, res1$uvec/nx, eps)
    fun.x1 <- res1$"-2LLR" + res2$"-2LLR"
    res1 <- eltest2up12(mean.fun, highx, x.sample = x, mu = c(1 -
        p, x2), y, vvec = rep(1/ny, ny), eps)
    res2 <- eltest2vp12(mean.fun, highx, x.sample = x, mu = truepauc
        ,
        y, res1$uvec/nx, eps)
    fun.x2 <- res1$"-2LLR" + res2$"-2LLR"
    err <- 100
    numit <- 0
    while (err > tol) {
        numit <- numit + 1
        if (fun.x1 < fun.x2) {
            xopt <- x1
            err <- (1 - grat) * abs((search.high - search.low)/xopt)
            if (err > tol) {
                search.high <- x2
                x2 <- x1
                fun.x2 <- fun.x1
                d <- grat * (search.high - search.low)
                x1 <- search.high - d
                res1 <- eltest2up12(mean.fun, highx, x.sample = x,
                  mu = c(1 - p, x1), y, vvec = rep(1/ny, ny),
                  eps)
                res2 <- eltest2vp12(mean.fun, highx, x.sample = x,
                  mu = truepauc, y, res1$uvec/nx, eps)
                fun.x1 <- res1$"-2LLR" + res2$"-2LLR"
            }
        }
```

```
        else {
            xopt <- x2
            err <- (1 - grat) * abs((search.high - search.low)/xopt)
            if (err > tol) {
                search.low <- x1
                x1 <- x2
                fun.x1 <- fun.x2
                d <- grat * (search.high - search.low)
                x2 <- search.low + d
                res1 <- eltest2up12(mean.fun, highx, x.sample = x,
                  mu = c(1 - p, x2), y, vvec = rep(1/ny, ny),
                  eps)
                res2 <- eltest2vp12(mean.fun, highx, x.sample = x,
                  mu = truepauc, y, res1$uvec/nx, eps)
                fun.x2 <- res1$"-2LLR" + res2$"-2LLR"
            }
        }
    }
    res1 <- eltest2up12(mean.fun, highx, x.sample = x, mu = c(1 -
        p, xopt), y, vvec = rep(1/ny, ny), eps)
    res2 <- eltest2vp12(mean.fun, highx, x.sample = x, mu = truepauc
        ,
        y, res1$uvec/nx, eps)
    return(`-2LLR` = res1$"-2LLR" + res2$"-2LLR")
}
```

neighb.xy3p12

Empirical likelihood test for pAUC at [0, p) or (p, 1], $0 < p < 1$

Description:

This function applys the profile algorithm described in Section 4.3

Usage:

```
neighb.xy3p12(vp, x, y, true, tol = 0.001, eps, ...)
```

Arguments:

vp	a vector of the False Positive Rate cutting point for pAUC; vp = (0, p) or vp = (p, 1)
x	a sample of test results from non-diseased population
y	a sample of test results from diseased population
true	the pAUC to be tested
tol	the maximum distance between the current guess and the current range on Golden Section Search Optimization method
eps	a smoothing parameter
...	additional arguments, if any, to pass to the function

Details:

This function searches the minimum of the empirical likelihood ratio test statistics among the consecutive non-diseased sample points. Since the test statistics is a piecewise constant function of the quantile for (1 - p), the minimum is easily found if searching starts at some strategical quantile such as the sample quantile at (1 - p).

Value:

"-2LLR"	the -2 loglikelihood ratio; approximate chi square distribution with one degree of freedom under H_0
Pval	the observed P-value by chi square approximation

Example:

```
p <- 0.4
rate.x <- 1
rate.y <- 0.25
rp.exp2<-function(p, rate.x, rate.y)
  {
    y<- 1- pexp(qexp(1-p, rate=rate.x), rate=rate.y)
    y
  }
p1 <- 0
p2 <- p
###true pAUC at (0, p)
truepauc <- integrate(rp.exp2, rate.x, rate.y,lower=p1,upper=p2)$
   value
nx <- 50
ny <- 40
x <- rexp(nx, rate.x)
y <- rexp(ny, rate.y)
neighb.xy3p12(vp =c(p1, p2), x=x, y=y, true=truepauc, tol=0.001, eps
   =1/nx)
```

Code

```
neighb.xy3p12 <- function (vp, x, y, true, tol = 0.001, eps, ...)
{
    if (vp[1] == 0 & vp[2] > 0 & vp[2] < 1) {
        p <- vp[2]
    }
    if (vp[2] == 1 & vp[1] > 0 & vp[1] < 1) {
        p <- vp[1]
    }
    nx <- length(x)
    ny <- length(y)
    sortx_w <- myWdataclean2(z = x)
    sortedx <- sortx_w$value
```

```
    cx <- cumsum(sortx_w$weight/nx)
    idex <- ifelse(p < 1, max(which(cx <= 1 - p)), 1)
    abs <- data.frame(x = idex)
    highx = sortedx[idex]
    best <- eltest2step.p12(highx, x = x, y = y, vp = vp, truepauc =
        true,
        tol, eps)
    neg2.llr <- data.frame(value = best)
    ite <- TRUE
    while (ite == TRUE) {
        neib1 <- idex - 1
        if ((idex - 1 > 0) & (!any(abs == neib1))) {
            abs <- rbind(abs, neib1)
            highx = sortedx[neib1]
            neg2.llr <- rbind(neg2.llr, eltest2step.p12(highx,
                x = x, y = y, vp = vp, truepauc = true, tol,
                eps))
        }
        neib2 <- idex + 1
        if ((neib2 <= nx) & (!any(abs == neib2))) {
            abs <- rbind(abs, neib2)
            highx = sortedx[neib2]
            neg2.llr <- rbind(neg2.llr, eltest2step.p12(highx,
                x = x, y = y, vp = vp, truepauc = true, tol,
                eps))
        }
        if (min(neg2.llr) == best) {
            return(list(`-2LLR` = best, Pval = 1 - pchisq(best,
                df = 1)))
            ite <- FALSE
        }
        else {
            best = min(neg2.llr)
            idex <- abs$x[which(neg2.llr == best)]
            ite <- TRUE
        }
    }
}
```

eltest2step.all

Empirical likelihood test for AUC

Description:

Compute the empirical likelihood ratio with AUC fixed at AUC_0

Usage:

eltest2step.all(true, x, y, tol)

Arguments:

x	a sample of test results from non-diseased population
y	a sample of test results from diseased population
true	the AUC to be tested
tol	the maximum distance between the current guess and the current range on Golden Section Search Optimization method

Details:

This function searches the minimum of the empirical likelihood ratio test statistics among the consecutive non-diseased sample points. Since the test statistics is a piecewise constant function of the quantile for (1 - p), the minimum is easily found if searching starts at some strategical quantile such as the sample quantile at (1 - p).

Value:

"-2LLR"	the -2 loglikelihood ratio; approximate chi square distribution with one degree of freedom under H_0
Pval	the observed P-value by chi square approximation

Example:

```
rate.x <- 1
rate.y <- 0.25
rp.exp2<-function(p, rate.x, rate.y)
  {
    y<- 1- pexp(qexp(1-p, rate=rate.x), rate=rate.y)
    y
  }
p1 <- 0
p2 <- 1
###true AUC
truepauc <- integrate(rp.exp2, rate.x, rate.y,lower=p1,upper=p2)$
   value
nx <- 40
ny <- 30
x <- rexp(nx, rate.x)
y <- rexp(ny, rate.y)
eltest2step.all(x=x, y=y, true=truepauc)
```

Code

```
eltest2step.all <- function (true, x, y, tol = 0.001)
{
    nx <- length(x)
```

```
ny <- length(y)
est.pauc <- rep(1/nx, nx) %*% mean.whole(x, y) %*% rep(1/ny,
    ny)
search.low <- ifelse((min(est.pauc, true) - 0.1) > 0, (min(est.
   pauc,
    true) - 0.1), 0)
search.high <- ifelse((max(est.pauc, true) + 0.1) < 1, (max(est.
   pauc,
    true) + 0.1), 1)
grat <- (sqrt(5) - 1)/2
d <- grat * (search.high - search.low)
x2 <- search.low + d
x1 <- search.high - d
res1 <- eltest2u.all(x.sample = x, mu = x1, y, vvec = rep(1/ny,
    ny))
res2 <- eltest2v.all(x.sample = x, mu = true, y, res1$uvec/nx)
fun.x1 <- res1$"-2LLR" + res2$"-2LLR"
res1 <- eltest2u.all(x.sample = x, mu = x2, y, vvec = rep(1/ny,
    ny))
res2 <- eltest2v.all(x.sample = x, mu = true, y, res1$uvec/nx)
fun.x2 <- res1$"-2LLR" + res2$"-2LLR"
err <- 100
numit <- 0
while (err > tol) {
    numit <- numit + 1
    if (fun.x1 < fun.x2) {
        xopt <- x1
        err <- (1 - grat) * abs((search.high - search.low)/xopt)
        if (err > tol) {
            search.high <- x2
            x2 <- x1
            fun.x2 <- fun.x1
            d <- grat * (search.high - search.low)
            x1 <- search.high - d
            res1 <- eltest2u.all(x.sample = x, mu = x1, y,
              vvec = rep(1/ny, ny))
            res2 <- eltest2v.all(x.sample = x, mu = true,
              y, res1$uvec/nx)
            fun.x1 <- res1$"-2LLR" + res2$"-2LLR"
        }
    }
    else {
        xopt <- x2
        err <- (1 - grat) * abs((search.high - search.low)/xopt)
        if (err > tol) {
            search.low <- x1
            x1 <- x2
            fun.x1 <- fun.x2
            d <- grat * (search.high - search.low)
            x2 <- search.low + d
            res1 <- eltest2u.all(x.sample = x, mu = x2, y,
              vvec = rep(1/ny, ny))
            res2 <- eltest2v.all(x.sample = x, mu = true,
              y, res1$uvec/nx)
```

```
                fun.x2 <- res1$"-2LLR" + res2$"-2LLR"
            }
        }
    }
    res1 <- eltest2u.all(x.sample = x, mu = xopt, y, vvec = rep(1/ny
        ,
        ny))
    res2 <- eltest2v.all(x.sample = x, mu = true, y, res1$uvec/nx)
    fun.xopt <- res1$"-2LLR" + res2$"-2LLR"
    return(list(`-2LLR` = fun.xopt, Pval = 1 - pchisq(fun.xopt,
        df = 1)))
}
```

est.pAUC

Estimate pAUC/AUC by a non-parametric method

Description:

Compute the estimation of pAUC/AUC based on the equation in Detail

Usage:

est.pAUC(x, y, p, ...)

Arguments:

x	a sample of test results from non-diseased population
y	a sample of test results from diseased population
p	a vector with two elements: $p = (p1, p2)$ for pAUC at $(p1, p2)$, $0 \leq p1 < p2 \leq 1$; $p = (0, 1)$ for AUC.
...	additional arguments, if any, to pass to the function

Details:

the estimation of pAUC is based on equation (3.2).

Value:

the returned value is the estimate of pAUC/AUC

Example:

```
### An example for pAUC at (0.05, 0.55)
nx <- 50
rate.x <- 1
ny <- 50
```

```
rate.y <- 0.25
p1 <- 0.05
p2 <- 0.55
eps <- 1/nx
x <- rexp(nx, rate.x)
y <- rexp(ny, rate.y)
est.pAUC(x, y, p=c(p1, p2), eps=eps)

### An example for pAUC at (0, 0.45)
nx <- 50
ny <- 40
p1 <- 0
p2 <- 0.45
x <- rnorm(nx)
y <- rnorm(ny, 2, 1)
eps <- 1/nx
est.pAUC(x, y, p=c(p1, p2), eps=eps)

### An example for pAUC at (0.45, 1)
nx <- 100
ny <- 80
p1 <- 0.45
p2 <- 1
x <- rnorm(nx)
y <- rnorm(ny, 2, 1)
eps <- 1/nx
est.pAUC(x, y, p=c(p1, p2), eps=eps)

### An example for AUC
nx <- 40
ny <- 30
x <- rnorm(nx)
y <- rnorm(ny, 1, 1)
est.pAUC(x, y, p=c(0, 1))
```

Code

```
est.pAUC <- function (x, y, p, ...)
{
    sortedx <- x[order(x)]
    nx <- length(x)
    cx <- cumsum(rep(1/nx, nx))
    if (0 < p[1] & p[1] < p[2] & p[2] < 1) {
        lowx <- sortedx[max(which(cx <= 1 - p[2]))]
        highx <- sortedx[max(which(cx <= 1 - p[1]))]
        est <- rep(1/length(x), length(x)) %*% mean.pauc(x = x,
            y = y, lowx, highx, ...) %*% rep(1/length(y), length(y))
    }
    else if (p[1] == 0 & 0 < p[2] & p[2] < 1) {
        lowx <- sortedx[max(which(cx <= 1 - p[2]))]
        est <- rep(1/length(x), length(x)) %*% mean.paucp2(x = x,
            y = y, lowx, ...) %*% rep(1/length(y), length(y))
```

```
    }
    else if (p[2] == 1 & 0 < p[1] & p[1] < 1) {
        highx <- sortedx[max(which(cx <= 1 - p[1]))]
        est <- rep(1/length(x), length(x)) %*% mean.paucp1(x = x,
            y = y, highx, ...) %*% rep(1/length(y), length(y))
    }
    else if (p[1] == 0 & p[2] == 1) {
        est <- rep(1/length(x), length(x)) %*% mean.whole(x = x,
            y = y) %*% rep(1/length(y), length(y))
    }
    return(est)
}
```

find.UL

Find the Wilks Confidence Interval from the Given (empirical) Likelihood Ratio Function

Description:

This program uses simple search to find the upper and lower (Wilks) confidence limits based on the -2 log likelihood ratio, which the required input fun is supposed to supply.
Basically, starting from MLE, we search on both directions, by step away from MLE, until we find values that have -2LLR = level. (the value of -2LLR at MLE is supposed to be zero.) At current implementation, only handles one dimensional parameter, i.e. only confidence intervals, not confidence regions

Usage:

find.UL(step = 0.01, initStep = 0, fun, MLE, level = 3.84, ...)

Arguments:

step — a positive number. The starting step size of the search. Reasonable value should be about 1/5 of the SD of MLE

initStep — a nonnegative number. The first step size of the search. Sometimes, you may want to put a larger innitStep to speed the search

fun — a function that returns "-2LLR", which is the -2 log (empirical) likelihood ratio.

For pAUC/AUC: x is a non diseased sample; y is a diseased sample; true is the pAUC/AUC to be tested.
fun=neighb.xy(sp12 = vector("numeric",2), x, y, true, tol=.001, eps) for pAUC at [p1, p2] (p1 and p2 are false positive rates and $p1 < p2$) and $sp12 = (p1, p2)$;
neighb.xy3p12(vp, x, y, true, tol = 0.001, eps, ...) for pAUC at (p, 1] or [0, p). vp = (p, 1) or vp = (0, p), which is chosen based on the pAUC at (p, 1] or [0, p) accordingly;
fun=eltest2step.all(true, x, y, tol) for AUC.

For trimmed/truncated mean: x is a sample; true is the trimmed/truncated mean to be tested.
fun=neighb(sp12 = vector("numeric",2), x, true, eps) for trimmed mean at (p1, p2) ($0 < p1 < p2 < 1$), where sp12 = (p1, p2);
fun=neighb.p((p, x, true, eps)) for truncated mean at $[0, p)$ ($0 < p < 1$).

In all the above cases, eps is a smoothing parameter. tol is the maximum distance between the current guess and the current range on Golden Section Search Optimization method.

MLE — The MLE of the parameter. No need to be exact, as long as it is inside the confidence interval

level — an optional positive number, controls the confidence level. Default to 3.84 = chisq(0.95, df=1). Change to 2.70=chisq(0.90, df=1) to get a 90% confidence interval

... — additional arguments, if any, to pass to function

Details:

Basically we repeatedly testing the value of the parameter, until we find those which the -2 log likelihood value is equal to 3.84 (or other level, if set differently). If there is no value exactly equal to 3.84, we stop at the value which result a -2 log likelihood just below 3.84. (as in the discrete case, like quantiles.)

Value:

Low	the lower limit of the confidence interval
Up	the upper limit of the confidence interval
FstepL	the final step size when search lower limit. An indication of the precision
FstepU	Ditto. An indication of the precision of the upper limit
Lvalue	The -2LLR value of the final Low value. Should be approximately equal to level. If larger than level, than the confidence interval limit Low is wrong.
Uvalue	Ditto. Should be approximately equal to level

Example:

```
### An example for pAUC at (0.05, 0.55)
nx <- 50
rate.x <- 1
ny <- 50
rate.y <- 0.25
p1 <- 0.05
p2 <- 0.55
eps <- 1/nx
x <- rexp(nx, rate.x)
y <- rexp(ny, rate.y)
est <- est.pAUC(x, y, p=c(p1, p2), eps=eps)
find.UL(step=0.01, fun=neighb.xy, sp12=c(p1, p2), x=x, y=y, MLE=est,
    eps=eps)

### An example for pAUC at [0, 0.45)
nx <- 50
ny <- 40
p1 <- 0
p2 <- 0.45
x <- rnorm(nx)
y <- rnorm(ny, 2, 1)
eps <- 1/nx
est <- est.pAUC(x, y, p=c(p1, p2), eps=eps)
find.UL(step=0.01, fun=neighb.xy3p12, vp=c(p1, p2), MLE=est, x=x, y=
   y, eps=eps)

### An example for pAUC at (0.45, 1]
nx <- 100
ny <- 80
p1 <- 0.45
p2 <- 1
x <- rnorm(nx)
y <- rnorm(ny, 2, 1)
eps <- 1/nx
est <- est.pAUC(x, y, p=c(p1, p2), eps=eps)
find.UL(step=0.01, fun=neighb.xy3p12, vp=c(p1, p2), MLE=est, x=x, y=
   y, eps=eps)

### An example for AUC
```

```
nx <- 40
ny <- 30
x <- rnorm(nx)
y <- rnorm(ny, 1, 1)
est <- est.pAUC(x, y, p=c(0, 1))
find.UL(step=0.01, fun=eltest2step.all, MLE=est, x=x, y=y)

### An example of trimmed mean at (0.21, 0.79)
nsize <- 50
p1<- 0.21
p2<- 0.79
x <- rnorm(nsize)
est <- est.tr.mean(x, p=c(p1, p2), eps=1/nsize)
find.UL(step=0.01, fun=neighb, sp12=c(p1, p2), MLE=est, x=x, eps=1/
   nsize)

### An example of trunated mean at (0, 0.79)
nsize <- 50
p<- 0.89
x <- rnorm(nsize)
est <- est.tr.mean(x, p, eps=1/nsize)
find.UL(step=0.001, fun=neighb.p, p=p, MLE=est, x=x, eps=1/nsize)

### An example of Lorenz Curve at (0, 0.89)
nsize <- 500
p<- 0.89
x <- rexp(nsize, rate=0.25)
est <- est.tr.mean(x, p, eps=1/nsize)/mean(x)
find.UL(step=0.001, fun=neighb.p.lc, p=p, MLE=est, x=x, eps=1/nsize)
```

Code

```
find.UL <- function (step = 0.01, initStep = 0, fun, MLE, level =
   3.84, ...)
{
    value <- 0
    step1 <- step
    Lbeta <- MLE - initStep
    for (i in 1:8) {
        while (value < level) {
            Lbeta <- Lbeta - step1
            val <- fun(true = Lbeta, ...)
            value <- val$"-2LLR"
        }
        Lbeta <- Lbeta + step1
        step1 <- step1/10
        val <- fun(true = Lbeta, ...)
        value <- val$"-2LLR"
    }
    value1 <- value
    value <- 0
    Ubeta <- MLE + initStep
```

```
    for (i in 1:8) {
        while (value < level) {
            Ubeta <- Ubeta + step
            val <- fun(true = Ubeta, ...)
            value <- val$"-2LLR"
        }
        Ubeta <- Ubeta - step
        step <- step/10
        val <- fun(true = Ubeta, ...)
        value <- val$"-2LLR"
    }
    if ((value1 > level) | (value > level))
        warning("Something wrong. Check the MLE and step inputs.")
    return(list(Low = Lbeta, Up = Ubeta, FstepL = step1, FstepU =
       step,
        Lvalue = value1, Uvalue = value))
}
```

Internal Functions

Among the 13 internal functions, "myWdataclean2" is from Barton (2010) and "myfun5" is from r package 'emplik' by Zhou and Yang (2014). The annotated r codes of each internal function are as follows:

```
#####################################################################
# myWdataclean2 sorts the data, keeps the individual value, and
# saves the number of tied values as the weights.
#####################################################################
myWdataclean2<-function (z, wt = rep(1, length(z))) {
  niceorder <- order(z)
  sortedz <- z[niceorder]
  sortedw <- wt[niceorder]
  n <- length(sortedz)
  #y checks for jumps in sortedz using offsets of sortedz
  y <- sortedz[-1] != sortedz[-n]
  #ind stores jump indices (final index will be n)
  ind <- c(which(y | is.na(y)), n)
  #csum is cumulative sum of the weights
  csumw <- cumsum(sortedw)
  #value contains the (unique) obs in sortedz
  #weight has the weights of the obs in sortedz
  list(value = sortedz[ind], weight = diff(c(0,csumw[ind])))
}

#####################################################################
# myfun5 is a function for smoothing the cumulative dist fun
# F(X < theta), eps is the smoothing parameter
#####################################################################
myfun5 <- function(x, theta, eps) {
  u <- (x-theta)*sqrt(5)/eps
  INDE <- (u < sqrt(5)) & (u > -sqrt(5))
  u[u >= sqrt(5)] <- 0
```

```
  u[u <= -sqrt(5)] <- 1
  y <- 0.5 - (u - (u)^3/15)*3/(4*sqrt(5))
  u[ INDE ] <- y[ INDE ]
  return(u)
}

#####################################################################
# mean.pauc is a function for (Y > X)I(lowx < X <= highx)
# mean.pauc returns a maxtrix with dimensions of length(X) by length
   (Y)
# mean.pauc calls myfun5
# Y is a diseased sample
# X is a non diseased sample
# [lowx, highx] is X range for the partial AUC
# lowx is the (1 - p2)th quantile of X sample
# high is the (1 - p1)th quantile of X sample
# [p1, p2] is the range of FPR of the pAUC
#####################################################################

mean.pauc<- function(x, lowx, highx, eps, y){
  outer(x, y, FUN="<")* ((1-myfun5(x, lowx, eps)) * myfun5(x, highx,
      eps))
}

#####################################################################
# eltest2u is a function to call el.test in package "emplik" to
   compute -2LLR by
# assuming a fixed mean vector (mu) when Y probability equals to
   vvec
# eltest2u also returns X probability that minimizes -2LLR
# ab is a vector with 2 elements for the X range of the partial AUC.
    ab[1] < ab[2]
# x.sample is a non diseased sample
# y is a diseased sample
# mu is a vector with 3 elements (1-p2, 1-p1, pAUC)
# vvec is generally assumed as empirical distribution of Y sample.
#####################################################################
eltest2u <- function(ab = vector("numeric",2), x.sample, mu, y, vvec
   , eps )
{
  axb <- matrix(c(myfun5(x.sample, ab[1], eps), myfun5(x.sample, ab
     [2], eps), mean.pauc(x.sample, ab[1], ab[2], eps, y)%*%vvec),
     ncol=3)
  all <- el.test(axb, mu)
  list('-2LLR'=all$`-2LLR`, uvec=all$wts)
}

#####################################################################
# eltest2v is a function to call el.test in package "emplik" to
   compute -2LLR with
# a fixed partial pAUC at X probability equals to uvec
# ab is a vector with 2 elements for the X range of the partial AUC.
    ab[1] < ab[2]
# x.sample is a non diseased sample
```

```
# y is a diseased sample
# mu is pAUC
# uvec is generally returned from eltest2u
#########################################################################
eltest2v <- function(ab = vector("numeric",2), x.sample, mu, y, uvec
    , eps )
{
  axb <- matrix(uvec%*%mean.pauc(x.sample, ab[1], ab[2], eps, y),
     ncol=1)
  all <- el.test(axb, mu)
  list('-2LLR'=all$`-2LLR`, vvec=all$wts)
}

#########################################################################
# mean.paucp1 is a function for (Y > X)I(X <= highx), which is the
    pAUC at [p1, 1]
# mean.paucp1 returns a maxtrix with dimensions of length(X) by
    length(Y)
# mean.paucp1 calls myfun5
# Y is a diseased sample
# X is a non diseased sample
# (-\infty, highx] is X range for the partial AUC
# highx is the (1 - p1)th quantile of X sample
#########################################################################

mean.paucp1<- function(x, highx, eps, y){
  outer(x, y, FUN="<=")* myfun5(x, highx, eps)
}

#########################################################################
# mean.paucp2 is a function for (Y > X)I(X > lowx), which is the
    pAUC at [0, p2]
# mean.paucp2 returns a maxtrix with dimensions of length(X) by
    length(Y)
# mean.paucp2 calls myfun5
# Y is a diseased sample
# X is a non diseased sample
# [lowx, \infty) is X range for the partial AUC
# lowx is the (1 - p2)th quantile of X sample
#########################################################################

mean.paucp2<- function(x, lowx, eps, y){
  outer(x, y, FUN="<")* (1 - myfun5(x, lowx, eps))
}

#########################################################################
# eltest2up12 is a function to call el.test in package "emplik" to
    compute -2LLR by
# assuming a fixed mean vector (mu) when Y probability equals to
    vvec
# eltest2up12 also returns X probability that minimizes -2LLR
# mean.fun is mean.paucp1 or mean.paucp2, which is chosen based on
    the pAUC at [p1, 1] or [0, p2]
# mu is a vector with 2 elements (1-p1, pAUC) or (1-p2, pAUC)
```

```
# x.sample is a non diseased sample
# y is a diseased sample
# vvec is generally assumed as empirical distribution of Y sample
# highx is the (1 - p1)th quantile or the (1 - p2)th quantile
########################################################################
eltest2up12 <- function(mean.fun, x.sample, highx, mu, y, vvec, eps,
    ... )
{
  axb <- matrix(c(myfun5(x.sample, highx, eps), mean.fun(x.sample,
     highx, eps, y)%*%vvec), ncol=2)
  all <- el.test(axb, mu)
  list('-2LLR'=all$`-2LLR`, uvec=all$wts)
}

########################################################################
# eltest2vp12 is a function to call el.test in package "emplik" to
   compute -2LLR with
# a fixed partial pAUC at X probability equals to uvec
# mean.fun is mean.paucp1 or mean.paucp2, which is chosen based on
   the pAUC at [p1, 1] or [0, p2]
# x.sample is a non diseased sample
# y is a diseased sample
# mu is pAUC
# uvec is generally returned from eltest2up12
# highx is the (1 - p1)th quantile or the (1 - p2)th quantile
########################################################################
eltest2vp12 <- function(mean.fun, x.sample, highx, mu, y, uvec, eps,
    ... )
{
  axb <- matrix(uvec%*%mean.fun(x.sample, highx, eps, y), ncol=1)
  all <- el.test(axb, mu)
  list('-2LLR'=all$`-2LLR`, vvec=all$wts)
}

########################################################################
# mean.whole is a function for (Y > X), which is the AUC at [0, 1]
# mean.whole returns a matrix with dimensions of length(X) by length
   (Y)
# Y is a diseased sample
# X is a non diseased sample
########################################################################
mean.whole<- function(x, y){
  outer(x, y, FUN="<")
}

########################################################################
# eltest2u.all is a function to call el.test in package "emplik" to
   compute -2LLR by
# assuming a fixed mean AUC (mu) when Y probability equals to vvec
# eltest2u.all also returns X probability that minimizes -2LLR
# mu is the AUC
# x.sample is a non diseased sample
# y is a diseased sample
# vvec is generally assumed as empirical distribution of Y sample
```

```
#####################################################################

eltest2u.all <- function(x.sample, mu, y, vvec )
{
  all <- el.test(mean.whole(x.sample,  y)%*%vvec, mu)
  list('-2LLR'=all$`-2LLR`, uvec=all$wts)
}
#####################################################################
# eltest2v.all is a function to call el.test in package "emplik" to
   compute -2LLR with
# a fixed pAUC at X probability equals to uvec
# x.sample is a non diseased sample
# y is a diseased sample
# mu is pAUC
# uvec is generally returned from eltest2up12
#####################################################################

eltest2v.all <- function(x.sample, mu, y, uvec )
{
  axb <- matrix(uvec%*%mean.whole(x.sample,  y), ncol=1)
  all <- el.test(axb, mu)
  list('-2LLR'=all$`-2LLR`, vvec=all$wts)
}
```

Chapter 5 Future Work

This dissertation research could be extended in the future as the following:

1. Censoring data, which is the partial known value of a observation, exists in many situations. For example, in a clinical trial with overall survival as the end point, we may only know that an alive patient's survival time (T) is longer than certain observed number (t) by the end of the clinical trial, i.e., $T > t$. Censoring data exists in economics as well, for example, a wealthy family's income is usually known as more than a certain number. One of the future work of this research is to include the censoring data in our inference method such that we can do the inference on the trimmed mean survival time or the generalized Lorenz Curve with censoring data.

2. The other future work of this research on Lorenz Curve is to draw the confidence band of Lorenz Curve and do the statistical inference on Gini index.

3. Diagnostic tests are often influenced by subjects' age, gender, and other covariates. Our analysis in this thesis of pAUC representing the accuracy of a diagnostic test at some area does not include the influences by covariates. The future research is to include covariates so that we can do regression analysis on pAUC.

4. The computation speed of R package 'pAUC' can be improved if the loops in R can be written in C.

Appendix

R codes for Simulations in Chapter 2

R code for Figure 2.1

```
set.seed(123)
maxit <- 400
nsize <- 50
p1<- 0.21
p2<- 0.79
q1 <- qnorm(p1)
q2 <- qnorm(p2)
rp_trim <- function(x){
  y <- x*dnorm(x)
  y
}
omu_trim <- integrate(rp_trim, lower=q1, upper=q2)$value
mu_trim <- c(p1, p2, omu_trim)
chisq_trim3 <- numeric(0)
chisq_trim1 <- numeric(0)
for (ite in 1:maxit){
    x <- rnorm(nsize)
    res_trim3 <- eltest(c(q1, q2), x, mu=mu_trim, eps=1/nsize)
    chisq_trim3[ite] <- res_trim3$'-2LLR'
    res_trim1 <- neighb(sp12=c(p1, p2), x=x, true=omu_trim, eps=1/
       nsize)
    chisq_trim1[ite] <- res_trim1$'-2LLR'
}
par(mfrow = c(1, 2))
plot(qchisq(1:maxit/(maxit+1), df=3), sort(chisq_trim3), xlab="Chisq
   (3) Quantiles", ylab="Sorted -2LLR values")
abline(a=0, b=1)
text(1.5, 16, "P=(0.21, 0.79)", cex=0.85, adj=0)
text(1.5, 14, "X: N(0, 1), N=50", cex=0.85, adj=0)
text(1.5, 12, bquote(paste(H[0], ": ", eq(2.4))), cex=0.85, adj=0)

plot(qchisq(1:maxit/(maxit+1), df=1), sort(chisq_trim1), xlab="Chisq
   (1) Quantiles", ylab="Sorted -2LLR values")
abline(a=0, b=1)
text(1.5, 8, "P=(0.21, 0.79)", cex=0.85, adj=0)
text(1.5, 7, "X: N(0, 1), N=50", cex=0.85, adj=0)
text(1.5, 6, bquote(paste(H[0], ": ", mu[T])), cex=0.85, adj=0)
```

R code for Figures 2.2 and 2.3

```
set.seed(123)
maxit <- 800
```

```
nsize <- 500
p<- 0.89
rate <- 0.25
q <- qexp(p, rate)
rp_glc <- function(x){
  y <- x*dexp(x, rate=0.25)
  y
}
omu_glc <- integrate(rp_glc, lower= 0, upper=q)$value
mu_glc <- c(p, omu_glc)
lc <- omu_glc*rate
chisq_glc2 <- numeric(0)
chisq_glc1 <- numeric(0)
chisq_lc2 <- numeric(0)
chisq_lc1 <- numeric(0)
for (ite in 1:maxit){
  x <- rexp(n=nsize, rate=rate)
  res_glc2 <- eltest.p(q, x, mu=mu_glc, eps=1/nsize)
  chisq_glc2[ite] <- res_glc2$'-2LLR'
  res_glc1 <- neighb.p(p, x=x, true=omu_glc, eps=1/nsize)
  chisq_glc1[ite] <- res_glc1$'-2LLR'
  res_lc2 <- eltest.p.lc(q, x, mu=c(p, lc) , eps=1/nsize)
  chisq_lc2[ite] <- res_lc2$'-2LLR'
  res_lc1 <- neighb.p.lc(p, x=x, true=lc, eps=1/nsize)
  chisq_lc1[ite] <- res_lc1$'-2LLR'
}
par(mfrow = c(1, 2))
plot(qchisq(1:(length(chisq_glc2[chisq_glc2<50]))/(length(chisq_glc2
    [chisq_glc2<50])+1),df=2), sort(chisq_glc2[chisq_glc2<50]), xlab=
    "Chisq(2) Quantiles", ylab="Sorted -2LLR values")
abline(a=0, b=1)
text(6, 3, "P=0.89", cex=0.85, adj=0)
text(6, 2, "X: exp(rate=0.25), N=500", cex=0.85, adj=0)
text(6, 1, bquote(paste(H[0], ": ", eq(2.17))), cex=0.85, adj=0)

plot(qchisq(1:(length(chisq_glc1[chisq_glc1<50]))/(length(chisq_glc1
    [chisq_glc1<50])+1),df=1), sort(chisq_glc1[chisq_glc1<50]), xlab=
    "Chisq(1) Quantiles", ylab="Sorted -2LLR values")
abline(a=0, b=1)
text(5, 2.6, "P=0.89", cex=0.85, adj=0)
text(5, 1.8, "X: exp(rate=0.25), N=500", cex=0.85, adj=0)
text(5, 1, bquote(paste(H[0], ": ", GLC[p])), cex=0.85, adj=0)


par(mfrow = c(1, 2))
plot(qchisq(1:(length(chisq_lc2[chisq_lc2<50]))/(length(chisq_lc2[
    chisq_lc2<50])+1),df=2), sort(chisq_lc2[chisq_lc2<50]), xlab="
    Chisq(2) Quantiles", ylab="Sorted -2LLR values")
abline(a=0, b=1)
text(1.4, 14, "P=0.89", cex=0.85, adj=0)
text(1.4, 13, "X: exp(rate=0.25), N=500", cex=0.85, adj=0)
text(1.4, 12, bquote(paste(H[0], ": eq(2.23)")), cex=0.85, adj=0)
```

```
plot(qchisq(1:(length(chisq_lc1[chisq_lc1<50]))/(length(chisq_lc1[
    chisq_lc1<50])+1),df=1), sort(chisq_lc1[chisq_lc1<50]), xlab="
    Chisq(1) Quantiles", ylab="Sorted -2LLR values")
abline(a=0, b=1)
text(1.5, 11, "P=0.89", cex=0.85, adj=0)
text(1.5, 10, "X: exp(rate=0.25), N=500", cex=0.85, adj=0)
text(1.5, 9, bquote(paste(H[0], ": ", LC[p])), cex=0.85, adj=0)
```

R code for Table 2.1

```
library(pAUC)
trim.mean <- function(maxit=1000, nsize=60, true.m=0, t.scale=1, p1
    =0.1, p2=0.9, eps=1/sqrt(nsize)){
all.x <- numeric(nsize)
results <- matrix(NA, maxit, 3)
for (i in 1:maxit){
  x <- rlogis(nsize, true.m, t.scale)
  niceorder <- order(x)
  sortedx <- x[niceorder]
  all.x <- cbind(all.x, x)
  el.est <- est.tr.mean(x.sample=x, p=c(p1, p2), eps=eps)
  el.ci <- find.UL(step=0.01, initStep=0, fun=neighb, sp12=c(p1, p2)
      , MLE=el.est, x=x, eps=eps, level=qchisq(0.95, 1))
  results[i, 1] <- (true.m >= el.ci$Low/(p2-p1)) & (true.m <= el.ci$
     Up/(p2-p1))
  results[i, 2] <- (el.ci$Up - el.ci$Low)/(p2-p1)
  temp <- neighb(sp12=c(p1, p2), x=x, true=true.m*(p2-p1), eps=eps)
  results[i, 3] <- temp$'-2LLR'
 }
 return(list(logist.sample=all.x[, 2:1001], results=results))
}

set.seed(123)
mean40 <- trim.mean(nsize=40, eps=1/sqrt(40))
write.csv(mean40$logist.sample, file="C:/Users/C173518/Downloads/
    logist40sample2.csv")
save(mean40, file="C:/Users/C173518/Downloads/logist40result2.RData"
    )
set.seed(123)
mean60 <- trim.mean(nsize=60, eps=1/sqrt(60))
write.csv(mean60$logist.sample, file="C:/Users/C173518/Downloads/
    logist60sample2.csv")
save(mean60, file="C:/Users/C173518/Downloads/logist60result2.RData"
    )
set.seed(123)
mean80 <- trim.mean(nsize=80, eps=1/sqrt(80))
write.csv(mean80$logist.sample, file="C:/Users/C173518/Downloads/
    logist80sample2.csv")
save(mean80, file="C:/Users/C173518/Downloads/logist80result2.RData"
    )
```

SAS code for Table 2.1

```
%macro comp();
%do i=2 %to 1001;
```

```
ods output TrimmedMeans = trm&i WinsorizedMeans = wsm&i;
proc univariate data=two trim=0.1 (type=twosided) winsor=0.1(type=
   twosided);
  var var&i;
run;
%end;
%mend comp;

%macro trim_winsor(sample=);
dm log 'clear' ; dm output 'clear';
proc import datafile="C:\Users\C173518\Downloads\&sample..csv"
  out=one
  dbms=csv replace;
  getnames=yes;
run;

data two;
  set one;
  drop var1;
  rename x=var2;
run;
%comp();
data alltrm;
length varname $10;
  set trm:;
  if LCLMean<=0<= UCLMean then p=1;
  else p=0;
  l=UCLMean-LCLMean;
run;
data allwsm;
length varname $10;
  set wsm:;
  if LCLMean<=0<= UCLMean then p=1;
  else p=0;
  l=UCLMean-LCLMean;
run;
ods pdf style=journal file="C:\Users\C173518\Downloads\&sample..pdf
   ";
title "Winsorized Mean";
proc means data=allwsm;
  var p l;
run;
title "Trimmed Mean";
proc means data=alltrm;
  var p l;
run;
ods pdf close;
proc datasets library=work;
  delete one two alltrm allwsm wsm: trm:;
run;quit;
%mend;

%trim_winsor(sample=logist40sample2);
%trim_winsor(sample=logist60sample2);
```

```
%trim_winsor(sample=logist80sample2);
```

R code for Figure 2.4

```
library(pAUC)
library(ggplot2)
cps <- read.csv(file = "C:\\Users\\yumin\\Documents\\Research\\
    succeeded\\cps06.csv", header = TRUE)
cps0 <- subset(cps, HHINCOME >= 0)
indiana <- subset(cps0, STATECENSUS == 'Indiana')
ky <- subset(cps0, STATECENSUS == 'Kentucky')
income.in <- indiana$HHINCOME
income.ky <- ky$HHINCOME
indi <- matrix(NA, nrow=9, ncol=3)
kent <- matrix(NA, nrow=9, ncol=3)
eps.in <- 1/length(income.in)
eps.ky <- 1/length(income.ky)
for (i in 1:9){
  p <- i/10;
  est.in <- est.tr.mean(income.in, p, eps.in)/mean(income.in)
  ul.in <- find.UL(step=0.001, fun=neighb.p.lc, p=p, MLE=est.in, x=
      income.in, eps=eps.in)
  indi[i,] <- c(est.in, ul.in$Low, ul.in$Up)
  est.ky <- est.tr.mean(income.ky, p, eps.ky)/mean(income.ky)
  ul.ky <- find.UL(step=0.001, fun=neighb.p.lc, p=p, MLE=est.ky, x=
      income.ky, eps=eps.ky)
  kent[i,] <- c(est.ky, ul.ky$Low, ul.ky$Up)
}
ind <- data.frame(matrix(c(seq(0, 1, 0.1), 0, indi[,1], 1, 0, indi
    [,2], 0, 0, indi[,3], 0), ncol=4, dimnames=list(NULL, c('p', 'est
    ', 'low', 'up'))))
ken <- data.frame(matrix(c(seq(0, 1, 0.1), 0, kent[,1], 1, 0, kent
    [,2], 0, 0, kent[,3], 0), ncol=4, dimnames=list(NULL, c('p', 'est
    ', 'low', 'up'))))
ind$state <- rep('IN', 11)
ken$state <- rep('KY', 11)
all <- rbind(ind, ken)
par(lwd=2)
ggplot(all, aes(x=p, y=est, colour=state, group=state)) + geom_
    errorbar(aes(ymin=low, ymax = up), width=0.01) + geom_line() +
    geom_point() + ylab('LC(p)') + theme_bw() + theme(legend.
    justification=c(1,0), legend.position=c(1,0))
```

R codes for Simulations in Chapter 3

R code for Figure 3.1

```
set.seed(123)
rp<-function(p, mux, stddx, muy, stddy)
{
  y<-1-pnorm(qnorm(1-p, mux, stddx), muy, stddy)
  y
```

```
}
#AUC for N(1.2, 0.8) and N(0,1)
auc1 <- integrate(rp, mux=0, stddx=1, muy=1.2, stddy=0.8,lower=0,
   upper=1)$value
#AUC for N(2, 1.8) and N(0,1.2)
auc2 <- integrate(rp, mux=0, stddx=1.2, muy=2.027, stddy=1.8,lower
   =0,upper=1)$value
x <- seq(0, 1, by=0.01)
c <- qnorm(1-x)
y <- 1 - pnorm(c, 1.2, 0.8)
x1 <- seq(0, 1, by=0.01)
c1 <- qnorm(1-x1, 0, 1.2)
y1 <- 1 - pnorm(c1, 2.027, 1.8)
plot(x, y, type='l', xlab='1 - Specificity (FPR)', ylab='Sensitivity
    (TPR)' )
abline(a=0, b=1)
lines(x1, y1)
lines(x=c(0.5, 0.5), y=c(0, (1 - pnorm(qnorm(0.5), 1.2, 0.8))))
lines(x=c(0.7, 0.7), y=c(0, (1 - pnorm(qnorm(0.3), 1.2, 0.8))))
text(x=0.15, y=0.5, 'A', cex=0.8)
text(x=0.05, y=0.6, 'B', cex=0.8)
text(x=0.5, y=0, expression(paste('p'[1], '=0.5')), cex=0.7)
text(x=0.7, y=0, expression(paste('p'[2], '=0.7')), cex=0.7)
```

R code for Figure 3.2 and Figure 3.3

```
# A function to compute the test statistics of pAUC(p1, p2) and 0<p1
   <p2<1 based on normal samples

norm.simu.p12 <- function(mux, stddx, nx, muy, stddy, ny, p1, p2,
   maxit=1000, tol=.001, eps){
  rp<-function(p, mux, stddx, muy, stddy)
  {
    y<-1-pnorm(qnorm(1-p, mux, stddx), muy, stddy)
    y
  }
  truepauc<-integrate(rp, mux=mux, stddx=stddx, muy=muy, stddy=stddy
     ,lower=p1,upper=p2)$value
  t1 <- qnorm(1-p2, mux, stddx)
  t2 <- qnorm(1-p1, mux, stddx)
  chisp <- numeric(maxit)
  chisp3 <- numeric(maxit)
  for (ite in 1:maxit){
    x <- rnorm(nx, mux, stddx)
    y <- rnorm(ny, muy, stddy)
    chisp3[ite] <- eltest2step(x=x, ab=c(t1, t2), y=y, p1=p1, p2=p2,
        truepauc=truepauc, tol, eps)
    temp <- neighb.xy(sp12=c(p1, p2), x=x, y=y, true=truepauc, tol,
       eps)
    chisp[ite] <- temp$'-2LLR'
  }
  return(list(chisp=chisp, chisp3=chisp3))
}
```

```
# A function to compute the test statistics of pAUC[0, p) or pAUC(p,
    1] based on normal samples

norm.simu.p_1 <- function(mux, stddx, nx, muy, stddy, ny, p1, p2,
   maxit=1000, tol=.001, eps){
  rp<-function(p, mux, stddx, muy, stddy)
  {
    y<-1-pnorm(qnorm(1-p, mux, stddx), muy, stddy)
    y
  }
  truepauc<-integrate(rp, mux=mux, stddx=stddx, muy=muy, stddy=stddy
     ,lower=p1,upper=p2)$value
  t <- qnorm(1-p1, mux, stddx)
  chisp <- numeric(maxit)
  chisp2 <- numeric(maxit)
  for (ite in 1:maxit){
    x <- rnorm(nx, mux, stddx)
    y <- rnorm(ny, muy, stddy)
    chisp2[ite] <- eltest2step.p12(x=x, t, y=y, vp=c(p1, p2),
       truepauc=truepauc, tol, eps)
    temp <- neighb.xy3p12(vp=c(p1, p2), x=x, y=y, true=truepauc, tol
       , eps)
    chisp[ite] <- temp$'-2LLR'
  }
  return(list(chisp=chisp, chisp2=chisp2))
}

# A function to compute the test statistics of pAUC(p1, p2) and 0<p1
   <p2<1 based on exponential samples

norm.simu.p12 <- function(mux, stddx, nx, muy, stddy, ny, p1, p2,
   maxit=1000, tol=.001, eps){
exp.simu.p12 <- function(rate.x, nx, rate.y, ny, p1, p2, maxit=1000,
    tol=.001, eps){
  rp.exp2<-function(p, rate.x, rate.y)
  {
    y<- 1- pexp(qexp(1-p, rate=rate.x), rate=rate.y)
    y
  }
  truepauc <- integrate(rp.exp2, rate.x, rate.y, lower=p1, upper=p2)
     $value
  t1 <- qexp(1-p2, rate.x)
  t2 <- qexp(1-p1, rate.x)
  chisp <- numeric(maxit)
  chisp3 <- numeric(maxit)
  for (ite in 1:maxit){
    x <- rexp(nx, rate.x)
    y <- rexp(ny, rate.y)
    chisp3[ite] <- eltest2step(x=x, ab=c(t1, t2), y=y, p1=p1, p2=p2,
        truepauc=truepauc, tol, eps)
    temp <- neighb.xy(sp12=c(p1, p2), x=x, y=y, true=truepauc, tol,
       eps)
    chisp[ite] <- temp$'-2LLR'
  }
```

```
    return(list(chisp=chisp, chisp3=chisp3))
}

# A function to compute the test statistics of pAUC[0, p) or pAUC(p,
    1] based on exponential samples

exp.simu.p2 <- function(rate.x, nx, rate.y, ny, p1, p2, maxit=1000,
   tol=.001, eps){
  rp.exp2<-function(p, rate.x, rate.y)
  {
    y<- 1- pexp(qexp(1-p, rate=rate.x), rate=rate.y)
    y
  }
  truepauc <- integrate(rp.exp2, rate.x, rate.y,lower=p1,upper=p2)$
     value
  t <- qexp(1-p2, rate.x)
  chisp <- numeric(maxit)
  chisp2 <- numeric(maxit)
  for (ite in 1:maxit){
    x <- rexp(nx, rate.x)
    y <- rexp(ny, rate.y)
    chisp2[ite] <- eltest2step.p12(x=x, t, y=y, vp=c(p1, p2),
       truepauc=truepauc, tol, eps)
    temp <- neighb.xy3p12(vp=c(p1, p2), x=x, y=y, true=truepauc, tol
       , eps)
    chisp[ite] <- temp$'-2LLR'
  }
    return(list(chisp=chisp, chisp2=chisp2))
}

set.seed(123)
ptm <- proc.time()
chisp_n55_95<-norm.simu.p12(mux=0, stddx=1, nx=95, muy=1, stddy=1,
   ny=85, p1=0.55, p2=0.90, maxit=1000, eps=1/sqrt(95))
save(chisp_n55_95, file="chisp_n55_95")
proc.time() - ptm

ptm <- proc.time()
chisp_n60_100<-norm.simu.p_1(mux=0, stddx=1, nx=100, muy=1, stddy=1,
    ny=95, p1=0.55, p2=1, maxit=1000, eps=1/10)
save(chisp_n60_100, file="chisp_n60_100")
proc.time() - ptm

set.seed(123)
ptm <- proc.time()
chisp1exp05_50 <- exp.simu.p12(rate.x=1, nx=75, rate.y=0.25, ny=65,
   p1=0.2, p2=0.5, maxit=1000, eps=1/sqrt(75))
save(chisp1exp05_50, file="chisp1exp05_50")
proc.time() - ptm

ptm <- proc.time()
chisp1exp00_45 <- exp.simu.p2(rate.x=1, nx=50, rate.y=0.25, ny=40,
   p1=0, p2=0.45, maxit=1000, eps=1/sqrt(50))
save(chisp1exp00_45, file="chisp1exp00_45")
```

```
proc.time() - ptm

###############chi_sqare_2_3.png###############
par(mfrow=c(2,2), mar=c(2.5, 2.5, 0, 0.5), mgp=c(0.95, 0.3, 0), cex.
    lab=0.75, cex.axis=0.65, cex.main=0.65, tck=0.01)

plot(qchisq(1:1000/(1000+1),df=3), sort(chisp1exp05_50$chisp3), xlab
    ="Chisq(3) Quantiles", ylab="Sorted -2LLR Values")
abline(a=0, b=1)
text(1.5, 10, "P=(0.2, 0.5)", cex=0.75, adj=0)
text(1.5, 9, "X: exp(1), m=75", cex=0.75, adj=0)
text(1.5, 8, "Y: exp(0.25), n=65", cex=0.75, adj=0)

plot(qchisq(1:1000/(1000+1),df=3), sort(chisp_n55_95$chisp3), xlab="
    Chisq(3) Quantiles", ylab="Sorted -2LLR Values")
abline(a=0, b=1)
text(1.5, 14, "P=(0.55, 0.90)", cex=0.75, adj=0)
text(1.5, 13, "X: N(0, 1), m=95", cex=0.75, adj=0)
text(1.5, 12, "Y: N(1, 1), n=85", cex=0.75, adj=0)

plot(qchisq(1:1000/(1000+1),df=2), sort(chisp1exp00_45$chisp2), xlab
    ="Chisq(2) Quantiles", ylab="Sorted -2LLR Values")
abline(a=0, b=1)
text(1.5, 10, "P=(0, 0.45)", cex=0.75, adj=0)
text(1.5, 9, "X: exp(1), m=50", cex=0.75, adj=0)
text(1.5, 8, "Y: exp(0.25), n=40", cex=0.75, adj=0)

plot(qchisq(1:1000/(1000+1),df=2), sort(chisp_n60_100$chisp2), xlab=
    "Chisq(2) Quantiles", ylab="Sorted -2LLR Values", ylim=c(0, 20))
abline(a=0, b=1)
text(1.5, 11, "P=(0.55, 1)", cex=0.75, adj=0)
text(1.5, 10, "X: N(0, 1), m=100", cex=0.75, adj=0)
text(1.5, 9, "Y: N(1, 1), n=95", cex=0.75, adj=0)

###########chi_square_1.png####################

par(mfrow=c(2,2), mar=c(2.5, 2.5, 0, 0.5), mgp=c(0.95, 0.3, 0), cex.
    lab=0.75, cex.axis=0.65, cex.main=0.65, tck=0.01)

plot(qchisq(1:1000/(1000+1),df=1), sort(chisp1exp05_50$chisp), xlab=
    "Chisq(1) Quantiles", ylab="Sorted -2LLR Values")
abline(a=0, b=1)
text(1.5, 7.8, "P=(0.2, 0.5)", cex=0.75, adj=0)
text(1.5, 7.2, "X: exp(1), m=75", cex=0.75, adj=0)
text(1.5, 6.6, "Y: exp(0.25), n=65", cex=0.75, adj=0)

plot(qchisq(1:1000/(1000+1),df=1), sort(chisp_n55_95$chisp), xlab="
    Chisq(1) Quantiles", ylab="Sorted -2LLR Values")
abline(a=0, b=1)
text(1.5, 12, "P=(0.55, 0.90)", cex=0.75, adj=0)
text(1.5, 11, "X: N(0, 1), m=95", cex=0.75, adj=0)
text(1.5, 10, "Y: N(1, 1), n=85", cex=0.75, adj=0)
```

```
plot(qchisq(1:1000/(1000+1),df=1), sort(chisp1exp00_45$chisp), xlab=
   "Chisq(1) Quantiles", ylab="Sorted -2LLR Values")
abline(a=0, b=1)
text(1.5, 10, "P=(0, 0.45)", cex=0.75, adj=0)
text(1.5, 9, "X: exp(1), m=50", cex=0.75, adj=0)
text(1.5, 8, "Y: exp(0.25), n=40", cex=0.75, adj=0)

plot(qchisq(1:1000/(1000+1),df=1), sort(chisp_n60_100$chisp), xlab="
   Chisq(1) Quantiles", ylab="Sorted -2LLR Values")
abline(a=0, b=1)
text(1.5, 11, "P=(0.55, 1)", cex=0.75, adj=0)
text(1.5, 10, "X: N(0, 1), m=100", cex=0.75, adj=0)
text(1.5, 9, "Y: N(1, 1), n=95", cex=0.75, adj=0)
```

R code for Table 3.1

```
source('all_functions_pauc_eps.R')
###it was used before package 'pAUC' is developed###
ci.cover.norm.0p1 <- function(mux=0, stddx=1, muy=2,  stddy=2, nx
   =100, ny=100, p1=0, p2=0.7, n.repeat=1000, eps, level=3.84, ...){
  rp<-function(p, mux, stddx, muy, stddy)
  {
    y<-1-pnorm(qnorm(1-p, mux, stddx), muy, stddy)
    y
  }
  truepauc<-integrate(rp, mux=mux, stddx=stddx, muy=muy, stddy=stddy
     ,lower=p1,upper=p2)$value
  p <- ifelse(p1==0, p2, p1)
  results <- foreach(icount(n.repeat), .combine=cbind, .multicombine
     =TRUE, .init=numeric(2)) %dopar% {
    x <- rnorm(nx, mux, stddx)
    y <- rnorm(ny, muy, stddy)
    sortedx <- x[order(x)]
    cx<-cumsum(rep(1/nx, nx))
    highx <- sortedx[max(which(cx <= 1- p ))]
    if (p1==0) {
      est <-  rep(1/nx, nx)%*%mean.paucp2(x=x, lowx=highx, eps=eps,
         y=y)%*% rep(1/ny, ny)
    } else {
      est <-  rep(1/nx, nx)%*% mean.paucp1(x=x, highx=highx, eps=eps
         , y=y) %*% rep(1/ny, ny)
    }
    if (p1==0) {
      res.ci <- find.UL(step=0.01, fun=neighb.xy3p12, mean.fun=mean.
         paucp2, p=p, nx=nx, x=x, ny=ny, y=y, MLE=est, eps=eps,
         level=level)
    } else {
      res.ci <- find.UL(step=0.01, fun=neighb.xy3p12, mean.fun=mean.
         paucp1, p=p, nx=nx, x=x, ny=ny, y=y, MLE=est, eps=eps,
         level=level)
    }
    prob <-  (truepauc >= res.ci$Low & truepauc <= res.ci$Up)
    ci.length <- res.ci$Up - res.ci$Low
    c(prob, ci.length)
  }
```

```
  return(results)
}
###the above function will be as the following after package 'pAUC'
    is developed###
ci.cover.norm.0p1 <- function(mux=0, stddx=1, muy=2,  stddy=2, nx
    =100, ny=100, p1=0, p2=0.7, n.repeat=1000, eps, level=3.84, ...){
  rp<-function(p, mux, stddx, muy, stddy)
  {
    y<-1-pnorm(qnorm(1-p, mux, stddx), muy, stddy)
    y
  }
  truepauc<-integrate(rp, mux=mux, stddx=stddx, muy=muy, stddy=stddy
      ,lower=p1,upper=p2)$value
  results <- foreach(icount(n.repeat), .combine=cbind, .multicombine
      =TRUE, .init=numeric(2)) %dopar% {
  x <- rnorm(nx, mux, stddx)
  y <- rnorm(ny, muy, stddy)
  est <- est.pAUC (x, y, p=c(p1 , p2), eps=eps)
  res.ci <- find.UL(step =0.01, fun = neighb.xy3p12 , vp=c(p1, p2),
      MLE =est, x=x, y=y, eps =eps )
  prob <-  (truepauc >= res.ci$Low & truepauc <= res.ci$Up)
  ci.length <- res.ci$Up - res.ci$Low
  c(prob, ci.length)
  }
  return(results)
}
############### pAUC at [0, 0.1)
M <-20
library(doMC)
registerDoMC(M)
library(emplik)
RNGkind("L'Ecuyer-CMRG")
set.seed(123)
## start M workers
s <- .Random.seed
for (i in 1:M) {
  s <- nextRNGStream(s)
  # send s to worker i as .Random.seed
}
norm.ci0p1.100.100 <- ci.cover.norm.0p1( nx=100, ny=100, p1=0, p2
    =0.1, n.repeat=1000, eps=1/100)
save.image(norm.ci0p1.100.100, file='norm.ci0p1.100.100.RData',
    version = NULL, ascii = TRUE, compress=FALSE)
norm.ci0p1.50.50 <- ci.cover.norm.0p1(nx=50, ny=50, p1=0, p2=0.1, n.
    repeat=1000, eps=1/50)
save.image(norm.ci0p1.50.50, file='norm.ci0p1.50.50.RData', version
    = NULL, ascii = TRUE, compress=FALSE)
norm.ci0p1.30.30 <- ci.cover.norm.0p1( nx=30, ny=30, p1=0, p2=0.1, n
    .repeat=1000, eps=1/30)
save.image(norm.ci0p1.30.30, file='norm.ci0p1.30.30.RData', version
    = NULL, ascii = TRUE, compress=FALSE)
norm.ci0p1.50.30 <- ci.cover.norm.0p1(nx=50, ny=30, p1=0, p2=0.1, n.
    repeat=1000, eps=1/50)
```

```
save.image(norm.ci0p1.50.30, file='norm.ci0p1.50.30.RData', version
    = NULL, ascii = TRUE, compress=FALSE)
norm.ci0p1.80.50 <- ci.cover.norm.0p1(nx=80, ny=50, p1=0, p2=0.1, n.
    repeat=1000, eps=1/80)
save.image(norm.ci0p1.80.50, file='norm.ci0p1.80.50.RData', version
    = NULL, ascii = TRUE, compress=FALSE)

############### pAUC at [0, 0.7)
M <-20
library(doMC)
registerDoMC(M)
library(emplik)
RNGkind("L'Ecuyer-CMRG")
set.seed(123)
## start M workers
s <- .Random.seed
for (i in 1:M) {
  s <- nextRNGStream(s)
  # send s to worker i as .Random.seed
}
norm.ci0p7.100.100 <- ci.cover.norm.0p1( nx=100, ny=100, p1=0, p2
    =0.7, n.repeat=1000, eps=1/100)
save.image(norm.ci0p7.100.100, file='norm.ci0p7.100.100.RData',
    version = NULL, ascii = TRUE, compress=FALSE)
norm.ci0p7.50.50 <- ci.cover.norm.0p1(nx=50, ny=50, p1=0, p2=0.7, n.
    repeat=1000, eps=1/50)
save.image(norm.ci0p7.50.50, file='norm.ci0p7.50.50.RData', version
    = NULL, ascii = TRUE, compress=FALSE)
norm.ci0p7.30.30 <- ci.cover.norm.0p1( nx=30, ny=30, p1=0, p2=0.7, n
    .repeat=1000, eps=1/30)
save.image(norm.ci0p7.30.30, file='norm.ci0p7.30.30.RData', version
    = NULL, ascii = TRUE, compress=FALSE)
norm.ci0p7.50.30 <- ci.cover.norm.0p1(nx=50, ny=30, p1=0, p2=0.7, n.
    repeat=1000, eps=1/50)
save.image(norm.ci0p7.50.30, file='norm.ci0p7.50.30.RData', version
    = NULL, ascii = TRUE, compress=FALSE)
norm.ci0p7.80.50 <- ci.cover.norm.0p1(nx=80, ny=50, p1=0, p2=0.7, n.
    repeat=1000, eps=1/80)
save.image(norm.ci0p7.80.50, file='norm.ci0p7.80.50.RData', version
    = NULL, ascii = TRUE, compress=FALSE)

############### pAUC at (0.05, 0.5)

M <-20
library(doMC)
registerDoMC(M)
library(emplik)
RNGkind("L'Ecuyer-CMRG")
set.seed(123)
## start M workers
s <- .Random.seed
for (i in 1:M) {
  s <- nextRNGStream(s)
  # send s to worker i as .Random.seed
```

```
}
###it was used before package 'pAUC' is developed###
ci.cover.norm.p12 <- function(mux=0, stddx=1, muy=2,  stddy=2, nx
   =100, ny=100, p1=0.05, p2=0.5, n.repeat=1000, eps,...){
  rp<-function(p, mux, stddx, muy, stddy)
  {
    y<-1-pnorm(qnorm(1-p, mux, stddx), muy, stddy)
    y
  }
  truepauc<-integrate(rp, mux=mux, stddx=stddx, muy=muy, stddy=stddy
     ,lower=p1,upper=p2)$value
  results <- foreach(icount(n.repeat), .combine=cbind, .multicombine
     =TRUE, .init=numeric(2)) %dopar% {
    x <- rnorm(nx, mux, stddx)
    y <- rnorm(ny, muy, stddy)
    sortedx <- x[order(x)]
    cx<-cumsum(rep(1/nx, nx))
    lowx <- sortedx[max(which(cx <= 1- p2))]
    highx <- sortedx[max(which(cx <= 1- p1 ))]
    est <- rep(1/nx, nx)%*% mean.pauc(x=x, lowx=lowx, highx=highx,
       eps=eps, y=y)%*% rep(1/ny, ny)
    res.ci <- find.UL(step=0.01, fun=neighb.xy, sp12=c(p1, p2), MLE=
       est, nx=nx, x=x, ny=ny, y=y, eps=eps)
    prob <-  (truepauc >= res.ci$Low & truepauc <= res.ci$Up)
    ci.length <- res.ci$Up - res.ci$Low
    c(prob, ci.length)
  }
  return(results)
}
###the above function will be as the following after package 'pAUC'
   is developed###
ci.cover.norm.p12 <- function(mux=0, stddx=1, muy=2,  stddy=2, nx
   =100, ny=100, p1=0.05, p2=0.5, n.repeat=1000, eps,...){
  rp<-function(p, mux, stddx, muy, stddy)
  {
    y<-1-pnorm(qnorm(1-p, mux, stddx), muy, stddy)
    y
  }
  truepauc<-integrate(rp, mux=mux, stddx=stddx, muy=muy, stddy=stddy
     ,lower=p1,upper=p2)$value
  results <- foreach(icount(n.repeat), .combine=cbind, .multicombine
     =TRUE, .init=numeric(2)) %dopar% {
    x <- rnorm(nx, mux, stddx)
    y <- rnorm(ny, muy, stddy)
    est <- est.pAUC(x, y, p=c(p1 , p2), eps=eps)
    res.ci <- find.UL( step =0.01, fun = neighb.xy, sp12 =c(p1, p2),
        x=x, y=y, MLE =est, eps=eps )
    prob <-  (truepauc >= res.ci$Low & truepauc <= res.ci$Up)
    ci.length <- res.ci$Up - res.ci$Low
    c(prob, ci.length)
  }
  return(results)
}
```

```
norm.ci.100.100 <- ci.cover.norm.p12( nx=100, ny=100, p1=0.05, p2
   =0.5, n.repeat=1000, eps=1/100)
save.image(norm.ci.100.100, file='norm.ci.100.100.RData', version =
   NULL, ascii = TRUE, compress=FALSE)
norm.ci.50.50 <- ci.cover.norm.p12(nx=50, ny=50, p1=0.05, p2=0.5, n.
   repeat=1000, eps=1/50)
save.image(norm.ci.50.50, file='norm.ci.50.50.RData', version = NULL
   , ascii = TRUE, compress=FALSE)
norm.ci.30.30 <- ci.cover.norm.p12( nx=30, ny=30, p1=0.05, p2=0.5, n
   .repeat=1000, eps=1/30)
save.image(norm.ci.30.30, file='norm.ci.30.30.RData', version = NULL
   , ascii = TRUE, compress=FALSE)
ptm <- proc.time()
norm.ci.50.30 <- ci.cover.norm.p12(nx=50, ny=30, p1=0.05, p2=0.5, n.
   repeat=1000, eps=1/50)
proc.time() - ptm
save.image(norm.ci.50.30, file='norm.ci.50.30.RData', version = NULL
   , ascii = TRUE, compress=FALSE)
norm.ci.80.50 <- ci.cover.norm.p12(nx=80, ny=50, p1=0.05, p2=0.5, n.
   repeat=1000, eps=1/80)
save.image(norm.ci.80.50, file='norm.ci.80.50.RData', version = NULL
   , ascii = TRUE, compress=FALSE)
```

R code for Table 3.2

```
###it was used before package 'pAUC' is developed###
ci.cover.exp.0p1 <- function(rate.x=1, rate.y=0.25, nx=100, ny=100,
   p1=0.05, p2=0.5, n.repeat=1000, eps, level=3.84, ...){
  rp.exp2<-function(p, rate.x, rate.y)
  {
    y<- 1- pexp(qexp(1-p, rate=rate.x), rate=rate.y)
    y
  }
  truepauc <- integrate(rp.exp2, rate.x, rate.y,lower=p1, upper=p2)$
     value
  p <- ifelse(p1==0, p2, p1)
  results <- foreach(icount(n.repeat), .combine=cbind, .multicombine
     =TRUE, .init=numeric(2)) %dopar% {
    x <- rexp(nx, rate.x)
    y <- rexp(ny, rate.y)
    sortedx <- x[order(x)]
    cx<-cumsum(rep(1/nx, nx))
    highx <- sortedx[max(which(cx <= 1- p ))]
    if (p1==0) {
      est <-  rep(1/nx, nx)%*%mean.paucp2(x=x, lowx=highx, eps=eps,
         y=y)%*% rep(1/ny, ny)
    } else {
      est <-  rep(1/nx, nx)%*% mean.paucp1(x=x, highx=highx, eps=eps
         , y=y) %*% rep(1/ny, ny)
    }
    if (p1==0) {
      res.ci <- find.UL(step=0.01, fun=neighb.xy3p12, mean.fun=mean.
         paucp2, p=p, nx=nx, x=x, ny=ny, y=y, MLE=est, eps=eps,
         level=level)
    } else {
```

```
      res.ci <- find.UL(step=0.01, fun=neighb.xy3p12, mean.fun=mean.
          paucp1, p=p, nx=nx, x=x, ny=ny, y=y, MLE=est, eps=eps,
          level=level)
    }
    prob <- (truepauc >= res.ci$Low & truepauc <= res.ci$Up)
    ci.length <- res.ci$Up - res.ci$Low
    c(prob, ci.length)
  }
  return(results)
}
###the above function will be as the following after package 'pAUC'
    is developed###
ci.cover.exp.0p1 <- function(rate.x=1, rate.y=0.25, nx=100, ny=100,
    p1=0.05, p2=0.5, n.repeat=1000, eps, level=3.84, ...){
  rp.exp2<-function(p, rate.x, rate.y)
  {
    y<- 1- pexp(qexp(1-p, rate=rate.x), rate=rate.y)
    y
  }
  truepauc <- integrate(rp.exp2, rate.x, rate.y,lower=p1, upper=p2)$
      value
  results <- foreach(icount(n.repeat), .combine=cbind, .multicombine
      =TRUE, .init=numeric(2)) %dopar% {
    x <- rexp(nx, rate.x)
    y <- rexp(ny, rate.y)
    est <- est.pAUC (x, y, p=c(p1 , p2), eps=eps)
    res.ci <- find.UL(step =0.01, fun = neighb.xy3p12, vp=c(p1, p2),
        MLE =est, x=x, y=y, eps =eps )
    prob <- (truepauc >= res.ci$Low & truepauc <= res.ci$Up)
    ci.length <- res.ci$Up - res.ci$Low
    c(prob, ci.length)
  }
  return(results)
}
############### pAUC at [0, 0.1)

M <-20
library(doMC)
registerDoMC(M)
library(emplik)
RNGkind("L'Ecuyer-CMRG")
set.seed(123)
## start M workers
s <- .Random.seed
for (i in 1:M) {
  s <- nextRNGStream(s)
  # send s to worker i as .Random.seed
}
exp.ci0p1.100.100 <- ci.cover.exp.0p1(rate.x=1, rate.y=0.25, nx=100,
    ny=100, p1=0, p2=0.1, n.repeat=1000, eps=1/100)
save.image(exp.ci0p1.100.100, file='exp.ci0p1.100.100.RData',
    version = NULL, ascii = TRUE, compress=FALSE)
exp.ci0p1.50.50 <- ci.cover.exp.0p1(rate.x=1, rate.y=0.25, nx=50, ny
    =50, p1=0, p2=0.1, n.repeat=1000, eps=1/50)
```

```
save.image(exp.ci0p1.50.50, file='exp.ci0p1.50.50.RData', version =
    NULL, ascii = TRUE, compress=FALSE)
exp.ci0p1.30.30 <- ci.cover.exp.0p1(rate.x=1, rate.y=0.25, nx=30, ny
    =30, p1=0, p2=0.1, n.repeat=1000, eps=1/30)
save.image(exp.ci0p1.30.30, file='exp.ci0p1.30.30.RData', version =
    NULL, ascii = TRUE, compress=FALSE)
exp.ci0p1.50.30 <- ci.cover.exp.0p1(rate.x=1, rate.y=0.25, nx=50, ny
    =30, p1=0, p2=0.1, n.repeat=1000, eps=1/50)
save.image(exp.ci0p1.50.30, file='exp.ci0p1.50.30.RData', version =
    NULL, ascii = TRUE, compress=FALSE)
exp.ci0p1.80.50 <- ci.cover.exp.0p1(rate.x=1, rate.y=0.25, nx=80, ny
    =50, p1=0, p2=0.1, n.repeat=1000, eps=1/80)
save.image(exp.ci0p1.80.50, file='exp.ci0p1.80.50.RData', version =
    NULL, ascii = TRUE, compress=FALSE)

############### pAUC at [0, 0.7)

M <-20
library(doMC)
registerDoMC(M)
library(emplik)
RNGkind("L'Ecuyer-CMRG")
set.seed(123)
## start M workers
s <- .Random.seed
for (i in 1:M) {
  s <- nextRNGStream(s)
  # send s to worker i as .Random.seed
}
exp.ci0p7.100.100 <- ci.cover.exp.0p1(rate.x=1, rate.y=0.25, nx=100,
     ny=100, p1=0, p2=0.7, n.repeat=1000, eps=1/100)
save.image(exp.ci0p7.100.100, file='exp.ci0p7.100.100.RData',
    version = NULL, ascii = TRUE, compress=FALSE)
exp.ci0p7.50.50 <- ci.cover.exp.0p1(rate.x=1, rate.y=0.25, nx=50, ny
    =50, p1=0, p2=0.7, n.repeat=1000, eps=1/50)
save.image(exp.ci0p7.50.50, file='exp.ci0p7.50.50.RData', version =
    NULL, ascii = TRUE, compress=FALSE)
exp.ci0p7.30.30 <- ci.cover.exp.0p1(rate.x=1, rate.y=0.25, nx=30, ny
    =30, p1=0, p2=0.7, n.repeat=1000, eps=1/30)
save.image(exp.ci0p7.30.30, file='exp.ci0p7.30.30.RData', version =
    NULL, ascii = TRUE, compress=FALSE)
exp.ci0p7.50.30 <- ci.cover.exp.0p1(rate.x=1, rate.y=0.25, nx=50, ny
    =30, p1=0, p2=0.7, n.repeat=1000, eps=1/50)
save.image(exp.ci0p7.50.30, file='exp.ci0p7.50.30.RData', version =
    NULL, ascii = TRUE, compress=FALSE)
exp.ci0p7.80.50 <- ci.cover.exp.0p1(rate.x=1, rate.y=0.25, nx=80, ny
    =50, p1=0, p2=0.7, n.repeat=1000, eps=1/80)
save.image(exp.ci0p7.80.50, file='exp.ci0p7.80.50.RData', version =
    NULL, ascii = TRUE, compress=FALSE)

############### pAUC at (0.05, 0.5)
###it was used before package 'pAUC' is developed###
ci.cover.exp.p12 <- function(rate.x=1, rate.y=0.25, nx=100, ny=100,
    p1=0.05, p2=0.5, n.repeat=1000, eps,...){
```

```
  rp.exp2<-function(p, rate.x, rate.y)
  {
    y<- 1- pexp(qexp(1-p, rate=rate.x), rate=rate.y)
    y
  }
  truepauc <- integrate(rp.exp2, rate.x, rate.y,lower=p1, upper=p2)$
     value
  results <- foreach(icount(n.repeat), .combine=cbind, .multicombine
     =TRUE, .init=numeric(2)) %dopar% {
    x <- rexp(nx, rate.x)
    y <- rexp(ny, rate.y)
    sortedx <- x[order(x)]
    cx<-cumsum(rep(1/nx, nx))
    lowx <- sortedx[max(which(cx <= 1- p2))]
    highx <- sortedx[max(which(cx <= 1- p1 ))]
    est <- rep(1/nx, nx)%*% mean.pauc(x=x, lowx=lowx, highx=highx,
       eps=eps, y=y)%*% rep(1/ny, ny)
    res.ci <- find.UL(step=0.01, fun=neighb.xy, sp12=c(p1, p2), MLE=
       est, nx=nx, x=x, ny=ny, y=y, eps=eps)
    prob <- (truepauc >= res.ci$Low & truepauc <= res.ci$Up)
    ci.length <- res.ci$Up - res.ci$Low
    c(prob, ci.length)
  }
  return(results)
}
###the above function will be as the following after package 'pAUC'
   is developed###
ci.cover.exp.p12 <- function(rate.x=1, rate.y=0.25, nx=100, ny=100,
   p1=0.05, p2=0.5, n.repeat=1000, eps,...){
  rp.exp2<-function(p, rate.x, rate.y)
  {
    y<- 1- pexp(qexp(1-p, rate=rate.x), rate=rate.y)
    y
  }
  truepauc <- integrate(rp.exp2, rate.x, rate.y,lower=p1, upper=p2)$
     value
  results <- foreach(icount(n.repeat), .combine=cbind, .multicombine
     =TRUE, .init=numeric(2)) %dopar% {
    x <- rexp(nx, rate.x)
    y <- rexp(ny, rate.y)
    est <- est.pAUC(x, y, p=c(p1 , p2), eps=eps)
    res.ci <- find.UL( step =0.01, fun = neighb.xy, sp12 =c(p1, p2),
        x=x, y=y, MLE =est, eps=eps )
    prob <- (truepauc >= res.ci$Low & truepauc <= res.ci$Up)
    ci.length <- res.ci$Up - res.ci$Low
    c(prob, ci.length)
  }
  return(results)
}
M <-20
library(doMC)
registerDoMC(M)
library(emplik)
RNGkind("L'Ecuyer-CMRG")
```

```
set.seed(123)
## start M workers
s <- .Random.seed
for (i in 1:M) {
  s <- nextRNGStream(s)
  # send s to worker i as .Random.seed
}
exp.ci.100.100 <- ci.cover.exp.p12(rate.x=1, rate.y=0.25, nx=100, ny
   =100, p1=0.05, p2=0.5, n.repeat=1000, eps=1/100)
save.image(exp.ci.100.100, file='exp.ci.100.100.RData', version =
   NULL, ascii = TRUE, compress=FALSE)
exp.ci.50.50 <- ci.cover.exp.p12(rate.x=1, rate.y=0.25, nx=50, ny
   =50, p1=0.05, p2=0.5, n.repeat=1000, eps=1/50)
save.image(exp.ci.50.50, file='exp.ci.50.50.RData', version = NULL,
   ascii = TRUE, compress=FALSE)
exp.ci.30.30 <- ci.cover.exp.p12(rate.x=1, rate.y=0.25, nx=30, ny
   =30, p1=0.05, p2=0.5, n.repeat=1000, eps=1/30)
save.image(exp.ci.30.30, file='exp.ci.30.30.RData', version = NULL,
   ascii = TRUE, compress=FALSE)
exp.ci.50.30 <- ci.cover.exp.p12(rate.x=1, rate.y=0.25, nx=50, ny
   =30, p1=0.05, p2=0.5, n.repeat=1000, eps=1/50)
save.image(exp.ci.50.30, file='exp.ci.50.30.RData', version = NULL,
   ascii = TRUE, compress=FALSE)
exp.ci.80.50 <- ci.cover.exp.p12(rate.x=1, rate.y=0.25, nx=80, ny
   =50, p1=0.05, p2=0.5, n.repeat=1000, eps=1/80)
save.image(exp.ci.80.50, file='exp.ci.80.50.RData', version = NULL,
   ascii = TRUE, compress=FALSE)
```

R codes for Simulations in Chapter 4

R code for Figure 4.1

```
myfun5 <- function(x, theta, eps) {
  u <- (x-theta)*sqrt(5)/eps
  INDE <- (u < sqrt(5)) & (u > -sqrt(5))
  u[u >= sqrt(5)] <- 0
  u[u <= -sqrt(5)] <- 1
  y <- 0.5 - (u - (u)^3/15)*3/(4*sqrt(5))
  u[ INDE ] <- y[ INDE ]
  return(u)
}
theta <- 0;
eps <- 0.6
x <- seq ( -1.8 , 1.8 , 0.1)
plot (x, myfun5 (x, theta , eps ), type ='l', xlim =c(-2, 2) , ylim
   =c( -0.5, 1.5) , xlab ='x', ylab = expression ( paste ( bold (I),
    epsilon , "(x, x*)")))
lines (x=c( -0.6 , -0.6) , y=c( -0.3 , 1.3) , lty =2)
lines (x=c(0.6 , 0.6) , y=c( -0.3 , 1.3) , lty =2)
text (x= -0.6 , y= -0.45 , expression ( paste ('x = x', '*', ' - ',
   epsilon )), cex =0.8)
```

```
text (x=0.6 , y= -0.45 , expression ( paste ('x = x', '*', ' + ',
    epsilon )), cex =0.8)
text (x=1, y=1, expression ( paste ('x*'==0) ), cex =0.8)
text (x=1.02 , y=0.8 , expression ( paste ( epsilon == 0.6) ), cex
    =0.8)
```

R code for Figure 4.2

```
**********************************************************************
The following R codes call the internal functions 'eltest2u' and
'eltest2v' of package 'pAUC'
**********************************************************************
set.seed(2)
omu <- 0
p1 <- 0.2
p2 <- 0.6
muy<-1
stddy<-1
rp<-function(p,muy,stddy)
{
  y<-1-pnorm(qnorm(1-p),muy,stddy)
  y
}
truepauc<-integrate(rp, muy=muy,stddy=stddy,lower=p1,upper=p2)$value
nx <- 50
ny <- 45
x <- rnorm(nx, omu, 1)
y <- rnorm(ny, muy, stddy)
sortedx <- x[order(x)]
cx<-cumsum(rep(1/nx, nx))
sp12 = c(p1, p2)
idex1 <- max(which(cx <=1- sp12[2] ))
idex2 <- max(which(cx <=1- sp12[1] ))
ab <- sortedx[c(idex1, idex2)]
est.pauc <- rep(1/nx, nx)%*% mean.pauc(x, ab[1], ab[2], eps=1/nx, y)
    %*% rep(1/ny, ny)
search.low <- ifelse((min(est.pauc, truepauc)-0.1) >0, (min(est.pauc
    , truepauc)-0.1), 0)
search.high <- ifelse((max(est.pauc, truepauc)+0.1) <1 , (max(est.
    pauc, truepauc)+0.1), 1)
temp.pauc <- seq(search.low,search.high, length.out=30)
res.plot <- numeric(30)
for (i in 1:30){
  res1p <- eltest2u(ab, x.sample=x, mu=c(1-p2, 1-p1, temp.pauc[i]),
      y, vvec=rep(1/ny, ny), eps=1/nx)
  res2p <- eltest2v(ab, x.sample=x, mu=truepauc, y, res1p$uvec/nx,
      eps=1/nx)
  res.plot[i] <-  res1p$"-2LLR" + res2p$"-2LLR"
}
par(mfrow = c(1, 1), mar = c(3, 3.5, 1, 0.5), mgp = c(2, 0.7, 0),
    cex.lab = 0.95, cex.axis = 0.85, cex.main=0.65)
plot(temp.pauc, res.plot, ylim=c(0, 100), ylab="sum of -2LLR from
    steps 1 and 2")
points(est.pauc, 0, pch=6)
points(truepauc, 0, pch=17)
```

```
legend(temp.pauc[20], 25, pch=c(6, 17), legend=c('est. pAUC', 'true
    pAUC'), bty='n', cex=0.85)
```

R code for Figure 4.3

```
set.seed(5)
omu <- 0
p1 <- 0.2
p2 <- 0.8
mu <- c(p1, p2, omu)
nx <- 200
x <- rnorm(nx, omu, 1)
p <- 0.15
#p is a number chosen to search a among samples in (p1-p, p1+p),
    search b among samples in (p2-p, p2+p)
sortedx <- x[order(x)]
cx<-cumsum(rep(1/nx, nx))
ax <- sortedx[which((cx >= p1 - p) & (cx <= p1 + p))]
bx <- sortedx[which((cx >= p2 - p) & (cx <= p2 + p))]
nax <- length(ax)
nbx <- length(bx)
results <- matrix(NA, nrow=nax, ncol=nbx)
for (i in 1:nax){
  for (j in 1:nbx){
    axb <- matrix(c(as.vector(x < ax[i] ), as.vector(x < bx[j]), as.
       vector(x*(x >= ax[i] & x <= bx[j]))), ncol=3)
    all <- el.test(axb, mu)
    results[i, j] <-  all$`-2LLR`
  }
}
persp(ax, bx, (-1)*results, theta=30, phi=30, ticktype="detailed")
ab <- which(results==min(results), arr.ind = TRUE)
res <- persp(ax, bx, (-1)*results, theta=30, phi=30, ticktype="
    detailed")
points(trans3d(ax[ab[1]], bx[ab[2]], (-1)*results[ab[1], ab[2]],
    pmat=res), col='red', pch=16)
```

Bibliography

Barton, W. H. (2010). Comparison of two samples by a nonparametric likelihood ratio test. *University of Kentucky Doctoral Dissertations. Paper 99.*

Beach, C. M. and Davidson, R. (1983). Distribution-free statistical inference with lorenz curves and income shares. *The Review of Economic Studies*, 50:723–735.

Bickel, P. J. (1965). On some robust estimates of location. *The Annals of Mathematical Statistics*, 36:847–858.

Bishop, J., Chakraborti, S., and Paul D. Thistle, title = Asymptotically Distribution-Free Statistical Inference for Generalized Lorenz Curves, j. . T. Y. . . V. . . P. . .

Caperra, P. and Rivest, L. (1995). On the variance of the trimmed mean. *Mathematical Proceedings of the Cambridge Philosophical Society*, 22:79–85.

Casella, G. and Berger, R. L. (2002). *Statistical Inference*. Duxbury Press, Pacific Grove, California.

Chakraborti, S. (1994). Asymptotically distribution-free statistical inference for generalized lorenz curves based on complete data. *Statistics Probability Letters*, 21:229–235.

Chen, S. X. and Hall, P. (1993). Smoothed empirical likelihood confidence intervals for quantiles. *The Annals of Statistics*, 21:1166–1181.

Davison, A. C. (2003). *Statistical Models*. Cambridge University Press.

Dixon, W. J. and Tukey, J. W. (1968). Approximate behavior of the distribution of winsorized t (trimming/winsorization 2). *Technometrics*, 10:83–98.

Dodd, L. and Pepe, M. (2003). Partial auc estimation and regression. *Biometrics*, 59:614–623.

Gastwirth, J. L. (1972). The estimation of the lorenz curve and gini index. *The Review of Economics and Statistics*, 54:306–316.

Hajian-Tilaki, K. (2013). Roc curve analysis for medical diagnostic test evaluation. *Caspian Journal of Internal Medicine*, 4(2):627–635.

Hampel, F. R. (1985). The breakdown points of the mean combined with some rejection rules. *Technometrics*, 27:95–107.

Hanley, J. A. and McNeil, B. J. (1982). The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143:29–36.

Jihnhee Yu, Luge Yang, A. V. A. D. H. (2011). A generalized empirical likelihood approach for two-group comparisons given a u-statistic constraint.

Lorenz, M. O. (1905). Methods of measuring the concentration of wealth. *Publications of the American Statistical Association*, 9:209–219.

McClish, D. K. (1989). Analyzing a portion of the roc curve. *RadiologyMedical Decision Making*, 9(3):190–195.

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

Owen, A. B. (2001). *Empirical Likelihood*. Chapman Hall/CRC, Boca Raton, London, New York, Washington, D.C.

Qin, G., Jin, X., and Zhou, X. (2011). Non-parametric interval estimation for the partial area under the roc curve. *The Canadian Journal of Statistics*, 39(1):17–33.

Qin, G. and Tsao, M. (2002). Empirical likilihood ratio confidnece inteval for the trimmed mean. *Communications in Statistics, Theory and Methods*, 31:2197–2202.

Qin, G., Yang, B., and Belinga-Hall, N. E. (2013). Empirical likelihood-based inferences for the lorenz curve. *Annals of the Institute of Statistical Mathematics*, 65:1–21.

Qin, J. and Lawless, J. (1994). Empirical likelihood and general estimating equations. *The Annals of Statistics*, 22:300–325.

SAS Institute Inc. (2010). *Base SAS® 9.2 Procedures Guide: Statistical Procedures, Third Edition*. Cary, NC.

Sen, P. K. (1967). A note on asymptotically distribution-free confidnece bounds for $p\{X < Y\}$, based on two independent samples. *Sankhya: The Indian Journal of Statistics*, 29:95–102.

Stigler, S. M. (1973). The asymptotic distribution of the trimmed mean. *The Annals of Statistics*, 1:472–477.

Thomas, D. R. and Grunkemeier, G. L. (1975). Confidence interval estimation of survival probabilities for censored data. *Journal of the American Statistical Association*, 70:865–871.

Tukey, J. W. and McLaughlin, D. H. (1963). Less vulnerable confidence and significance procedures for location based on a single sample: Trimming/winsorization 1. *Sankhya: The Indian Journal of Statistics, Series A*, 25:331–352.

Wilks, S. S. (1938). The large-sample distribution of the likelihood ratio for testing composite hypotheses. *The Annals of Mathematical Statistics*, 9:60–62.

Y. Jiang, CE. Metz, R. N. (1996). A receiver operating characteristic partial area index for highly sensitive diagnostic tests. *Radiology*, 201(3):745–750.

Yang, H., Lu, K., and Zhao, Y. (2016). A nonparametric approach for partial areas under roc curves and ordinal dominance curves. *Statistica Sinica*.

Zhou, M. (2016). *Empirical Likelihood Method in Survival Analysis*. CRC Press, Boca Raton.

Zhou, M. and Yang, Y. (2014). *emplik: Empirical likelihood ratio for censored/truncated data*. R package version 0.9-9-6.

Zweig, M. H. and Campbell, G. (1993). Receiver-operating characteristic (roc) plots: A fundamental evaluation tool in clinical medicine. *Clinical Chemistry*, 39(4):561–577.

Vita

- Place of birth: Liaoning, China

- Educational institutions attended and degrees already awarded

  PhD student in Statistics: University of Kentucky
  MS in Statistics: University of Kentucky
  PhD in Materials Science and Engineering: University of Kentucky
  MS in Mineral Processing: Northeast University, China
  BS in Mineral Processing: Northeast University, China

- Professional publications in University of Kentucky:

  Yumin Zhao and Mai Zhou, Statistical Inference on Partial Area Under ROC Curves by Empirical Likelihood Method, Statistics in Medicine, submitted
  Yumin Zhao and Mai Zhou, R package 'pAUC', http://cran.r-project.org, to be submitted