Estimation Following Self-designing Clinical Trial

Shengli Shi

A thesis submitted in partial fulfillment of the
requirements for the degree of

Master of Science

University of Washington

2003

Program authorized to Offer Degree: Public Health and Community Medicine-
Biostatistics

University of Washington
Graduate School

This is to certify that I have examined this copy of a master's thesis by

Shengli Shi

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____
Scott S Emerson

_____
Thomas Lumley

Date: _____

In presenting this thesis in partial fulfillment of the requirements for a Master's degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this thesis is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Any other reproduction for any purposes or by any means shall not be allowed without my written permission.

Signature_____

Date_____

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

The author wishes to express sincere appreciation to Department of Biostatistics, where he has had the opportunity to study statistical theories and applications, and to my advisor, Dr. Scott S Emerson.

Chapter 1

INTRODUCTION

Clinical trials that evaluate the efficacy of new treatment methods are necessary to ensure scientific credibility and to protect the safety of the research subjects. Interim analysis of the data becomes necessary for various reasons: to choose the best treatment for the subjects, to avoid toxic effects revealed by the trial data, and to use as small as possible a sample size while ensuring scientific credibility. In order to address these goals, interim analyses are used to estimate and/or test the magnitude of treatment effect.

Methods have also been described for using the interim results to adjust the planned sample size in order to deal with imprecise estimates of trial design parameters. It is not uncommon that the investigators underestimated the standard error or event rate, in which case the original sample size might not be large enough to detect a meaningful difference [13]. Similarly, the actual schedule of interim analyses might differ from the plan, thus altering statistical power [2]. Approaches to clinical trial design which address these issues include group sequential trials (GST) and, more recently, the self-designing clinical trial (SDCT).

Group sequential clinical trial (GST) designs have been developed to perform interim analysis of the data and reach the scientific and ethical goals [1]. Different kinds of group sequential clinical trial designs have been studied and widely used [1, 2, 3, 4, 9]. Most of these designs can be viewed as stopping rules defined for the estimate of treatment effect. Although different designs use different group sequential stopping rules, the purpose of these methods all focus on deciding whether there is already sufficient evidence of either beneficial or inferior effect of the new treatment, or whether the trial needs to be continued.

The classical GST designs perform tests to decide whether to stop or continue the study

at each stage. The aim is to terminate the study early due to overwhelming evidence from the interim findings. Prior to the first analysis, a stopping rule is specified which provides the thresholds corresponding to early stopping in such a way as to maintain the overall type I error. In most cases, either the maximal sample size or the power to detect a particular alternative is fixed at the design stage.

Fisher's self-designing clinical trial (SDCT)[13], on the other hand, is a relatively new flexible sequential inference procedure in which neither the maximal sample size nor the power need be specified in advance. In adapting the sample size according to the strength of effect observed at an interim analysis, the SDCT addresses issues related to minimizing the sample size necessary to answer the scientific question in a credible fashion. However, it also aims to solve the problem of incorrect estimation of original sample size for prospective clinical trials, and even allows for changes in the definition of the clinically important difference that the trial should detect [13].

There exist significant distinctions between the self-designing clinical trial method and classical GST methods. Classical methods test the treatment effect and can allow early termination at each interim analysis, but the rule must be specified in advance. The self-designing method adaptively chooses the maximal sample size, but only tests for treatment effect after the method terminates the trial- when the variance is used up. When using this approach, Fisher proposed using a test statistic, which has a standard normal distribution $T \sim N(0,1)$ under the null hypothesis regardless of the way in which interim results were used to determine sampling plan. Although there seems to be some benefits for the self-designing clinical trial method, there are also possible shortcomings for this design, and the costs could be more than the benefits.

The SDCT can be criticized on several grounds.

1) Is it really necessary? Careful evaluation of a clinical trial can handle many of the situations used to motivate use of SDCT [18].

2) Can the design of an SDCT be inherently more powerful than a GST? Tsiatis and Metha argue that we can always find a GST that is as efficient [16].

3) When using the SDCT, is the T statistic advocated by Fisher the best one to use? That is, there would seem to be inefficiency due to the fact that the T statistic is not solely a function of the sufficient statistic.

The third issue is the question we address here by looking at inference based on T and inference based on the maximum likelihood estimate (MLE) of the treatment effect. In the setting of a prespecified self-designing clinical trial sampling plan, we compare the efficiency and bias of these statistics, as well as the statistical power when using them.

The SDCT is introduced in section 2. The methods that we used to compare these criteria are introduced in section 3. The results of numerical integration studies are discussed in section 4. The comparison of the numbers of stages of analysis, sample sizes, and power in group sequential methods and SDCT are provided in section 5.

Chapter 2

NOTATION FOR MONITORING AND CONTRAST GST & SDCT

Suppose $\vec{X}$ is a vector of doubly subscripted observations with X_{ij} the jth observation $(j = 1, 2, ..., n_i)$ in stage i. Further assume $X_{ij} \sim N(\mu, \sigma^2)$. For notational convenience, define $\overline{X}_i = \frac{1}{n_i} \sum_{j=1}^{n_i} X_{ij}$ which, when n_i is known, has distribution

$$\overline{X}_i | n_i \sim N\left(\mu, \frac{\sigma^2}{n_i}\right).$$

In order to test $H_0 : \mu = \mu_0$, we also define

$$Z_i = \sqrt{n_i}\left(\frac{\overline{X}_i - \mu_0}{\sigma}\right)$$

which has conditional null distribution $Z_i | n_i \sim N(0, 1)$.

The SDCT can be motivated by considering test statistic T, a weighted average of the observations defined as

$$T(\vec{X}) = \sum_{1}^{m} a_i Z_i = \sum_{1}^{m} a_i \sqrt{n_i} \overline{X}_i,$$

where a_i, n_i, and $\overline{X}_i$ represent the weight, sample size and group mean for stage i, respectively, and m is the maximum stage number. Note that $\vec{a} = (a_1, ..., a_m), \vec{n} = (n_1, n_2, ..., n_m), \vec{Y} = (\overline{X}_1, \overline{X}_2, .., \overline{X}_m)$ is sufficient for calculating T and estimating μ. The weights a_i and sample sizes n_i are chosen sequentially such that

$$\sum_{i=1}^{m} a_i^2 = 1$$

for some random integer $m \geq 1$. The weight assigned to the i-th stage must be determined in advance of performing the i-th analysis, but need not be decided upon except at the immediately preceding analysis. Thus at the beginning of the trial, one needs to choose the sample size for the first analysis and the weight to be assigned to the first stage. Then, at each analysis, the weight to be assigned to the next stage is determined. The statistic is

computed in a way which allows any information available at the interim analysis to be used to determine the next stage's weight while protecting the experiment wise type I error. The study can and must terminate at the next stage if all remaining weight is assigned to that stage. The weighted average of each block of data comprises the final statistic T, which is used for testing the hypotheses.

Notationally, the adaptive selection of the weights and sample size proceeds as follows. Basically, after each stage i, the weight function and sample size of stage $i+1$ is determined based on the prior history of the trial. That is n_{i+1} can be a function of $\{X_{kj} : k = 1, ..., i; j = 1, ..., n_k\}$. Specifically, in the SDCT, at start of the study, we identify a sample size n_1 for stage 1 and a weight $0 < a_1 \leq 1$. This weight represents the influence of the first stage on the final statistic. The trial proceeds by observing $X_{11}, ..., X_{1n_1}$. Then a sample size for the next stage and a weight $0 < a_2 \leq \sqrt{1 - a_1^2}$ is computed, perhaps depending upon the data observed to date. The trial proceeds in this manner. After observing the i-th stage data, we choose n_{i+1} as some function of $\{X_{kj} : 1 \leq k \leq i, 1 \leq j \leq n_k\}$ and weight a_{i+1} subject to $0 \leq a_{i+1} \leq \sqrt{1 - \sum_{k=1}^{i} a_k^2}$. The final (or m-th stage) occurs when we choose $a_m = \sqrt{1 - \sum_{k=1}^{m-1} a_k^2}$.

Following the derivation of Fisher [13], under the null hypothesis $\mu = 0$, T has a standard normal distribution with

$$Var(T) = \sum_{i=1}^{m} a_i^2 = \sum_{i=1}^{m} p_i = 1$$

$$\text{under } H_0 : z_1 \sim N(0,1), z_2|n_2 \sim N(0,1), z_i|n_i \sim N(0,1)$$

$$a_1 z_1 + a_2 z_2 + ...n_m|n_m \sim N(0, a_1^2 + a_2^2 + \cdots + a_m^2),$$

$$\text{therefore, under } H_0 : T \sim N(0,1)$$

From the above derivation it can be seen that $p_i = a_i^2$ represents the proportion of the variance of T assigned to the i-th stage of the trial. Hence T is called a variance-spending statistic.

2.1 Example

In this investigation, without losing generality, we consider a paired two sample comparison, such as one eye randomly treated with a control method while the other eye is treated with the new treatment of interest. Let the observed difference of treatment effect between the treatment of interest and the control be $X_{ij} \sim N(\mu, 1)$ where $i = 1, 2$ is the stage while $j = 1, ..., n_i$ is the number of subjects at stage i. We are interested in testing $H_0 : \mu = 0$ and consider a two-stage clinical trial design (i.e. $m = 2$).

For this investigation, we assume a prespecified adaptive method for choosing the sample size for the second stage. Then at the first analysis we choose n_2 based on the first stage results according to $Z_1 = z_1$. Let $a_1 = 0.5$ and $n_1 = 33$ be fixed at design time. Thus we have decided to spend 50% of our weight at the first analysis. We then use a rule

$$n_2 = g(Z_1) = 0.5n_1 + 3.5n_1 \times \frac{\phi\left(\frac{z_1}{0.196\sqrt{n_1}} - 1\right)}{\phi(0)}$$

Under this plan, n_2 ranges from $0.5 \times n_1$ to $4 \times n_1$ with maximum occurring when $z_1 = 0.196\sqrt{n_1}$.

In practice, n_2 could be generated by more complicated methods. Our choice was governed by simplicity and its similarity to common practices. Figure 2.1 displays how n_2 varies as a function of the sample mean in the first stage. Such a rule provides approximately 97% power to detect an alternative $H_1 : \mu_1 = 0.35$. Note that relatively lower sample sizes are used when $\mu = 0$ or $\mu = 0.35$ than for intermediate alternatives. In the next section, we illustrate that this adaptive plan is roughly equivalent to the more commonly used GST with respect to operating characteristics.

2.2 Comparison to Group Sequential Tests: OBF, Pocock, Triangular

We can compare our adaptive SDCT to the more familiar GST with respect to such operating characteristics as power and average sample N (ASN). Although in a real situation, we would likely choose to use GST with more equal sample sizes, for comparability with the SDCT, we will choose GST having $n = 2$ and $n_1 = 33$. Boundary shapes considered will include the O'Brien Fleming (OBF), Pocock, and Triangular. Figure 2.2 displays such

boundaries for GST having 97% power to detect $\mu = 0.35$. We will consider, however, the comparability of GST to the SDCT when GST have either the same power or the same ASN as the SDCT for specific alternatives.

Group sequential tests have become a standard solution to the problem of increased overall false positive error rate during repeated significance testing in sequential monitoring. For a general one-sided group sequential design with a maximum of K stages of interim analyses, K pairs of stopping boundary values (b_k, c_k) are specified to maintain the overall false positive error rate equals to α. At the end of stage k where $1 < k < K$, a standardized test statistic U_k is calculated from all the data collected so far. $U_k = \frac{\sum_{i=1}^{k} n_i \overline{X}_i}{\sqrt{\sum_{i=1}^{k} n_i \sigma^2}}$. Under fixed sample testing $U_k \sim N(0,1)$. The trial will be terminated with rejection of alternative hypothesis if $U_k < c_k$ or with rejection of the null hypothesis if $U_k > b_k$. To ensure that the trial ends at $K - th$ stage, b_K is set to equal c_K. The most well-known group sequential tests include Pocock [1], O'Brien-Fleming [3], and Triangular tests [9]. Kittelson & Emerson described a unifying notation in which the above three tests correspond to $b_k = \mu_b + (A + \pi_k^{-P}(1 - \pi_k^R))G$ for appropriate choices of A,P,R,G and where $\pi_k = \frac{\sum_{i=1}^{k} n_i}{\sum_{i=1}^{m} n_i}$ [17].

We consider GST having $m = 2$, $n_1 = 33$, and either the same power to detect on alternative or the same ASN. Boundary shapes considered include the O'Brien-Fleming (OBF) [3], Pocock [1], and Triangular tests [9]. Figure 2.2 displays such boundaries for GST having 97% power to detect $\mu = 0.35$. With matched power, the ASN of SDCT is much larger than those of the other three designs when the true mean and the power are high (Table 2.1). When we compare the designs with matched ASN, clearly at the power range around 0.9, the SDCT showed the lowest power (Table 2.2). In a real GST, we would likely have choosen more equal group sizes and/or more frequent analyses, however, for comparability with SDCT we kept the same n_1.

Table 2.1 Comparison of the ASN Among Various Designs with Matched Power

True Mean	SDCT	OBF	Pocock	Tri
0.10	130.37	139.90	155.55	138.68
0.15	134.56	139.30	148.30	137.87
0.20	135.63	137.51	140.24	134.64
0.25	134.00	133.97	129.43	127.76
0.30	129.18	128.26	116.11	116.81
0.35	122.16	118.16	100.07	100.08
0.40	113.10	103.70	83.33	79.75
0.45	103.48	85.04	66.92	61.07
0.50	93.47	67.37	54.06	58.83

Table 2.2 Comparison of the Power Among Various Designs with Matched ASN

True Mean	SDCT	OBF	Pocock	Tri
0.10	0.2190	0.2074	0.1949	0.2120
0.15	0.4254	0.4137	0.3961	0.4210
0.20	0.6509	0.6449	0.6373	0.6587
0.25	0.8264	0.8265	0.8387	0.8478
0.30	0.9274	0.9290	0.9489	0.9503
0.35	0.9713	0.9749	0.9875	0.9884
0.40	0.9880	0.9921	0.9968	0.9979
0.45	0.9939	0.9979	0.9991	0.9997
0.50	0.9966	0.9995	0.9998	0.9999

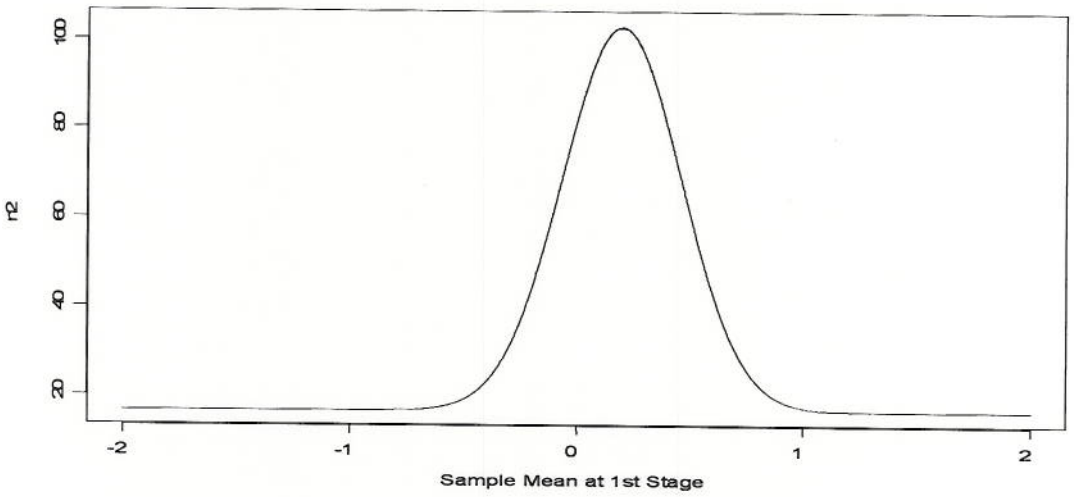

Figure 2.1: Generation of n2

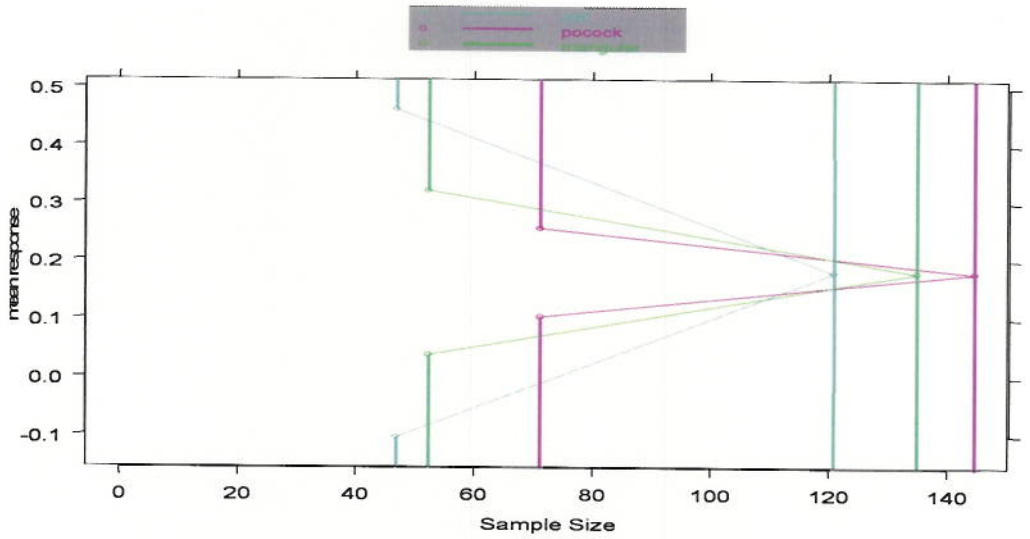

Figure 2.2: Comparison of stopping boundaries for GST designs

Chapter 3

STATISTICAL METHODOLOGY

Other authors have addressed the overall efficiency of SDCT compared to GST [16]. In this research, however, we assume a SDCT will be used, and we investigate more efficient alternatives to the test statistic T, which Fisher envisioned using as the primary test statistic. Specifically, we consider the MLE: $\overline{X}$, which is easily shown to be

$$\overline{X} = \frac{\sum_{i=1}^{m} n_i \overline{X}_i}{\sum_{i=1}^{m} n_i}$$

where m is the maximum number of stages. The group sizes, $n_1, n_2, ..., n_m$, can be random with n_i a function of $\overline{X}_1, ...\overline{X}_{i-1}$.

There are several criteria by which we can compare competing statistics for SDCT: the precision of the statistic as measured by width of CI; the bias of the point estimate derived from the statistic, and the power to test the true difference. Our interest is comparing efficiency of CI based on sampling distribution for each statistic, which when $m = 2$ are $T = a_1 Z_1 + a_2 Z_2$ and sample mean

$$\overline{X} = \frac{n_1 \overline{X}_1 + n_2 \overline{X}_2}{n_1 + n_2}$$

The density function for T can be found as

$$T|Z_1 \sim a_1 Z_1 + a_2 Z_2 | Z_1$$

, so

$$T|Z_1 \sim a_1 Z_1 + a_2 N\left(\sqrt{n_2(Z_1)}\mu, 1\right) \sim N\left(a_1 Z_1 + a_2\sqrt{n_2}\mu, a_2^2\right)$$

$$f_T(t) = \int_{-\infty}^{\infty} \frac{1}{a_2}\phi\left(\frac{t - a_1 Z_1 - a_2\sqrt{n_2}\mu}{a_2}\right) f(z_1) dz_1$$

$$f_T(t) = \int_{-\infty}^{\infty} \frac{1}{a_2}\phi\left(\frac{t - a_1 Z_1 - a_2\sqrt{n_2}\mu}{a_2}\right) \phi\left(z_1 - \sqrt{n_1}\mu\right)\, dz_1$$

Similarly, $\overline{X} = \frac{n_1\overline{X}_1 + n_2\overline{X}_2}{n_1+n_2}$ has density

$$f_{\overline{x}}(x) = \int \frac{n_1 + n_2}{\sqrt{n_2}}\phi\left(\frac{(n_1+n_2)x - n_1 x_1 - n_2\mu}{\sqrt{n_2}}\right) \sqrt{n_1}\phi\left(\sqrt{n_1}\left(x_1 - \mu\right)\right) dx_1$$

where for notational simplicity, we have suppressed the dependence of n_2 on z_1, x_1 in the above formulas.

3.1 Construction of confidence intervals

One can construct a confidence interval by inverting P-values computed under various hypotheses(Figure 3.1-3.2). In applying this approach, our statistic $S(\vec{X})$ defines an ordering of the outcome space. Two such orderings we consider are those based on the MLE sample mean and Fisher's T statistics. For each hypothesized μ, one can define acceptance regions based on some statistic, $S(\overline{X}_1, \overline{X}_2, ..., \overline{X}_m) : A_\mu^\alpha = \{\vec{x} : pr[S(\vec{X}) \geq S(\vec{x})|\mu] > \alpha\}$. We then invert these acceptance region to define $(1-\alpha)$ confidence sets $I^\alpha(\vec{x})$ for an observed $\vec{x}$.

$$I^\alpha(\vec{x}) = \{\mu : \vec{x} \in A_\mu^\alpha\}$$

In Figure 3.1, we display the 2.5th, 50th and 97.5th percentile of the distribution of $\overline{X}$ at each hypothesized value of μ. Thus for instance when $\mu = 0$, the 2.5th percentile of $\overline{X}$ is -0.264, the median of $\overline{X}$ is -0.01, the 97.5th percentile is 0.166. In order to use this information to find a MUE or 95% CI, we invert the quantile functions. So, for an observed $\overline{x} = 0.3$, we note that observation is the 97.5th percentile when $\mu = 0.468$ (the intersection of the the horizontal line at $\overline{x} = 0.3$ with the 97.5th quantile occurs when $\mu = 0.468$). Similarly $\overline{x} = 0.3$ is the 2.5th percentile when $\mu = 0.119$. Thus a 95% CI is $(0.119, 0.468)$ when $\overline{x} = 0.3$. By using the median contour we find the $MUE = 0.3$ in an analogous manner. Using Figure 3.2 in an analogous manner for an observed T, we find $MUE = 0.296$, 95% CI= $(0.120, 0.716)$.

In numerical integration, we did not observe nonmonotonicity in quantiles of distribution as a function of μ, hence any lack of stochactic ordering would not appear to have major impact on the existence of true CI. We can then construct the confidence intervals. We can evaluate the efficiencies of the two statistics by comparing the lengths of confidence intervals constructed based on corresponding orderings of the outcome space.

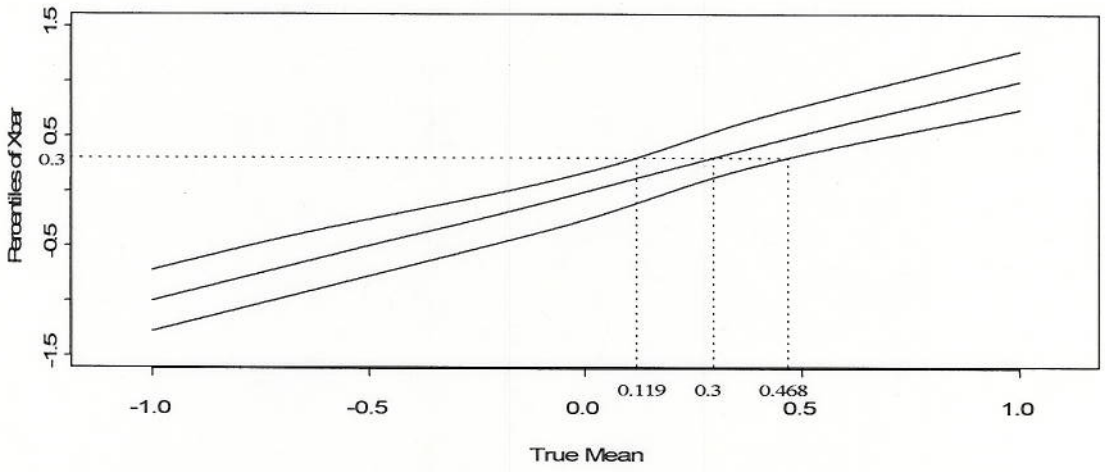

Figure 3.1: Integrated percentiles of Xbar

3.2 Point Estimates

Several methods of deriving point estimate have been explored for sequential tests [7]. The median unbiased estimate (MUE) for the unknown normal mean is defined as follows. Given an observed test statistic $S(\vec{X}) = s$, one can define a MUE $\tilde{\mu}_s$ based on $S(\vec{X})$ as

$$pr\{S(\vec{X}) > s|\tilde{\mu}\} = \frac{1}{2}$$

This is equivalent to assuming we observed the median value of our statistic. In this paper, we investigate the median unbiased estimate (MUE), $\tilde{\mu}_{SM}$ based on the sample mean ordering, and $\tilde{\mu}_T$, based on the T ordering. The definition of percentiles of our estimates and the concepts of finding MUE/CI by inverting are demonstrated in an example showed in Figure 3.1-3.2.

The other estimate we used is bias adjusted mean estimate (BAM), $\breve{\mu}$, defined by

$$E(S(\vec{X})|\breve{mu}) = s$$

which is independent of any particular ordering of the outcome space. This is equivalent to assuming we observe the mean value of our statistic. We investigated the BAM $\breve{\mu}_{SM}$, which is based on sample mean observation, and $\breve{\mu}_T$, which is based on T.

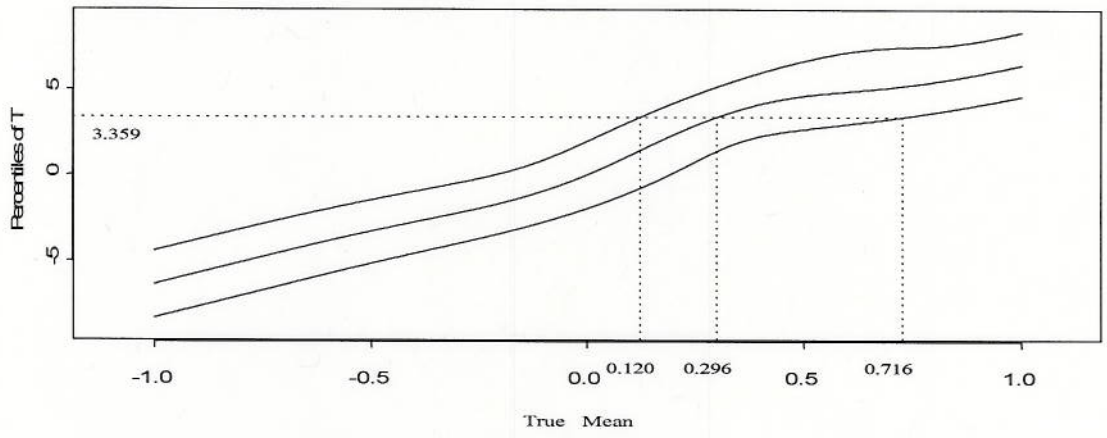

Figure 3.2: Integrated percentiles of T

We compare the bias between the estimates of the unknown true mean generated through $\overline{X}$ and T using both BAM and MUE.

3.3 Generation of the tables and powers

For those numerical results mentioned above, the density was numerically integrated with simulations used to verify program accuracy using a program written in Splus. We tabled the expectation and the 2.5%, 50% (median), and 97.5% quantiles of the distribution of $\overline{X}$ and T for a range of θ. These tables were then inverted to find confidence bounds and point estimates for any particular value of $(\overline{x}_1, \overline{x}_2)$: given $\overline{x}_1, \overline{x}_2$, compute $T = t$ and $\overline{X} = x$, then find θ_L so that observed statistic $\overline{x}$ or T is 97.5 percenntile of respective distribution. The intervals of tabulation were small enough to provide values accurate to 0.001. Using the density functions of $\overline{X}$ and T ((1) and (2)), the values of acceptance regions, medians, expected values of $\overline{X}$ and T, and the powers were computed for values of the true mean ranging from -2 to 2 in intervals of 0.001 using numerical integration.

3.4 Measures used to compare $\overline{X}$ and T

The integrated (or simulated) expected values and percentiles for both $\overline{X}$ and T were tabulated to get the confidence interval bounds and point estimates by inversion using the methods described in 2. In addition to the power, we also calculate two more measures to compare the behavior of $\overline{X}$ and T: efficiency and bias. To compare the efficiency of T relative to $\overline{X}$, we define

$$\text{efficiency} = \frac{\text{length of bound for } \overline{X}}{\text{length of bound for T}}$$

If efficiency $= 1$, then $\overline{X}$ and T are equally efficient. If efficiency < 1, then $\overline{X}$ is more efficient than T.

Another important factor concerning the statistics used is the bias of the statistics when estimating the true mean μ. We compared point estimates developed by $\overline{X}$ and T using BAM and MUE methods as mentioned in 2. We define Bias as $B = E(\hat{\mu} - \mu)$, where $\hat{\mu}$ is the estimate of true mean by $\overline{X}$ or T through either BAM or MUE.

Chapter 4

RESULTS

As anticipated, with increased treatment effect (the absolute values of the true mean $|\mu|$), the power increased (Figure 4.1). Figure 4.2 shows the absolute difference between the powers of the tests using $\overline{X}$ and T respectively for each true mean μ. We can see tests using $\overline{X}$ have higher power than those using T when the averaged power is high. Clinical trials designs generally strive for high power to make sure the true benefitial effects will not be missed. The lower power of T means T might be less efficient and could result in more trial stages and bigger sample size.

As shown in Figure 4.3, $\overline{X}$ is more efficient because the efficiency is always less than 1. Weighting each alternative between -2 and 2 equally, the mean efficiency is only 0.804, suggesting that on average, the length of the confidence bound developed by $\overline{X}$ is only 80.4% of that when using T.

The biases caused by $\overline{X}$ and T for MUE and BAM are showed in Figure 4.4 and 4.6 respectively. We also checked the difference between the bias caused by $\overline{X}$ and T. As shown in Figure 4.5 and 4.7. In both BAM and MUE methods, the difference (bias of T -bias of $\overline{X}$) is most often larger than 0. Therefore the T statistics caused higher bias for both the BAM and MUE. Table 4.1 shows the average bias for $\overline{X}$ and T for BAM and MUE when weighing each alternative between -2 and 2 equally.

Table 4.1 Bias of point estimates using $\overline{X}$ and T

Method	$\overline{X}$	T
BAM	0.003164±0.002327	0.01709±0.01696
MUE	0.007865±0.004976	0.01776±0.01848

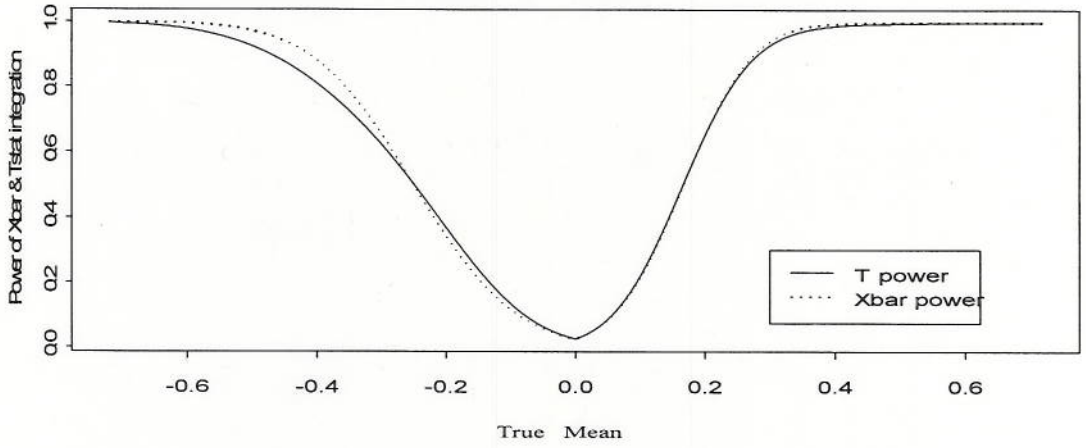

Figure 4.1: Power using Xbar and T via integration

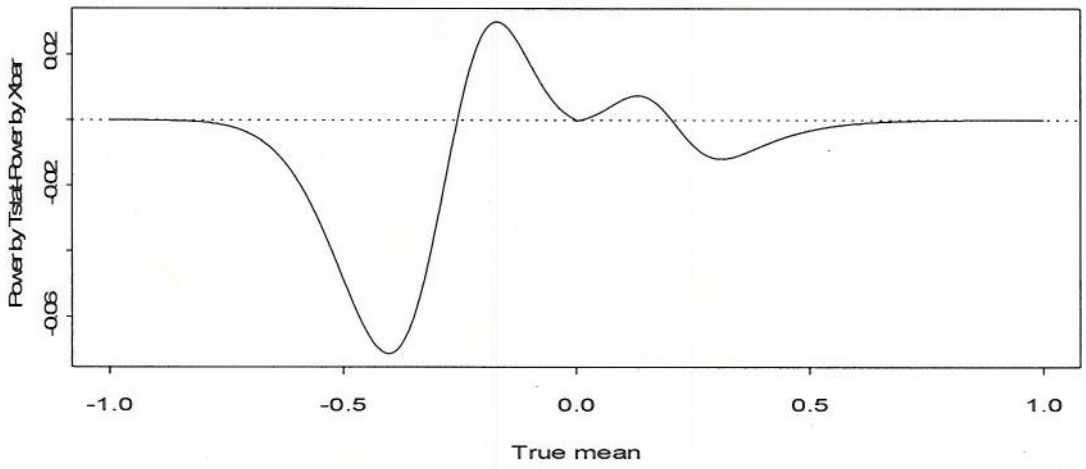

Figure 4.2: T power - Xbar power by integration

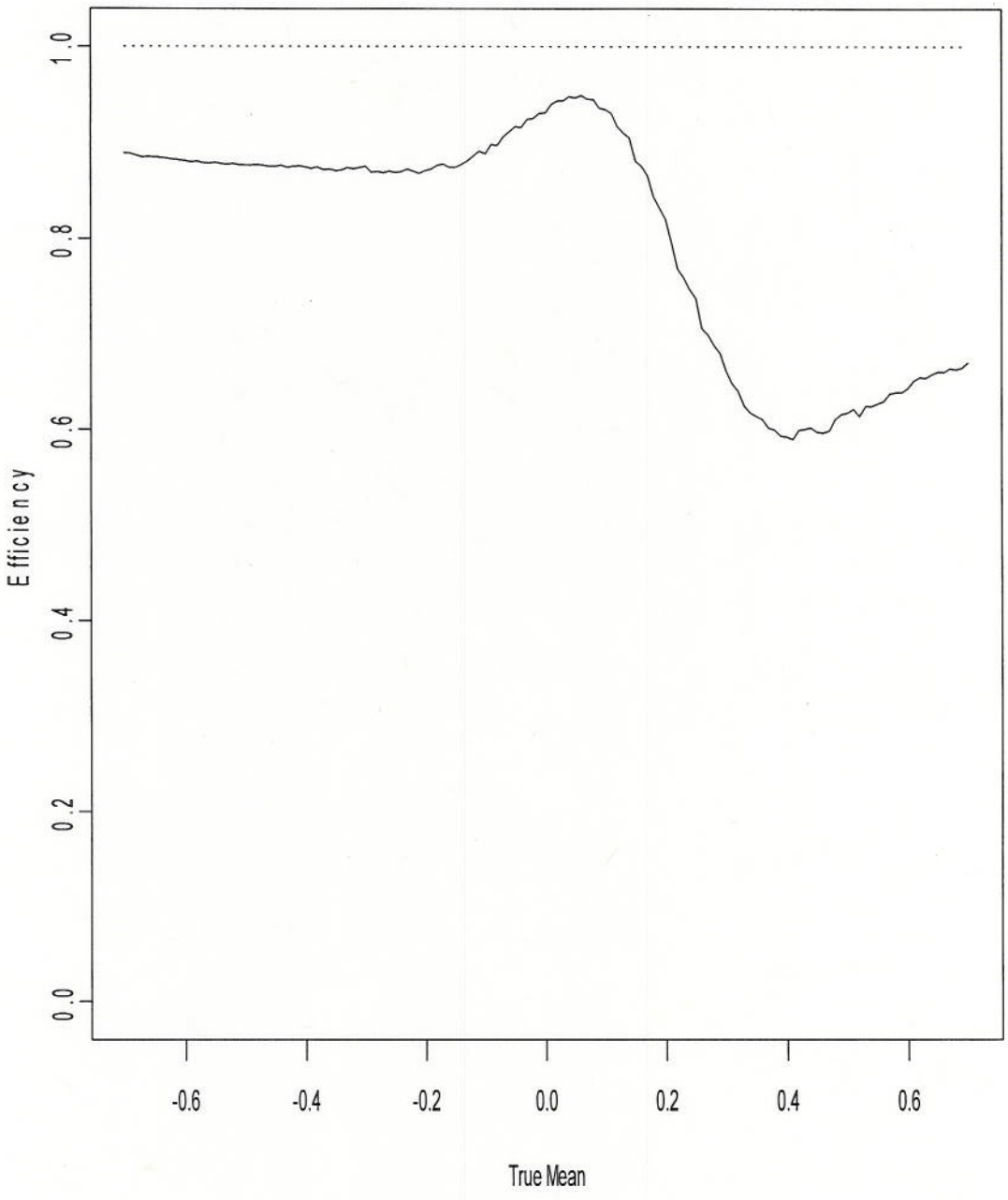

Figure 4.3: Efficiency of Xbar and T by integration

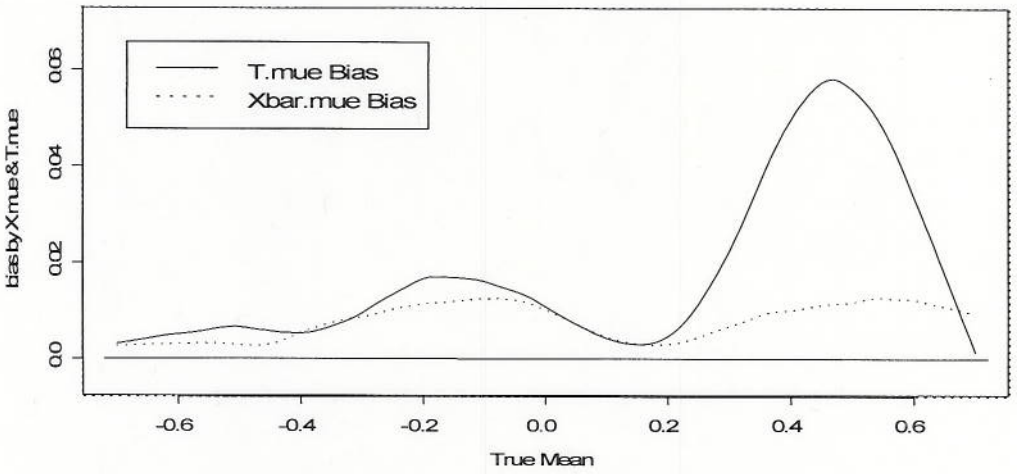

Figure 4.4: Integrated bias of Xbar.mue and T.mue

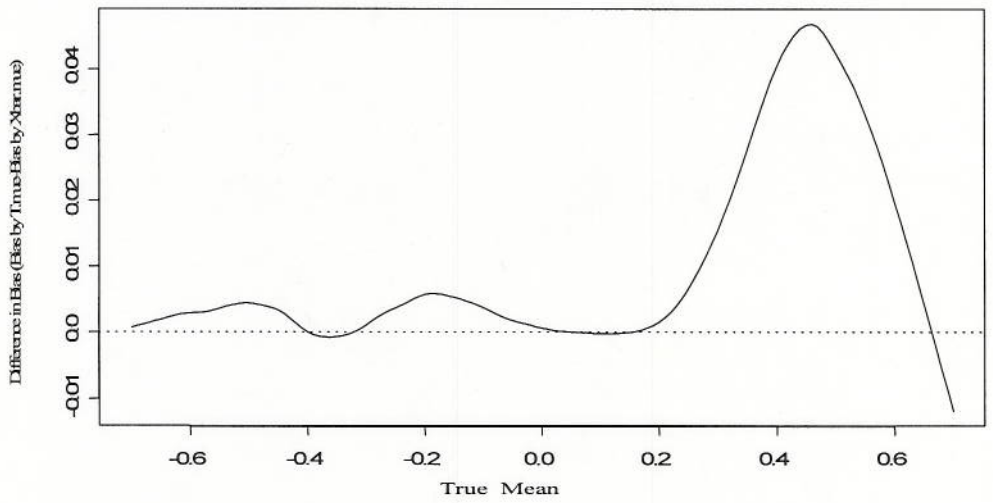

Figure 4.5: Difference of bias between Xbar.mue and T.mue

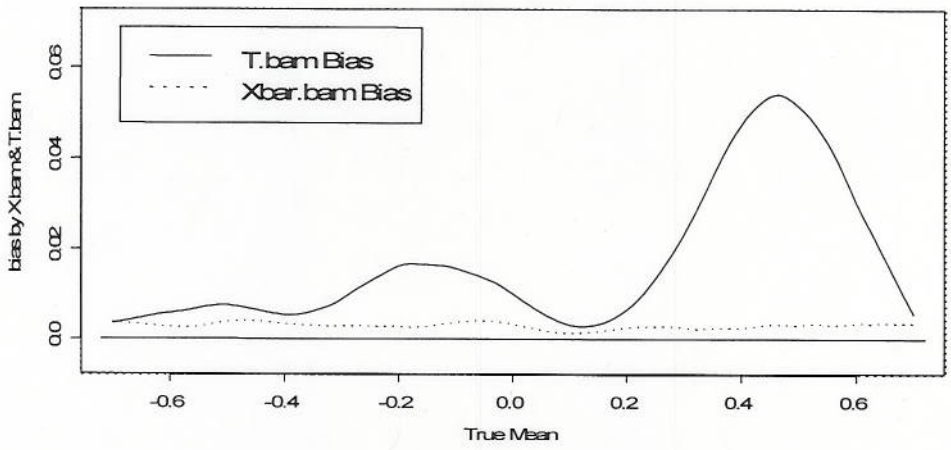

Figure 4.6: Integrated bias of Xbar.bam and T.bam

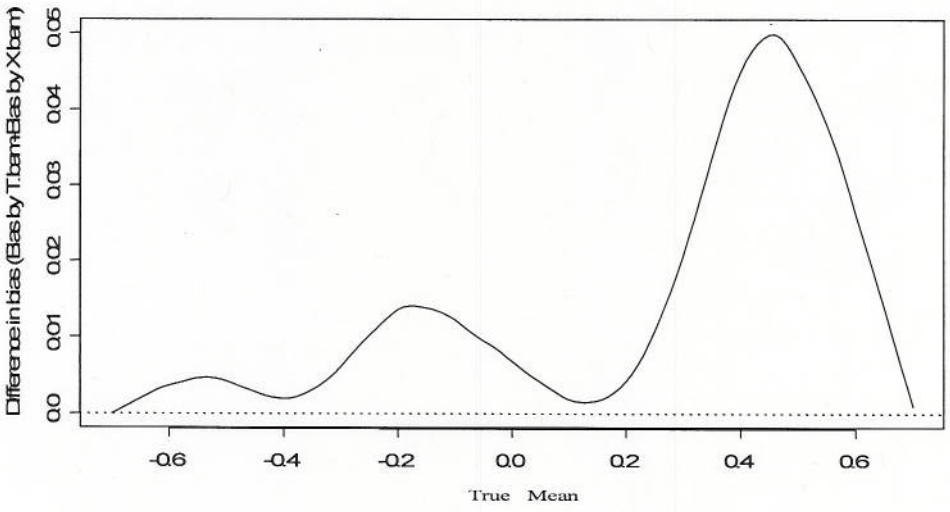

Figure 4.7: Difference of bias between Xbar.bam and T.bam

Chapter 5

DISCUSSION

In this paper, we investigated estimation following self-designing clinical trials. We used numerical integration to do the above estimations and the results were confirmed by simulation study. The performance of the statistic T used in SDCT were compared to that of the sample mean. From our investigations, the use of T statistic may contribute greatly to the inefficiency of SDCT.

1) In terms of the capability to detect the true treatment effect, T seems to be less efficient because the average confidence interval of T is wider than that of $\overline{X}$ across all alternatives examined.

2) The power of the tests using T to detect the same treatment effect is often worse than that using $\overline{X}$ under the same settings, especially for those alternatives for which power is high.

3) When used for point estimation, the bias is also worse using T than using $\overline{X}$ via both BAM and MUE approaches.

Comparisons among SDCT, Pocock, O'Brien-Fleming, and Triangular group sequential designs also showed worse performance of SDCT under some settings. Worse power with matched average sample sizes and larger sample size with matched power were observed with SDCT design for alternatives corresponding to high power.

Fisher argued that one of the advantages of SDCT is that when the original estimation of the true treatment effect is not accurate, SDCT can automatically adjust the study stages and sample sizes based on data obtained so far. However, classical group sequential designs can also provide such adjustment via earlier stopping or later extension based on interim analysis results.

There exist certain limitations with our study. We only looked at one stopping rule. We did demonstrate that we can generally improve the efficiency of the SDCT by using MLE instead of Fisher's statistic in this case. However, though our stopping rule was generally

in the spirit of OBF, Pocock, and Triangular tests, the exact gains in efficiency may not generalize to settings which use other rules for n_2 generation. Nonetheless, it does seem reasonable that the inefficiency of Fisher's T statistic would be observed in other settings.

Lastly we note that our investigation presumed prior knowledge of the entire stopping rule. Such knowledge may not always be present when an adaptive procedure such as SDCT is desired. However, we note again that careful evaluation of a clinical trial design may obviate the need for adaptive procedures which are based on interium estimates of the treatment effect. Furthermore, since SDCT designs are less efficient than nonadaptive designs [16], such prespecification of stopping rules would seem preferable whenever possible.

BIBLIOGRAPHY

[1] POCOCK, S. J. (1977). *Group sequential methods in the design and analysis of clinical trials.* Biometrika 60, 191-9.

[2] POCOCK, S. J. (1982). *Interim analyses for randomized clinical trials: The group sequential approach.* Biometrics 38, 153-62.

[3] O'BRIEN, P. C. & FLEMING, T. R. (1979). *A multiple testing procedure for clinical trials.* Biometrics 35, 549-56.

[4] DEMETS, D. L. & WARE, J. H. (1980). *Group sequential methods in clinical trials with a one-sided hypothesis.* Biometrika 67, 651-60.

[5] DEMETS, D. L. & WARE, J. H. (1982). *Asymmetric group sequential boundaries for monitoring clinical trials.* Biometrika 69, 661-3.

[6] EMERSON, S. S. & FLEMING, T. R. (1989). *Symmetric group sequential test designs.* Biometrics 45, 905-23.

[7] EMERSON, S.S. & FLEMING, T. R. (1990). *Parameter estimation following group sequential hypothesis testing.* Biometrika 77, 875-92.

[8] LAN, K. K. G. & DEMETS, D. L. (1983). *Discrete sequential boundaries for clinical trials.* Biometrika 70, 659-63.

[9] WHITEHEAD, J. (1983). *The design and analysis of sequential clinical trials.* Chichester: EllisHorwood.

[10] WHITEHEAD, J. (1986). *On the bias of maximum likelihood estimation following a sequential test.* Biometrika 73, 573-81.

[11] WHITEHEAD, J. & STRATTON, I. (1983). *Group sequential clinical trials with triangular continuation regions.* Biometrics 39, 227-36.

[12] FLEMING, T. R., HARRINGTON, D. P. & O'BRIEN, P. C. (1984). *Designs for group sequential tests.* Controlled Clin. Trials 5, 548-61.

[13] FISHER, L.D. (1998). *Self-designing clinical trials.* Statistics in Medicine 17, 1551-62.

[14] SHEN, Y. & FISHER L.D. (1999). *Statistical inference for self-designing clinical trials with a one-sided hypothesis.* Biometrics 55, 190-97.

[15] THACH, C.T. & FISHER, L.D. (2002). *Self-designing two-stage trials to minimize expected costs.* Biometrics 58, 432-38.

[16] TSIATIS, A.A. & MEHTA, C. (2003). *On the inefficiency of the adaptive design for monitoring clinical trials.* Biometrika 90, 367-78.

[17] KITTELSON, J. M. & EMERSON, S.S. (1999)*A unifying family of group sequential test designs.*Biometrics 55, 874-82.

[18] EMERSON, S. S. $ KITTELSON, J. M. (2003) *Evaluation of group sequential clinical trial designs.* University of Washington Biostatistics Working Paper Series. Working Paper 216. http://www.bepress.com/uwbiostat/paper216.

Appendix A

VERIFICATION BY SIMULATION

The above comparisons between $\overline{X}$ and T were confirmed by simulation study. The settings as described in section 3.3 notation part were simulated 100,000 times to calculate the mean, 2.5, 50, and 97.5 percentiles of $\overline{X}$ and T. These simulated numerical values used to generate the table were compared to the integrated ones in Figure A.1-A.2. The results of these two studies agreed well.

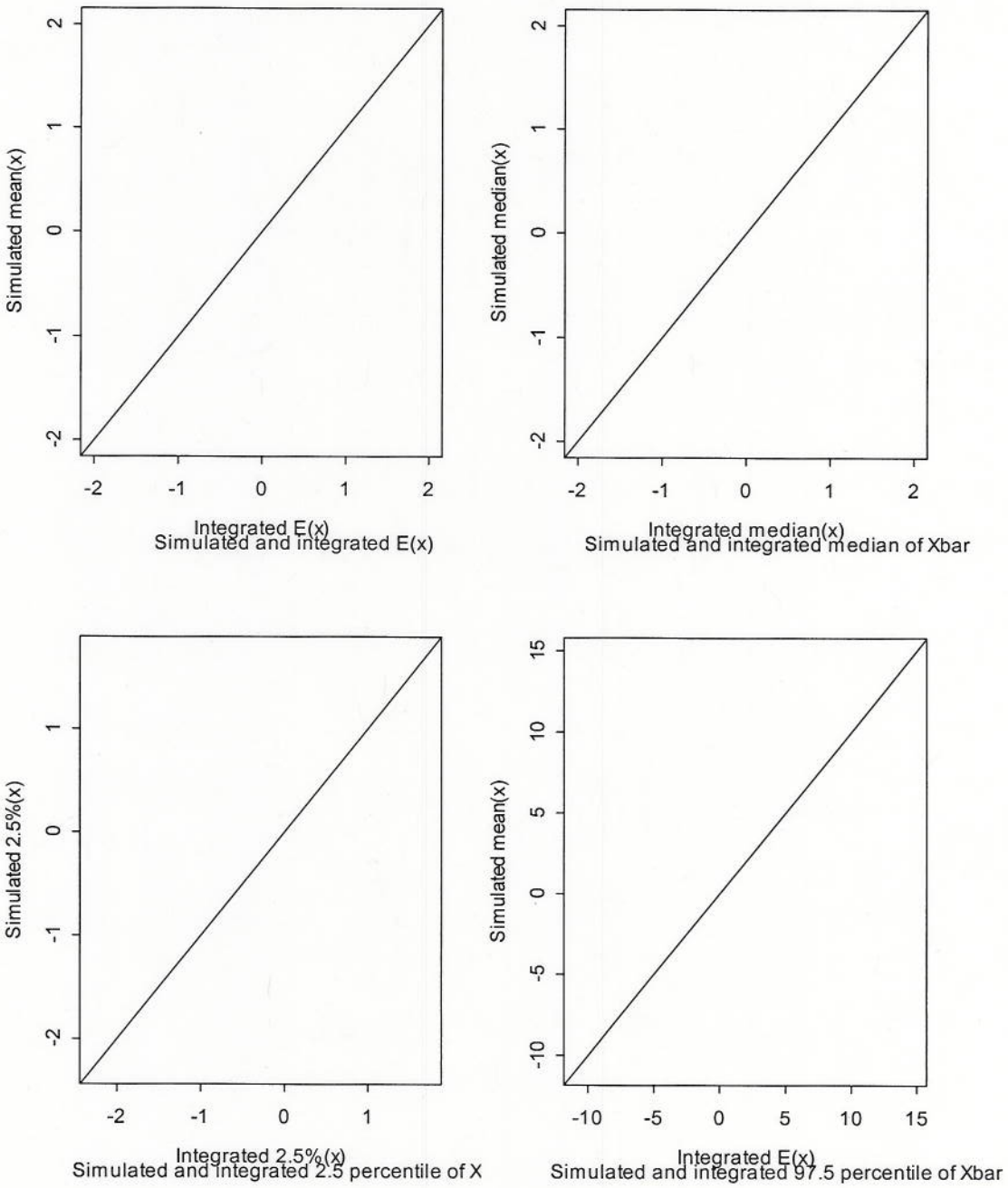

Figure A.1: Comparison between integrated and simulated Xbar statistics

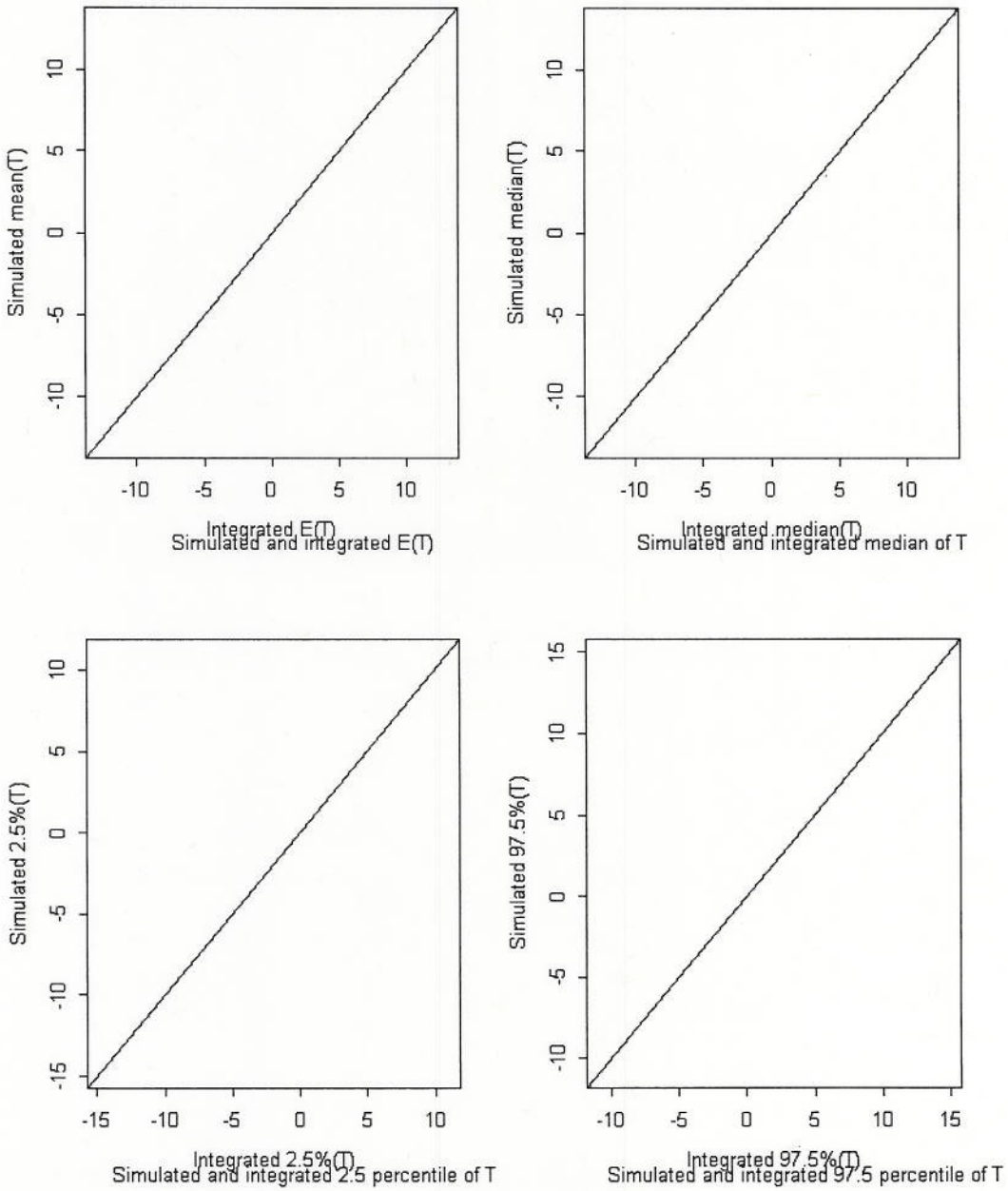

Figure A.2: Comparison between integrated and simulated T statistics

Appendix B

SPLUS CODE

```
##### By Integration

### Generate integration result table: InteTable

# moments function to do numerical integration

moments<-function (x0,xn,k,f){

   #integral of f(x) from x0 to xn divided into k intervals
   b<-c(7,rep(c(32,12,32,14),k))
   b[length(b)]<-7
   x<-seq(x0,xn,length=length(b))
   h<-x[2]-x[1]
   area<-sum((2*h/45)*b*f(x))
   area
}

parm <- c(0.5, 4, 0.196, 0.196)

# g is the function to calculate n2 based on x1
g <- function (x) {
        lo <- n1 * parm[1]
        hi <- n1 * parm[2]
        cntr <- parm[3]
        s <- parm[4]
        lo + (hi - lo) * dnorm(x, cntr, s) / dnorm(0, 0, s)
}

# gz another function to calculate n2 based on z1
gz<-function (z1) {
        lo <- n1 * parm[1]
        hi <- n1 * parm[2]
        cntr <- parm[3]
        s <- parm[4]
        lo + (hi - lo) * dnorm(z1/sqrt(n1),cntr, s) / dnorm(0, 0, s)
}

#f2<-function (x,y,a1,a2,mu,n1) {pnorm(y/a2,-a1*x/a2-
sqrt(g(x))*mu,a2)*dnorm(x,sqrt(n1)*mu,1)}
#f1<-function (x,y,a1,a2,mu,n1) {pnorm(y/a2,-a1*x/a2-
sqrt(g(x))*mu)*dnorm(x,sqrt(n1)*mu,1)}
#x<-seq(-10,10,0.01)
#a1<-0.5
#a2<-sqrt(1-a1^2)
```

```
#mu<-0
#n1<-33
#cumsum(f(x,y,a1,a2,mu,n1))

f<-function (x) {pnorm(y/a2,a1*x/a2+sqrt(gz(x))*mu,1)* dnorm(x,
sqrt(n1)*mu,1)}

hit<-NULL
met<-NULL
lot<-NULL
for (mu in seq(-2,2,0.001))
{

      z<-seq(sqrt(n1)*mu-10,sqrt(n1)*mu+10,0.001)
      zp<-NULL
      for (i in c(1:length(z)))
     {
      y<-z[i]
       x<-mu
      zp[i]<-moments(-10,10,100,f)
     }
      lot<-c(lot,max(z[zp<=0.025]))
      met<-c(met,max(z[zp<=0.5]))
       hit<-c(hit,max(z[zp<=0.975]))
}


# E(T|mu) Calculation

f<-function (x) {a2*sqrt(gz(x))*mu*exp((-0.5)*(x-
sqrt(n1)*mu)^2)/sqrt(2*pi)}
mus<-seq(-2,2,0.001)
i<-0
et<-NULL
for (mu in mus) {
      i<-i+1
      et[i]<-moments(-10,10,1000,f)+a1*sqrt(n1)*mu
      et
}

# Xbar integration


# now the Xbar. E(xbar|mu)
f<-function (x)
    {
      sqrt(n1)/sqrt(2*pi)*exp((-0.5)*((x-
mu)*sqrt(n1))^2)*(n1*x+gz(sqrt(n1)*x)*mu)/(n1+gz(sqrt(n1)*x))
    }

EXbar<-NULL
i<-0
for (mu in seq(-2,2,0.001))
{
      i<-i+1
```

```
      EXbar[i]<-moments(mu-10,mu+10,1000,f)
}

# now the integration of ci of EXbar.

f<-function (x)
{

pnorm(y,(n1*x+g(x)*mu)/(n1+g(x)),g(x)/((n1+g(x))*sqrt(g(x))))*dnorm(x,m
u,1/sqrt(n1))
}

i<-0
hix<-NULL
lox<-NULL
mus<-seq(-2,2,0.001)
for (mu in seq(-2,2,0.001))
{
      i<-i+1
      j<-0
      p<-NULL
      ys<-seq(mu-10,mu+10,0.001)
      for (y in seq(mu-10,mu+10,0.001))
   {
            j<-j+1

            p[j]<-moments(mu-10,mu+10,100,f)
   }
      lox[i]<-max(ys[p<=0.025])
      mex[i]<-max(ys[p<=0.5])
       hix[i]<-max(ys[p<=0.975])
}

# generate InteTable
mu<-seq(-2,2,0.001)
InteTable<-cbind(mu,EXbar,lox,mex,hix,et,lot,met,hit)


# generate Xbar.bam,Xbar.mue,Tstat.bam,and Tstat.mue via integration
after the InteTable is generated.

selfDesignInferenceA <- function (Xbar, Tstat, InteTable) {

        interpolateA <- function (obj, col, InteTable) {
                N <- dim(InteTable)[1]
                lo <- sum (InteTable[-1,col] <= obj) + 1
                lo <- ifelse(lo == 1, 2, ifelse(lo == N, N-1, lo))
                InteTable[lo,1] + (obj - InteTable[lo,col]) /
(InteTable[lo+1,col] - InteTable[lo,col]) * (InteTable[lo+1,1] -
InteTable[lo,1])
        }

        Xbar.bam <- interpolate (Xbar, 4, InteTable)
        Xbar.mue <- interpolate (Xbar, 6, InteTable)
```

```
        Xbar.ci <- c(interpolate (Xbar, 7, InteTable), interpolate
(Xbar, 5, InteTable))
        Tstat.bam <- interpolate (Tstat, 9, InteTable)
        Tstat.mue <- interpolate (Tstat, 11, InteTable)
        Tstat.ci <- c(interpolate (Tstat, 12, InteTable), interpolate
(Tstat, 10, InteTable))

c(Xbar,Xbar.bam,Xbar.mue,Xbar.ci,Tstat,Tstat.bam,Tstat.mue,Tstat.ci)
}

# generate the bam,mue table

selfDesignInferenceInte <- function (Xbar, Tstat, InteTable) {

        interpolateA <- function (obj, col, InteTable) {
                N <- dim(InteTable)[1]
                lo <- sum (InteTable[-1,col] <= obj) + 1
                lo <- ifelse(lo == 1, 2, ifelse(lo == N, N-1, lo))
                InteTable[lo,1] + (obj - InteTable[lo,col]) /
(InteTable[lo+1,col] - InteTable[lo,col]) *

                        (InteTable[lo+1,1] - InteTable[lo,1])
        }

        Xbar.bam <- interpolateA (Xbar, 4, InteTable)
        Xbar.mue <- interpolateA (Xbar, 6, InteTable)
        Xbar.ci <- c(interpolateA (Xbar, 7, InteTable), interpolateA
(Xbar, 5, InteTable))
        Tstat.bam <- interpolateA (Tstat, 9, InteTable)
        Tstat.mue <- interpolateA (Tstat, 11, InteTable)
        Tstat.ci <- c(interpolateA (Tstat, 12, InteTable), interpolateA
(Tstat, 10, InteTable))

c(Xbar,Xbar.bam,Xbar.mue,Xbar.ci,Tstat,Tstat.bam,Tstat.mue,Tstat.ci)
}

# generate the power for different true mus, like simulation method,
use the 2.5 and 97.5 percentile of that when mu=0. calculate belta for
that percentile at N(mu,1)..1-belta is the power.
mus<-seq(-2,2,0.001)
# check whether lox,hix,lot,and hit has the same length as mus.
ci0<-c(lox[mus==0],hix[mus==0],lot[mus==0],hit[mus==0])
xpower.int<-NULL
tpower.int<-NULL
for (i in 1:length(mus)) {
      xpower.int<-
c(xpower.int,pnorm(ci0[1],mus[i],1)+pnorm(ci0[2],mus[i],1))
      tpower.int<-
c(tpower.int,pnorm(ci0[3],mus[i],1)+pnorm(ci0[4],mus[i],1))
}

### generate the tables contains bias and efficiency using integrated
data: InteTable

mus<-seq(-0.7,0.7,0.001)
```

```
effi.Int<-NULL
effiv.Int<-NULL
bias.Int<-NULL
biasv.Int<-NULL

for (k in 1:length(mus)) {

  z <- simSelfDesign (n1 = 33, a1= 0.5, mu=mus[k], parm, n2g, N=1000)
            Inference <- NULL
            for (i in 1:1000) Inference <- rbind(Inference,
                    selfDesignInferenceA (z$Xbar[i], z$Tstat[i],
InteTable))

      effi.Int<-c(effi.Int,mean((Inference[,5]-
Inference[,4])/(Inference[,10]-Inference[,9])))
      effiv.Int<-c(effiv.Int,var((Inference[,5]-
Inference[,4])/(Inference[,10]-Inference[,9])))
      bias.Int<-rbind(bias.Int,apply(Inference[,c(2,3,7,8)],2,mean))
      biasv.Int<-rbind(biasv.Int,apply(Inference[,c(2,3,7,8)],2,var))
}

## to generate bias via mue and bam, as well as bias differences
between xbar and T

biasInt.xb<-abs(bias.Int[,2]-bias.Int[,1])
biasInt.xm<-abs(bias.Int[,3]-bias.Int[,1])
biasInt.tb<-abs(bias.Int[,4]-bias.Int[,1])
biasInt.tm<-abs(bias.Int[,5]-bias.Int[,1])

biasInt.bam<-biasInt.xb-biasInt.tb
biasInt.mue<-biasInt.xm-biasInt.tm


## Use simulation data

mus<-seq(-0.7,0.7,0.01)
effi.xt33<-NULL
effiv.xt33<-NULL
bias.xt33<-NULL
biasv.xt33<-NULL

for (k in 1:length(mus)) {

      z <- simSelfDesign (n1 = 33, a1= 0.5, mu=mus[k], parm, n2g,
N=1000)
            Inference <- NULL
            for (i in 1:1000) Inference <- rbind(Inference,
                    selfDesignInference (z$Xbar[i], z$Tstat[i],
Table33))

      effi.xt33<-c(effi.xt33,mean((Inference[,5]-
Inference[,4])/(Inference[,10]-     Inference[,9])))
      effiv.xt33<-c(effiv.xt33,var((Inference[,5]-
Inference[,4])/(Inference[,10]-Inference[,9])))
      bias.xt33<-rbind(bias.xt33,apply(Inference[,c(2,3,7,8)],2,mean))
```

```
      biasv.xt33<-rbind(biasv.xt33,apply(Inference[,c(2,3,7,8)],2,var))
      }


##### By Simulation


# Here is the n2g function that generates n2 based on x1, n1, and
parm(parameters preselected).

n2g <- function (x1, n1, parm) {
        lo <- n1 * parm[1]
        hi <- n1 * parm[2]
        cntr <- parm[3]
        s <- parm[4]
        lo + (hi - lo) * dnorm(x1/n1, cntr, s) / dnorm(0, 0, s)
}


simSelfDesign <- function (n1, a1, mu, parm, n2g, Nrpt=100000) {
        x1 <- rnorm(Nrpt, n1 * mu, sqrt(n1))
        z1 <- x1 / sqrt(n1)
        n2 <- n2g (x1, n1, parm)
        x2 <- rnorm(Nrpt, n2 * mu, sqrt(n2))
        z2 <- x2 / sqrt(n2)
        Tstat <- a1 * z1 + sqrt(1 - a1^2) * z2
        Xbar <- (x1 + x2) / (n1 + n2)
        list(Tstat=Tstat, Xbar=Xbar, n2=n2)
}

# simSelfDesign simulate the results of a self design analysis. Input
values include n1, a1, mu, parm, n2g, and Nrpt. It output three
vectors:n2, Tstat and Xbar for comparison. Here a2=sqrt(1-a1^2) to let
Tstat ~N(0,1) distribution under the Null hypothesis: mu=0.

parm <- c(0.5, 4, 0.196, 0.196)
mu <- seq(-2, 2, 0.001)

z <- simSelfDesign (n1 = 33, a1= 0.5, mu= 0, parm, n2g)
Table33 <- matrix(c(NA, NA, 0.025, mean(z$Xbar), quantile(z$Xbar,
c(0.025, 0.5, 0.975)),
        0.025, mean(z$Tstat), quantile(z$Tstat, c(0.025, 0.5,
0.975))),nrow=1)

for (m in mu) {
        z <- simSelfDesign (n1 = 33, a1= 0.5, mu= m, parm, n2g)
        Table33 <- rbind(Table33,
                c(m, mean(z$n2), sum(z$Xbar > Table33[1,7] |
z$Xbar<Table33[1,5]) / length(z$Xbar), mean(z$Xbar),
                        quantile(z$Xbar, c(0.025, 0.5, 0.975)),
                        sum(z$Tstat > Table33[1,12] |
z$Tstat<Table33[1,10]) / length(z$Tstat), mean(z$Tstat),
                        quantile(z$Tstat, c(0.025, 0.5, 0.975))))
}
```

```
# the variables in Table33: 1/ mu, 2/ mean(n2) for mu, 3/ ratio of
Xbars when m=mu larger than the 97.5% of that when mu=0, 4/ mean(Xbar)
for mu, 5/ 2.5, 6/ 50, and 7/ 97.5 quantiles of Xbar for mu, 8/ ratio
of Tstat when mu that are larger than the 97.5% Tstat when mu=0, 9/
mean(Tstat) for mu, 10/ 2.5, 11/ 50, 12/ 97.5 quantiles of Tstat for
mu.

selfDesignInference <- function (Xbar, Tstat, Table33) {

        interpolate <- function (obj, col, Table33) {
                N <- dim(Table33)[1]
                lo <- sum (Table33[-1,col] <= obj) + 1
                lo <- ifelse(lo == 1, 2, ifelse(lo == N, N-1, lo))
#Generate lo, if lo=1 -->lo=2, if lo=N --> lo=N-1; else lo=number of
mean(obj) for various mus that are less or equal to obj.

                Table33[lo,1] + (obj - Table33[lo,col]) /
(Table33[lo+1,col] - Table33[lo,col]) *

                        (Table33[lo+1,1] - Table33[lo,1])
        }

        Xbar.bam <- interpolate (Xbar, 4, Table33)
        Xbar.mue <- interpolate (Xbar, 6, Table33)
        Xbar.ci <- c(interpolate (Xbar, 7, Table33), interpolate (Xbar,
5, Table33))
        Tstat.bam <- interpolate (Tstat, 9, Table33)
        Tstat.mue <- interpolate (Tstat, 11, Table33)
        Tstat.ci <- c(interpolate (Tstat, 12, Table33), interpolate
(Tstat, 10, Table33))

c(Xbar,Xbar.bam,Xbar.mue,Xbar.ci,Tstat,Tstat.bam,Tstat.mue,Tstat.ci)
}


mus<-seq(-0.7,0.7,0.01)
effi.xt33<-NULL
effiv.xt33<-NULL
bias.xt33<-NULL
biasv.xt33<-NULL

for (k in 1:length(mus)) {

      z <- simSelfDesign (n1 = 33, a1= 0.5, mu=mus[k], parm, n2g,
N=1000)
            Inference <- NULL
            for (i in 1:1000) Inference <- rbind(Inference,
                    selfDesignInference (z$Xbar[i], z$Tstat[i],
Table33))

      effi.xt33<-c(effi.xt33,mean((Inference[,5]-
Inference[,4])/(Inference[,10]-     Inference[,9])))
      effiv.xt33<-c(effiv.xt33,var((Inference[,5]-
Inference[,4])/(Inference[,10]-Inference[,9])))
      bias.xt33<-rbind(bias.xt33,apply(Inference[,c(2,3,7,8)],2,mean))
```

```
      biasv.xt33<-rbind(biasv.xt33,apply(Inference[,c(2,3,7,8)],2,var))
      }


## to generate bias via mue and bam, as well as bias differences
between xbar and T

bias.xb<-abs(bias.xt33[,2]-bias.xt33[,1])
bias.xm<-abs(bias.xt33[,3]-bias.xt33[,1])
bias.tb<-abs(bias.xt33[,4]-bias.xt33[,1])
bias.tm<-abs(bias.xt33[,5]-bias.xt33[,1])

bias.bam<-bias.xb-bias.tb
bias.mue<-bias.xm-bias.tm


# to generate the ASN and power table
# generate the position of the target true mu
TrueMean<-NULL
pos<-seq(1402,2602,100)
asn<-NULL
xpower<-NULL
tpower<-NULL
for (i in pos){
      TrueMean<-c(TrueMean,InteTable[i,1])
      asn<-c(asn,Table33[i,2]+33)
      xpower<-c(xpower,InteTable[i,3])
      tpower<-c(tpower,InteTable[i,8])
       }

ASNtable<-cbind(TrueMean,xpower,tpower,asn)
ASNtable



##### Generate Figures and Tables

# figure 1:  n2 generation
par(mfrow=c(1,1))
Table33[-1,1]->mu1
Table33[-1,2]->n2
plot(mu1,n2,type='l',xlab='Sample Mean at 1st Stage',ylab='n2')
title(sub="Fig. 1 Generation of n2")


# integrated e and percentiles
mus<-seq(-2,2,0.001)

par(mfrow=c(2,1))
# Percentiles
numb<-seq(1001,3001)
InteTable[-1,1]->muf1
muf1<-muf1[numb]
InteTable[-1,7]->xhib
xhib<-xhib[numb]
```

```
InteTable[-1,6]->xmid
xmid<-xmid[numb]
InteTable[-1,5]->xlob
xlob<-xlob[numb]


plot(muf1,xhib,type='l',lty=1,xlim=c(-1.1,1.1),ylim=c(-
1.5,1.5),xlab='',ylab='')
par(new=T)
plot(muf1,xmid,type='l',lty=1,xlim=c(-1.1,1.1),ylim=c(-
1.5,1.5),xlab='',ylab='')
par(new=T)
plot(muf1,xlob,type='l',lty=1,xlim=c(-1.1,1.1),ylim=c(-
1.5,1.5),xlab="True Mean",ylab="Percentiles of Xbar")
title(sub="Fig.3 Integrated percentiles of Xbar")
lines(c(-1.5,0.48),c(0.3,0.3),lty=2)
lines(c(mus[round(InteTable[-1,5],3)==0.3],mus[round(InteTable[-
1,5],3)==0.3]),c(0.3,-2.9),lty=2)
lines(c(mus[round(InteTable[-1,6],3)==0.3],mus[round(InteTable[-
1,6],3)==0.3]),c(0.3,-2.9),lty=2)
lines(c(mus[round(InteTable[-1,7],3)==0.3],mus[round(InteTable[-
1,7],3)==0.3]),c(0.3,-2.9),lty=2)


InteTable[-1,12]->thib
thib<-thib[numb]
InteTable[-1,11]->tmid
tmid<-tmid[numb]
InteTable[-1,10]->tlob
tlob<-tlob[numb]

plot(muf1,thib,type='l',lty=1,xlim=c(-1.1,1.1),ylim=c(-
9,9),xlab='',ylab='')
par(new=T)
plot(muf1,tmid,type='l',lty=1,xlim=c(-1.1,1.1),ylim=c(-
9,9),xlab='',ylab='')
par(new=T)
plot(muf1,tlob,type='l',lty=1,xlim=c(-1.1,1.1),ylim=c(-9,9),xlab="True
Mean",ylab="Percentiles of T")
title(sub="Fig.4 Integrated percentiles of T")

lines(c(-2.5,0.726),c(3.390573,3.390573),lty=2)
lines(c(0.726,0.726),c(3.390573,-17),lty=2)
lines(c(0.3,0.3),c(3.390573,-17),lty=2)
lines(c(0.123,0.123),c(3.390573,-17),lty=2)

##### the power
# show the power by Tstat & Xbar using numerical integration
par(mfrow=c(2,1))
mus<-seq(-2,2,0.001)
pwrx<-InteTable[-1,3]
plot(mus[seq(1281,2720)],pwrx[seq(1281,2720)],type='l',lty=2,xlab="The
true mean",ylab="Power of Xbar & Tstat integration")
title(sub="Fig.5 power using Xbar & Tstat via integration")
par(new=T)
```

```
pwrt<-InteTable[-1,8]
plot(mus[seq(1281,2720)],pwrt[seq(1281,2720)],type='l',lty=1,xlab="",yl
ab="")
leg<-c("T power","Xbar power")
legend(0.3,0.3,legend=leg, lty=1:2)

# Compare the power by Tstat & Xbar via integration.
## The alteration of Tpower & Xpower as a function of the true mu.

# Absolute difference in integrated power (Xbar - Tstat) versus true
mean

plot(InteTable[seq(1001,3001),1],InteTable[seq(1001,3001),8]-
InteTable[seq(1001,3001),3],type="l",xlab='True mean',ylab='Power by
Tstat-Power by Xbar')
title(sub='Fig.6 Tstat power-Xbar power by integration')
lines(c(mus[round(mus,3)==-1.2],mus[round(mus,3)==1.2]),c(0,0),lty=2)



#### Now the efficiency.
par(mfrow=c(1,1))
mu<-seq(-0.7,0.7,0.01)
plot(mu,effiIn.xt33,type='l',ylim=c(0,1),xlab='True
Mean',ylab='Efficiency')
title(sub="Fig. 7 Efficiency of Xbar & Tstat by Integration")
lines(c(-0.7,0.7),c(1,1),lty=2)


#### now the bias
# use the code file BiasAnal + variation comparison?
# need to find out the Integration bias results.
bias.xb<-abs(bias.xt33[,2]-bias.xt33[,1])
bias.xm<-abs(bias.xt33[,3]-bias.xt33[,1])
bias.tb<-abs(bias.xt33[,4]-bias.xt33[,1])
bias.tm<-abs(bias.xt33[,5]-bias.xt33[,1])

bias.bam<-bias.xb-bias.tb
bias.mue<-bias.xm-bias.tm


# Now the integrated bias
#mu<-seq(-0.7,0.7,0.01)
#Ibias.xb<-abs(Ixbam-mu)
#Ibias.xm<-abs(Ixmue-mu)
#Ibias.tb<-abs(Itbam-mu)
#Ibias.tm<-abs(Itmue-mu)

#Ibias.bam<-Ibias.xb-Ibias.tb
#Ibias.mue<-Ibias.xm-Ibias.tm

par(mfrow=c(2,1))

plot(mu,smooth(Ibias.xm),xlab='',ylab='',type="l",lty=2,xlim=c(-
0.7,0.7),ylim=c(-0.005,0.07))
```

```
par(new=T)
plot(mu,smooth(Ibias.tm),xlab='True Mean',ylab='bias by X.mue &
T.mue',type="l",lty=1,xlim=c(-0.7,0.7),ylim=c(-0.005,0.07))
title(sub="Fig.8 Integrated bias of Xbar.mue & Tstat.mue")
lines(c(-0.72,0.72),c(0,0))

plot(mu,smooth(Ibias.tm-Ibias.xm),xlab='True Mean',ylab='Difference in
bias (Bias by T.mue-Bias by X.mue)',type='l')
title(sub="Fig.9 Difference between bias caused by Xbar.mue and
Tstat.mue")
lines(c(-2.1,2.1),c(0,0),lty=2)

plot(mu,smooth(Ibias.xb),xlab='',ylab='',type="l",lty=2,xlim=c(-
0.7,0.7),ylim=c(-0.005,0.07))
par(new=T)
plot(mu,smooth(Ibias.tb),xlab='True Mean',ylab='bias by X.bam &
T.bam',type="l",lty=1,xlim=c(-0.7,0.7),ylim=c(-0.005,0.07))
title(sub="Fig.10 Integrated bias of Xbar.bam & Tstat.bam")
lines(c(-0.72,0.72),c(0,0))
plot(mu,smooth(Ibias.tb-Ibias.xb),xlab='True Mean',ylab='Difference in
bias (Bias by T.bam-Bias by X.bam)',type='l')
title(sub="Fig.11 Difference between bias caused by Xbar.bam and
Tstat.bam")
lines(c(-2.1,2.1),c(0,0),lty=2)



#### comparison of the parameters obtained via simulation & numerical
integration: E(Xbar)-mean(Xbar), E(t)-mean(t), lot-.....

par(mfrow=c(2,2))
plot(InteTable[-1,4],Table33[-1,4],type='l',xlab='Integrated
E(x)',ylab='Simulated mean(x)')
title(sub='Fig.10 Simulated and integrated E(x)')
abline(0,1)


plot(InteTable[-1,6],Table33[-1,6],type='l',xlab='Integrated
median(x)',ylab='Simulated median(x)')
title(sub='Fig.11 Simulated and integrated median of Xbar')
abline(0,1)


plot(InteTable[-1,5],Table33[-1,5],type='l',xlab='Integrated
2.5%(x)',ylab='Simulated 2.5%(x)')
title(sub='Fig.12 Simulated and integrated 2.5 percentile of Xbar')
abline(0,1)

plot(InteTable[-1,12],Table33[-1,12],type='l',xlab='Integrated
E(x)',ylab='Simulated mean(x)')
title(sub='Fig.13 Simulated and integrated 97.5 percentile of Xbar')
abline(0,1)
```

```
plot(InteTable[-1,9],Table33[-1,9],type='l',xlab='Integrated
E(T)',ylab='Simulated mean(T)')
title(sub='Fig.14 Simulated and integrated E(T)')
abline(0,1)


plot(InteTable[-1,11],Table33[-1,11],type='l',xlab='Integrated
median(T)',ylab='Simulated median(T)')
title(sub='Fig.15 Simulated and integrated median of T')
abline(0,1)


plot(InteTable[-1,10],Table33[-1,10],type='l',xlab='Integrated
2.5%(T)',ylab='Simulated 2.5%(T)')
title(sub='Fig.16 Simulated and integrated 2.5 percentile of T')
abline(0,1)

plot(InteTable[-1,12],Table33[-1,12],type='l',xlab='Integrated
97.5%(T)',ylab='Simulated 97.5%(T)')
title(sub='Fig.17 Simulated and integrated 97.5 percentile of T')
abline(0,1)
```