5-1-2012

Sharpening the Boundaries of the Sequential Probability Ratio Test

Elizabeth Krantz

SHARPENING THE BOUNDARIES OF THE SEQUENTIAL PROBABILITY RATIO TEST

A Thesis
Presented to
The Faculty of the Department of Mathematics and Computer Science
Western Kentucky University
Bowling Green, Kentucky

In Partial Fulfillment
Of the Requirements for the Degree
Master of Science

By
Elizabeth Krantz

May 2012

SHARPENING THE BOUNDARIES OF THE SEQUENTIAL PROBABILITY RATIO TEST

Date Recommended 4/25/2012

Melanie A. Autin

Dr. Melanie Autin, Director of Thesis

Darwin C. Neal

Dr. David Neal

Dr. Ngoc Nguyen

Kinchel C Doerner 25-May-2012

Dean, Graduate Studies and Research Date

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude and appreciation to Dr. Melanie Autin for guiding me in this process. If it were not for her, I would have never pursued an interest in statistics and completed this thesis. She has invested countless hours assisting me in every way, and none of this would have happened without her. I am truly grateful for her as not only as an advisor but also as a mentor.

Also, I would like to express my gratitude to Dr. David Neal and Dr. Ngoc Nguyen for giving of their valuable time to read and give corrections to this thesis. Additionally, I would like to acknowledge Dr. Don Edwards for sharing in his intellectual property.

I must also not forget the people who urged me to pursue my dreams. I am so thankful for Cory as he has stood beside me and provided moral support even when he has been unsure of what he was supporting. Also, I am thankful for my Mom, Dad, Jammie, and Jimmy for their unending supply of encouragement, support, and faith in me.

TABLE OF CONTENTS

List of Figures v

List of Tables vi

Abstract vii

Chapter 1: Introduction 1

Chapter 2: Sequential Probability Ratio Test 3
2.1 Wald's Sequential Probability Ratio Test 3
2.2 Mingoti's Expansion of SPRT 9
2.3 Iglewicz's Expansion of SPRT 9
2.4 McWilliams' Expansion of SPRT 10
2.5 Ignatova, Deutsch, and Edwards' Expansion of SPRT 11

Chapter 3: A Method for Sharpening the Boundaries 18
3.1 Creating the Bounds 18
3.2 Results for the New Bounds 24

Chapter 4: Simulation Studies 34
4.1 Simulation Method 34
4.2 Simulation Results 35

Chapter 5: Conclusion and Future Work 47

Appendix 49
A.1 Boundaries for $n_T = 4$ 49
A.2 R Code 60
A.2.1 `check.step()` 60
A.2.2 `create.bounds()` 61
A.2.3 `alpha.bounded()` 64
A.2.4 `errors.bounded()` 66
A.2.5 `sim.sprt()` 69
A.2.6 `Wald.bnd()` 73
A.2.7 `Wald.step.bnd()` 74

References 75

LIST OF FIGURES

Figure 2.1 Full Pascal's Triangle Example 12

Figure 2.2 Modified Pascal's Triangle Method 13

Figure 2.3 Wald Boundary Example 17

Figure 3.1 Standard Hypothesis Test 19

Figure 4.1 Binomial $n_T = 10$ Simulation with $\pi_0 = 0.1$ and $\pi_1 = 0.7$ 42

Figure 4.2 Binomial $n_T = 10$ Simulation with $\pi_0 = 0.1$ and $\pi_1 = 0.9$ 43

Figure 4.3 Binomial $n_T = 10$ Simulation with $\pi_0 = 0.2$ and $\pi_1 = 0.8$ 44

Figure 4.4 Binomial $n_T = 10$ Simulation with $\pi_0 = 0.2$ and $\pi_1 = 0.9$ 45

Figure 4.5 Binomial $n_T = 10$ Simulation with $\pi_0 = 0.3$ and $\pi_1 = 0.9$ 46

LIST OF TABLES

Table 3.1 Possible Single Boundaries for $n_T = 4$ 21

Table 3.2 Total Number of Potential Boundaries for n_T 24

Table 3.3 Binomial Results for $n_T = 10$ 26

Table 3.4 Binomial Results for $n_T = 20$ 27

Table 3.5 Binomial Results for $n_T = 30$ 28

Table 3.6 Binomial Results for $n_T = 40$ 29

Table 3.7 Hypergeometric Results for $n_T = 10$ 30

Table 3.8 Hypergeometric Results for $n_T = 20$ 30

Table 3.9 Hypergeometric Results for $n_T = 30$ 31

Table 3.10 Hypergeometric Results for $n_T = 40$ 31

Table 4.1 Hypergeometric Simulation Results for $n_T = 10$ 36

Table 4.2 Hypergeometric Simulation Results for $n_T = 20$ 36

Table 4.3 Hypergeometric Simulation Results for $n_T = 30$ 37

Table 4.4 Hypergeometric Simulation Results for $n_T = 40$ 37

Table 4.5 Binomial Simulation Results for $n_T = 10$ 38

Table 4.6 Binomial Simulation Results for $n_T = 20$ 39

Table 4.7 Binomial Simulation Results for $n_T = 30$ 39

Table 4.8 Binomial Simulation Results for $n_T = 40$ 40

SHARPENING THE BOUNDARIES OF THE SEQUENTIAL PROBABILITY RATIO TEST

Elizabeth Krantz May 2012 75 Pages

Directed by: Dr. Melanie Autin, Dr. David Neal, and Dr. Ngoc Nguyen

Department of Mathematics & Computer Science Western Kentucky University

In this thesis, we present an introduction to Wald's Sequential Probability Ratio Test (SPRT) for binary outcomes. Previous researchers have investigated ways to modify the stopping boundaries that reduce the expected sample size for the test. In this research, we investigate ways to further improve these boundaries. For a given maximum allowable sample size, we develop a method intended to generate all possible sets of boundaries. We then find the one set of boundaries that minimizes the maximum expected sample size while still preserving the nominal error rates. Once the satisfying boundaries have been created, we present the results of simulation studies conducted on these boundaries as a means for analyzing both the expected number of observations and the amount of variability in the sample size required to make a decision in the test.

Chapter 1: Introduction

In statistical analysis we are concerned with data-driven decision making, and one avenue that we pursue to test conjectures is hypothesis testing. In hypothesis testing, we test H_0, the null hypothesis, versus H_1, the alternative hypothesis. In a standard hypothesis test, we take a sample of size n, and make a decision based on the entire collection of data. The sequential probability ratio test (SPRT) instead requires that each item be sampled one at a time. After each item, the researcher either makes a decision in the hypothesis test or decides to continue sampling by selecting another item. Thus, the SPRT can result in a significantly smaller sample size than a standard hypothesis test.

In practice, SPRT tends to be more efficient than standard hypothesis testing and is more cost effective in industry. For example, SPRT has been used in detecting Medicare fraud; instead of examining every case that a particular doctor turns in for Medicare to pay, the analyst looks at one case at a time and either makes a decision that the case is or is not fraudulent. From that, the investigator may be able to make a decision about the doctor without looking at all of his cases. Thus, money and time are possibly saved because the testing will likely terminate before all cases have been examined. Additionally, in quality control for industry, instead of examining an entire production line of goods, a company could find it more economical to look at one product at a time and determine whether or not the item is defective and, in turn, make a decision about the entire product line before sampling the entire group of products. Thus,

companies can increase profit by spending less time and money on testing for defects in production. They can also more quickly identify and correct production problems.

For the purpose of this thesis, we are interested in formulating a possible method that can reduce the expected number of observations necessary to make a decision in the sequential probability ratio test. Some research in this area has been conducted, but we set out to build upon and improve the methods that already exist.

In Chapter 2, we first look at the original SPRT developed by Wald and then continue by looking at expansions on SPRT by other researchers. Next, Chapter 3 will detail the process of creating new sets of boundaries as an exploration in sharpening the boundaries of the SPRT. In Chapter 4, we provide a description of simulation studies that were conducted as a way to analyze the new sets of boundaries. Finally, we summarize the research in Chapter 5 and examine some improvements that could be made in the future.

Chapter 2: Sequential Probability Ratio Test

2.1 Wald's Sequential Probability Ratio Test

In 1947, Abraham Wald published a book entitled *Sequential Analysis* [6], in which he details the process for conducting the sequential probability ratio test for quality control. We define a random variable X on the items being inspected by letting $X = 1$ if the item observed is found to be defective and $X = 0$ if the item observed is non-defective. Define $f(x,\pi)$ to be the probability distribution function of the random variable X where π is the proportion of defective items in a (possibly infinite) collection of size N. When considering binary outcomes (defective vs. non-defective), the first n successive observations of x will be designated as $x_1, x_2, \ldots, x_n$, where x_i indicates whether the item is found to be defective or non-defective. It is of interest to test two competing values of π. We let H_0 be the hypothesis that $\pi = \pi_0$ and H_1 be the hypothesis that $\pi = \pi_1$, where $\pi_0 < \pi_1$. Thus, $f(x,\pi_0)$ is the distribution function of X when the null hypothesis is true, and $f(x,\pi_1)$ is the distribution function of X when the alternative hypothesis is true. We let $y_n = \sum_{i=1}^{n} x_i$ be the total number of defective items in the n sampled items. For any positive value n, the probability that a sample $x_1, \ldots, x_n$ is observed is determined by the likelihood functions $L(y_n, \pi_0) = \prod_{i=1}^{n} f(x_i, \pi_0)$ when H_0 is true and $L(y_n, \pi_1) = \prod_{i=1}^{n} f(x_i, \pi_1)$ when H_1 is true. The sequential probability ratio test (SPRT) for testing H_0 against H_1 is determined by the following process.

Two positive constants A and B are chosen such that $B < A$. At each stage of the test (n^{th} trial), compute $\Lambda_n = \frac{L(y_n, \pi_1)}{L(y_n, \pi_0)}$. If $B \leq \Lambda_n \leq A$, continue the test by taking another observation. If $A < \Lambda_n$, the test is ended, and the null hypothesis is rejected. If $\Lambda_n < B$, the test is ended, and we fail to reject the null hypothesis.

For practicality in computation, $\ln(\Lambda_n)$ is often used over Λ_n since $\ln(\Lambda_n)$ can be written as the sum of n terms, i.e $\ln(\Lambda_n) = \ln\left(\frac{f(x_1, \pi_1)}{f(x_1, \pi_0)}\right) + \ldots + \ln\left(\frac{f(x_n, \pi_1)}{f(x_n, \pi_0)}\right)$. The i^{th} term of the sum is denoted as z_i. Thus, at each stage of the test (n^{th} trial) the cumulative sum $\sum_{i=1}^{n} z_i$ is computed. If $\ln(B) \leq \sum_{i=1}^{n} z_i \leq \ln(A)$, continue the test by taking more observations. If $\ln(A) < \sum_{i=1}^{n} z_i$, the test is ended, and H_0 is rejected. If $\sum_{i=1}^{n} z_i < \ln(B)$, the test is ended, and H_0 is not rejected.

Assuming that the test has not terminated prior to a given step n, we consider a sample (x_1, … , x_n). We say that the sample is of type 1 if $A < \Lambda_n$, which leads to rejection of H_0 at step n. We say that the sample is of type 2 if $\Lambda_n < B$, which leads to failing to reject H_0 at step n.

For any type 1 sample, the probability of achieving such a sample under H_1 is at least A times as large as the probability of achieving such a sample under H_0, as seen by $L(y_n, \pi_1) \geq A \cdot L(y_n, \pi_0)$. Therefore, the probability that the sequential test will terminate with rejection of H_0 is also at least A times as large under H_1 as it is under H_0. When H_0 is true, the probability that the test will

terminate with rejection of H_0 is α. This incorrect decision is a Type I error, where α is the Type I error rate. When H_1 is true, the probability that the test will correctly terminate with rejection of H_0 is denoted as $1-\beta$. Thus we obtain,

$$\begin{aligned} L(y_n, \pi_1) &\geq A \cdot L(y_n, \pi_0) \\ 1-\beta &\geq A\alpha \\ A &\leq \frac{1-\beta}{\alpha}, \end{aligned} \tag{2.1}$$

which determines an upper limit for A to be $\frac{1-\beta}{\alpha}$.

In a similar manner, a lower limit for B can be derived. For any given type 2 sample (x_1, … , x_n), the probability of achieving such a sample under H_1 is at most B times as large as the probability of achieving such a sample under H_0, $L(y_n, \pi_1) \leq B \cdot L(y_n, \pi_0)$. Therefore, the probability that the sequential test will terminate with failing to reject H_0 is also at most B times as large under H_1 as it is under H_0. When H_0 is true, the probability that the test will correctly terminate without rejection of H_0 is denoted $1-\alpha$. When H_1 is true, the probability that the test will incorrectly terminate with failure to reject H_0 is β. This incorrect decision is referred to as a Type II error, where β is the Type II error rate. Thus we obtain,

$$\begin{aligned} L(y_n, \pi_1) &\leq B \cdot L(y_n, \pi_0) \\ \beta &\leq B(1-\alpha) \\ B &\geq \frac{\beta}{1-\alpha}, \end{aligned} \tag{2.2}$$

which determines a lower limit for B to be $\frac{\beta}{1-\alpha}$,. Consequently, upper limits are obtained for α and β based on the above inequalities:

$$\alpha \le \frac{1}{A} \text{ and } \beta \le B. \tag{2.3}$$

We must also now consider the opposite approach in which we are not given the values of *A* and *B*. Previously, we chose a given *A* and *B* and determined the Type I and Type II error rates. In practice, testing will be conducted under predetermined maximum allowable error rates (α and β), and *A* and *B* must instead be determined. We would like to test with given strength (α, β); so, denote $A(\alpha, \beta)$ and $B(\alpha, \beta)$ as the values of *A* and *B* which satisfy the predetermined (α, β). From the previously mentioned inequalities, (2.1) implies $A(\alpha,\beta) \le \frac{1-\beta}{\alpha}$, and (2.2) implies $B(\alpha,\beta) \ge \frac{\beta}{1-\alpha}$.

Wald [6] proposed to set $A = \frac{1-\beta}{\alpha} = a(\alpha,\beta)$, which is in fact greater than or equal to the exact value of $A(\alpha, \beta)$, and $B = \frac{\beta}{1-\alpha} = b(\alpha,\beta)$ which is less than or equal to the exact value of $B(\alpha, \beta)$. Wald discusses the consequences that the assigned combinations of *A* and *B* have on the error rates α and β. When *A* is a value greater than or equal to $A(\alpha, \beta)$ and *B* is equal to $B(\alpha, \beta)$, the Type I error rate is less than the nominal α, but the Type II error rate is greater than β. If *A* is equal to $A(\alpha, \beta)$ and *B* is a value less than or equal to $B(\alpha, \beta)$, then the Type II error rate is less than the nominal β, but the Type I error rate is greater than α. If *A* is a value greater than $A(\alpha, \beta)$ and *B* is a value less than $B(\alpha, \beta)$, then the effect on the Type I and Type II error rates is unclear. Wald thus proposes to denote α' and β' as the respective error rates when $A=a(\alpha, \beta)$ and $B=b(\alpha, \beta)$.

From above, $\frac{\alpha'}{1-\beta'} \le \frac{1}{a(\alpha,\beta)} = \frac{\alpha}{1-\beta}$ and $\frac{\beta'}{1-\alpha'} \le \frac{1}{b(\alpha,\beta)} = \frac{\beta}{1-\alpha}$. Thus, upper limits for α' and β' can be found to be $\alpha' \le \frac{\alpha}{1-\beta}$ and $\beta' \le \frac{\beta}{1-\alpha}$. Through some arithmetic, it can be shown that $\alpha'+\beta' \le \alpha+\beta$. Thus, we set $A = \frac{(1-\beta)}{\alpha}$ and $B = \frac{\beta}{(1-\alpha)}$.

We now consider the case of sampling with replacement (or sampling without replacement from a population of infinite size). Since the probability of selecting a defective item is π for each item and observations are independent, each observation can be considered a Bernoulli trial. Thus, Wald's boundaries can be found as follows. In the likelihood ratio Λ_n, the binomial density can be substituted for the likelihood functions, giving

$$\Lambda_n = \frac{L(y_n,\pi_1)}{L(y_n,\pi_0)} = \frac{\binom{n}{y_n}\pi_1^{y_n}(1-\pi_1)^{n-y_n}}{\binom{n}{y_n}\pi_0^{y_n}(1-\pi_0)^{n-y_n}}. \quad . \tag{2.4}$$

Using the criterion $B \le \Lambda_n \le A$ presented by Wald,

$$B \le \frac{L(y_n,\pi_1)}{L(y_n,\pi_0)} = \frac{\binom{n}{y_n}\pi_1^{y_n}(1-\pi_1)^{n-y_n}}{\binom{n}{y_n}\pi_0^{y_n}(1-\pi_0)^{n-y_n}} \le A. \tag{2.5}$$

Notice that instead of computing this ratio at each observation, we can solve for y_n as follows:

$$\ln(B) \le \ln\left[\frac{\binom{n}{y_n}\pi_1^{\,y_n}(1-\pi_1)^{n-y_n}}{\binom{n}{y_n}\pi_0^{\,y_n}(1-\pi_0)^{n-y_n}}\right] \le \ln(A)$$

$$\ln(B) \le y_n \ln\left[\frac{\pi_1(1-\pi_0)}{\pi_0(1-\pi_1)}\right] + n\ln\left[\frac{(1-\pi_1)}{(1-\pi_0)}\right] \le \ln(A)$$

$$\frac{\ln(B) - n\ln\left[\frac{(1-\pi_1)}{(1-\pi_0)}\right]}{\ln\left[\frac{\pi_1(1-\pi_0)}{\pi_0(1-\pi_1)}\right]} \le y_n \le \frac{\ln(A) - n\ln\left[\frac{(1-\pi_1)}{(1-\pi_0)}\right]}{\ln\left[\frac{\pi_1(1-\pi_0)}{\pi_0(1-\pi_1)}\right]}. \qquad (2.6)$$

Notice that y_n is now bounded above and below by parallel lines. Thus, if the total number of defectives at the current position n is between or on these lines, y_n is said to fall in the "zone of indifference". Although values of y_n may occur here, sampling continues within this region, and we therefore do not refer to these results as possible outcomes of the test. We refer to y_n values that are above the upper boundary line as being in the "rejection region," for these are the outcomes that would lead to rejection of the null hypothesis. Though we never actually accept H_0 when values of y_n are below the lower bound, instead of referring to the region as the "zone of failing to reject," for simplicity we call it the "acceptance region;" values of y_n in this region are the outcomes that would lead to failing to reject the null hypothesis.

Wald [6] has shown that the sequential probability ratio test will eventually end in a decision being made with a probability of 1, but in practice, the sequential test is truncated at a maximum allowable sample size, n_T. If a decision has not been made prior to reaching the n_T observation, sampling stops

with the n_T observation. Then a decision is made based on the value of y_{n_T}. We say that sequential testing is a closed procedure because testing is limited to a maximum allowable size.

Research on SPRT has continued since its first introduction by Wald, and selected examples are presented in the following sections of this chapter.

2.2 Mingoti's Expansion of SPRT

Researchers have also been interested in determining the effects that the choices of the hypothesized values, π_0 and π_1, have on the required sample size. Mingoti [3] discusses the sample size needed to perform Wald's sequential probability ratio test when items are generated by a process for which the results of the inspections are correlated. It was shown that for values of π_0 and π_1 which are close together, the sample size is larger than that which are found to be necessary when using values of π_0 and π_1 that are more distant. Additionally, when considering correlation (ρ) between the results of the *n* observations with respect to a binary response variable, the sample size increases as the value of the correlation coefficient ρ increases. In fact, when comparing the corresponding expected sample size for when $\rho = 0$ to when $0.5 < \rho < 0.7$, the expected sample size is three times higher; it is six times higher when $\rho > 0.7$.

2.3 Iglewicz's Expansion of SPRT

It is most common for sequential procedures to be compared in terms of the expected sample size, which is also known as the average sample number

(ASN). This type of comparison does not consider sample size variability and is often biased in favor of tests with large sample-size variability. In terms of practicality, large sample-size variability is costly because it often leads to a large number of observations being required before the process terminates. Thus, Iglewicz [1] proposes an alternative measure for the ASN, $D(\pi) = \frac{N - E(n \mid \pi)}{\sigma_n(\pi)}$, where N is the number of required observations for a fixed sample size test, π is the true value of the parameter, $E(n|\pi)$ is the average sample number, and $\sigma_n(\pi)$ is sample-size standard deviation. $D(\pi)$ is a measure of the gain obtained by using a sequential procedure in place of a fixed sample size procedure. When $D(\pi)$ is large, the sequential test is efficient, but when $D(\pi)$ is small, the design is not.

2.4 McWilliams' Expansion of SPRT

Due to the lack of published research regarding truncation on the SPRT, McWilliams [4] presents two case studies that inspect the influence that the choice of truncation parameters have on test performance. He examines both the choice of the truncation sample size and the choice of the truncation rejection value R. When the test reaches the truncation sample size, R identifies how many defectives are required for rejection. Through the explanation of the two case studies, both choices appear have a deep impact on test performance. McWilliams compares the ASN under both the null hypothesis, $\pi = \pi_0$, and the alternative hypothesis, $\pi = \pi_1$, since the value of the ASN will differ depending

upon the true value of π in the population. Additionally, the author examines the amount of variability in required sample size and the value of the error rates. McWilliams concludes that adopting an aggressive truncation strategy results in a consequences in terms of the error probabilities and the ASN. The author does not generalize his results but expresses that, from his findings, the performance of the test over various values of R needs to be considered instead of an arbitrary choice of the rejection value.

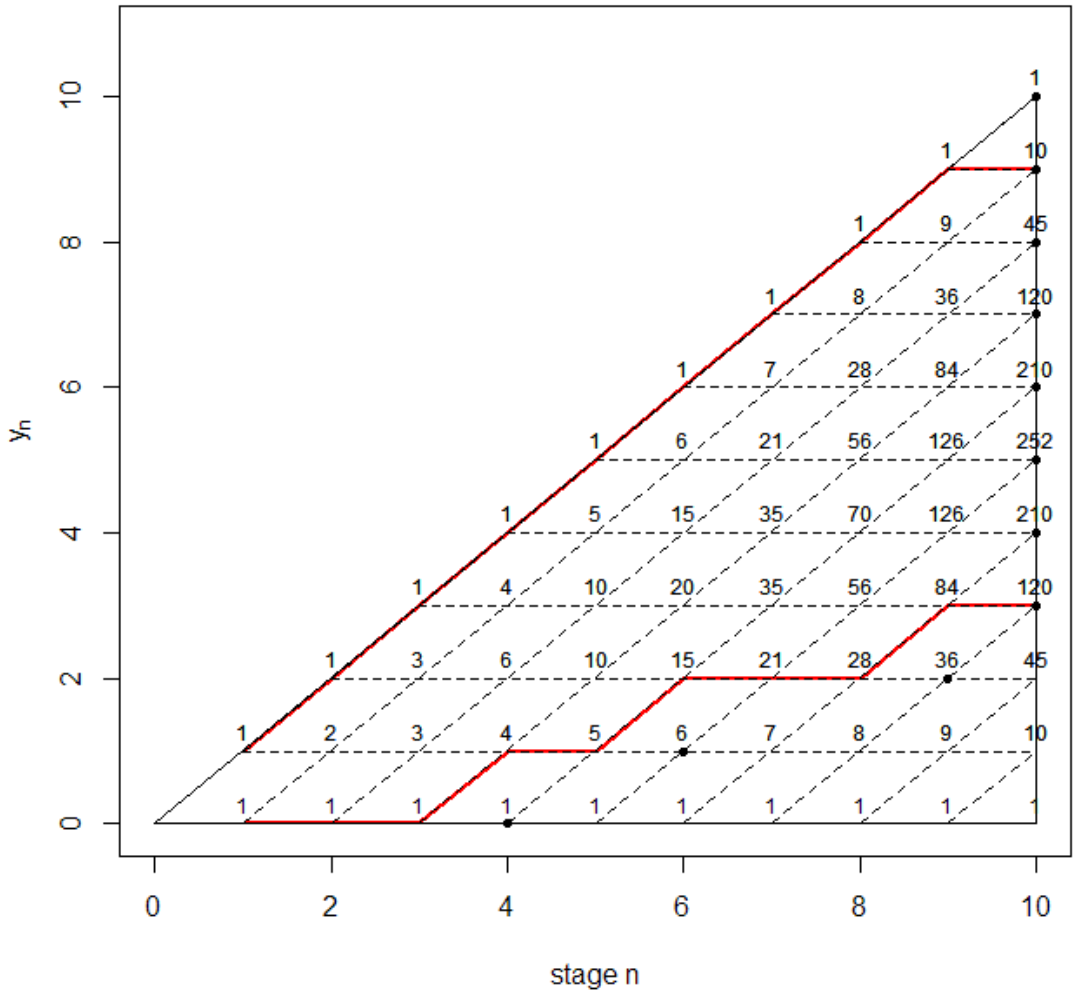
2.5 Ignatova, Deutsch, and Edwards' Expansion of SPRT

While considering closed sequential or multistage sampling from a lot of N items, the goal is to make conclusions about the proportion of defectives, π, in the sample. Instead of using asymptotic results, Ignatova, et. al. [2] show that exact inference about the proportion of defectives can be calculated using current statistical software.

While considering y_n to be the number of defectives at stage n, a decision is made whether to continue sampling or to end testing by comparing y_n to a set of boundaries. Since, in practice, sequential testing is truncated at n_T, the truncated SPRT is the sole focus for the authors. For a given outcome, y_n, there are $\binom{n}{y_n}$ sequences of y_n defectives and $(n - y_n)$ non-defectives that result in a total of y_n defectives in n trials. As an example, consider the case of $n_T = 10$, shown in Figure 2.1. Each number represented in Figure 2.1 represents the number of ways to get y_n (on the y-axis) defectives in n trials (on the x-axis); we

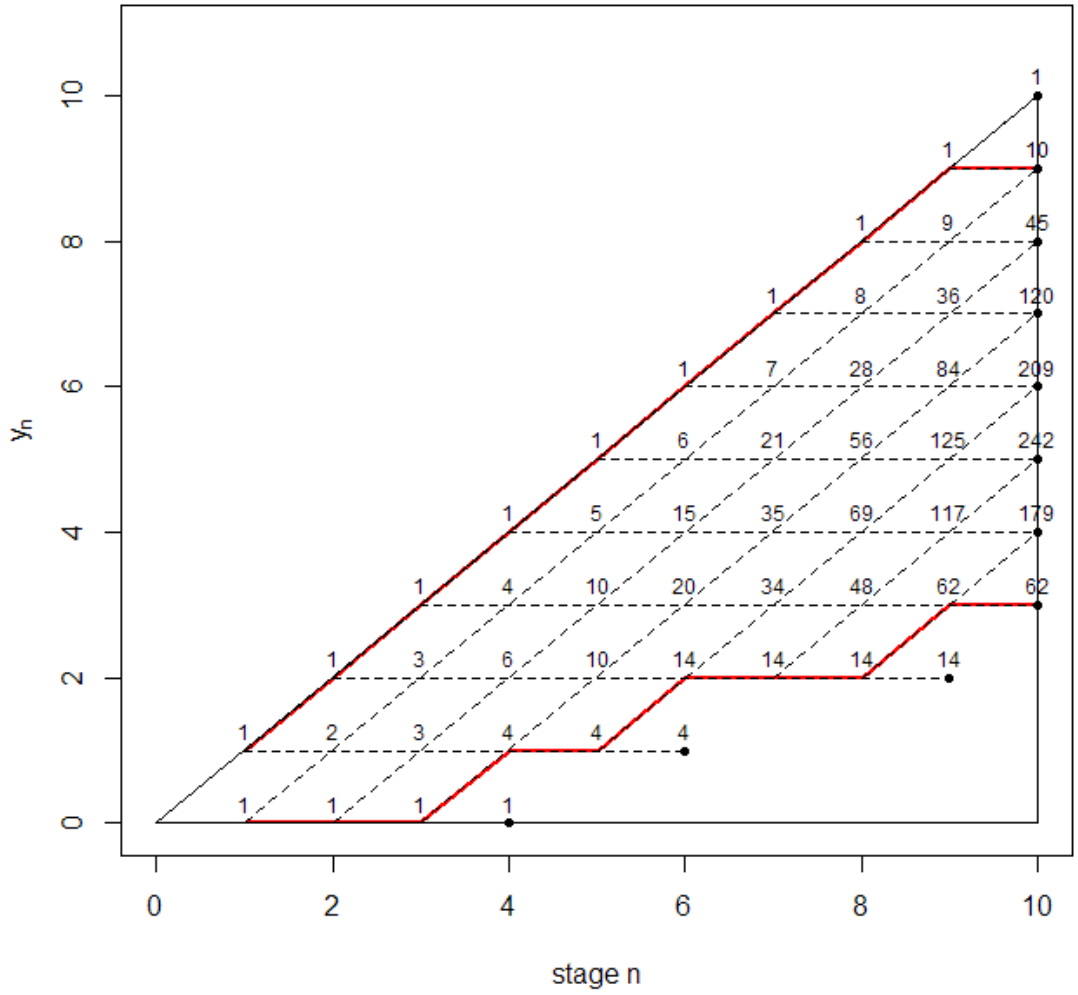
refer to this as a path-count. Thus, we can consider the number of "paths" to a specific point (n, y_n) on the graph. Arbitrarily chosen upper and lower boundaries are shown as a reference, but no modifications in the path counts have been made as a result of outcomes falling outside of the boundaries. The unrestricted path-counts are found via Pascal's triangle.

Figure 2.1: Full Pascal's Triangle Example



The boundaries must now be taken into account to determine the actual number of paths to y_n. An observation occurring outside of the boundaries results in sampling to end; thus, the paths extending from that point no longer exist. Based on a modified Pascal's triangle algorithm, [2] utilizes a path-count method to determine the various paths that lead to specific termination points, while staying within the boundaries. A visual example of the path-count method can be seen in Figure 2.2.

Figure 2.2: Modified Pascal's Triangle Method



Note that the solid dots that appear in Figure 2.2 are outcomes for which a decision is made and sampling ends. There are indeed values of y_n which exist between the lower and upper bounds, but no decision is made in the test and sampling continues. Notice that points (4,0), (6,1), (9,2), and (10,10) all occur outside of the upper and lower boundaries shown. Thus, paths extending from (4,0), (6,1), and (9,2) do not exist, and the path-count for successive points are altered. For example, the path-count to point (5,1) reduced from five in the Pascal's triangle example (Figure 2.1) to four in the modified Pascal's triangle method; the path-count to point (7,2) reduced from 21 to 14. Additionally, notice that some of the paths to the outcomes on the truncation boundary are also reduced. For example points (10,3) and (10,5) reduced from 120 to 62 and 252 to 242, respectively. Also, notice that not all path-counts were altered.

Ignatova, et. al. consider this method for both the binomial and hypergeometric cases. They have crossed into new territory by examining the hypergeometric case, as no other authors have pursued it [2]. Now consider the case of testing without replacement from a population of a finite size N. Thus, Wald's boundaries can be determined where y_n is a hypergeometric random variable. Similar to what was done in the binomial case in (2.4), the hypergeometric density can be substituted for the likelihood functions, giving

$$\Lambda_n = \frac{L(y_n, D_1)}{L(y_n, D_0)} = \frac{\dfrac{\binom{D_1}{y_n}\binom{N-D_1}{n-y_n}}{\binom{N}{n}}}{\dfrac{\binom{D_0}{y_n}\binom{N-D_0}{n-y_n}}{\binom{N}{n}}} = \frac{\binom{D_1}{y_n}\binom{N-D_1}{n-y_n}}{\binom{D_0}{y_n}\binom{N-D_0}{n-y_n}} \tag{2.7}$$

.

where D_0 is the number of defective items in the population under H_0 and D_1 is the number of defectives in the population under H_1.

Using the criterion $B \le \Lambda_n \le A$ presented by Wald, we have

$$B < \frac{\binom{D_1}{y_n}\binom{N-D_1}{n-y_n}}{\binom{D_0}{y_n}\binom{N-D_0}{n-y_n}} < A. \tag{2.8}$$

In the pursuit of trying to isolate y_n in (2.8), we begin trying to simplify, giving

$$B < \frac{(D_1)!(N-D_1)!}{(D_0)!(N-D_0)!} \cdot \frac{(D_0-y_n)!(N-D_0-n+y_n)!}{(D_1-y_n)!(N-D_1-n+y_n)!} < A$$
$$\frac{B(D_0)!(N-D_0)!}{(D_1)!(N-D_1)!} < \frac{(D_0-y_n)!(N-D_0-n+y_n)!}{(D_1-y_n)!(N-D_1-n+y_n)!} < \frac{A(D_0)!(N-D_0)!}{(D_1)!(N-D_1)!}. \tag{2.9}$$

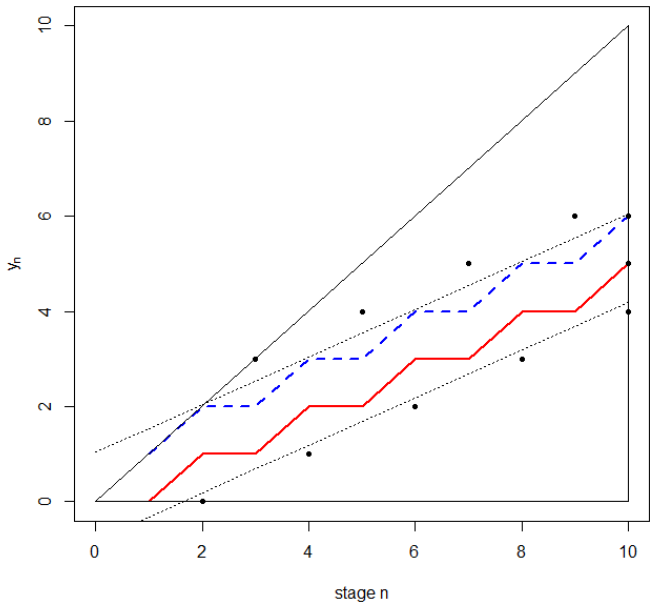
Although it was attempted, isolation of y_n could not be determined at this time.

The authors created a function entitled seqbin, using the statistical software package R [5], to determine sequential acceptance sampling for binary outcomes from a population of size N while sampling with or without replacement. Based on Wald's SPRT method, a truncation position n_T and upper and lower boundaries are given as arguments to the function. The authors determined that Wald's boundaries can be represented as a step boundary

because as n increases by one, the value of y_n stays the same (if that observation is non-defective) or increases by one (if that observation is defective). Sampling continues if y_n is between or on the boundaries and $n < n_T$. Using the boundary-modified Pascal algorithm, the seqbin function detects all possible outcomes (and the number of "paths" possible to reach that outcome) where y_n is less than the lower bound or greater than the upper bound (i.e., where the test does not stop before reaching n_T) or where y_n is on the truncation boundary. Additionally, the seqbin function calculates the number of defectives sampled and calculates the probability distribution of all possible outcomes.

We now consider an example of Wald's truncated SPRT for testing the null hypothesis that $\pi_0 = 0.2$ versus the alternative hypothesis that $\pi_1 = 0.8$, with $n_T = 10$, $\alpha = 0.05$, and $\beta = 0.10$. Using $A = (1-\beta)/\alpha$ and $\beta/(1-\alpha)$ as defined in section 2.1, we have $A = 18$, and $B \cong 0.1053$. From (2.6), the lower boundary line is given by $l_n \cong -0.8119819 + 0.5n$, and the upper boundary line is given by $u_n \cong 1.0424813 + 0.5n$, indicated by the dotted lines in Figure 2.3. The blue dashed line and red line represent step-boundary representations of the Wald upper and lower boundaries, respectively. The points in Figure 2.3 are the possible outcomes; notice that points exist in both the rejection region and acceptance region. The points that occur at $n = n_T = 10$ represent the outcomes that occur when sampling has not ended before reaching the truncation boundary. It is necessary that each of these points be included in either the rejection region or the acceptance region since a decision needs to be made in the test.

Figure 2.3: Wald Boundary Example

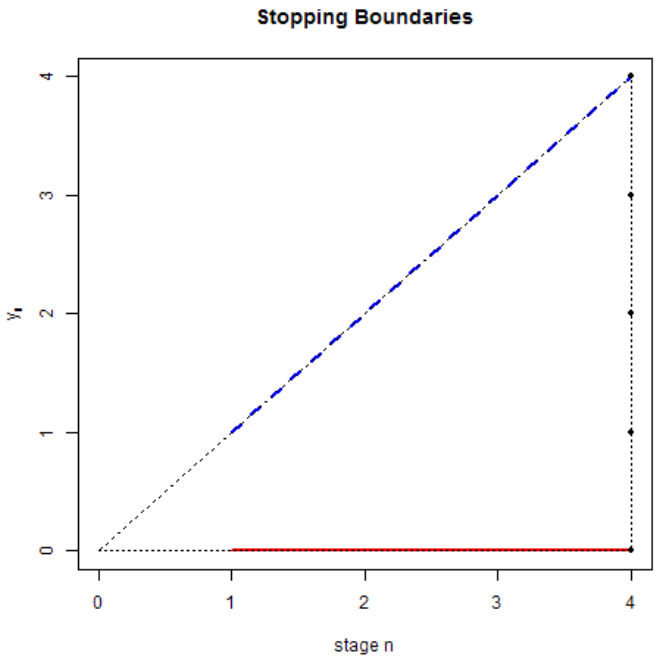


Chapter 3: A Method for Sharpening the Boundaries

3.1 Creating the Bounds

Based on the work previously completed by others, the goal of this thesis is to determine if there are better boundaries available than the ones which have been currently found in [6] and [2]. We would like to consider all possible sets of boundaries that exist for a given n_T. We will then use certain criteria to determine which of these sets of boundaries is the "best." We first consider the set of boundaries that are equivalent to the standard hypothesis test: the lower boundary is the line $y_n = 0$, and the upper boundary is the line $y_n = n$. In other words, the lower bound includes successive observations of all non-defective items and the upper bound includes successive observations of all defective items. Observations here are always in the zone of indifference, so the sample size will always reach the truncation value of n_T. This gives boundaries that form a triangle, as can be seen in Figure 3.1 for the case when $n_T = 4$.

Figure 3.1: Standard Hypothesis Test



Starting with this set of "triangle boundaries," we will find all possible sets of boundaries for a given n_T. We then want to determine all sets of upper and lower boundaries that have both the Type I and Type II error rates bounded. Then we would like to examine under various criteria which set of boundaries is indeed the best.

For a given n_T, we must first generate every possible set of boundaries. When a sample is taken at $n = 1$ (i.e., only the first item is sampled), either zero defective items can be observed ($x_1 = y_1 = 0$) or one defective item is observed ($x_1 = y_1 = 1$); thus, every upper and lower boundary is restricted to beginning at either $y_1 = 0$ or $y_1 = 1$. Next, it is required that the step size is non-decreasing and of at most one because as the number of observations increases by one, the value of the number of defectives (y_n) can only either stay the same or increase

by one. Additionally, the upper and lower boundaries are created such that they can touch one another but may not cross.

To create all sets of possible boundaries, we first create all possible single boundaries. Once this has been done, we combine all possible boundaries such that we have lower and upper boundaries that do not cross. To create all of the single boundaries, we first designate the initial boundary as (0,0,0,…,0), a boundary of n_T zeros (i.e., all n_T sampled items are non-defective). For an index *i* initially set equal to zero, we add 1 to the n_T - *i* position in the boundary, as we keep the bounds that are non-decreasing with a step-size of at most 1. After 1 can no longer be added to the n_T - *i* column due to the step-size requirement, *i* is incremented by one, and the process continues of adding 1 to the next position to the left until the step-size requirement is no longer fulfilled, and *i* is again incremented. The process continues successively until the boundary that represents a sample of n_T defective items, $(1,2,3,\ldots,n_T)$, is obtained. Working backwards, we then designate that the bound begins as $(1,2,3,\ldots,n_T)$ and for an index *i* initially set equal to zero, we subtract 1 from the n_T - *i* position in the boundary as we keep the bounds that are non-decreasing with a step-size of at most 1. After 1 can no longer be subtracted from the n_T - *i* position due to the step-size requirement, *i* is incremented by one, and the process continues of subtracting 1 from the next position to the left until the step-size requirement is no longer fulfilled, and *i* is again incremented. The process continues until the (0,0,0,…,0) boundary is obtained. We then combine all bounds that have been found and delete any duplicate boundaries. Table 3.1 displays the boundaries

generated for $n_T = 4$ via the above described process (before deleting duplicate boundaries).

Table 3.1: Possible Single Boundaries for $n_T = 4$

Building Up	Building Down
0 0 0 0	1 2 3 4
0 0 0 1	1 2 3 3
0 0 1 1	1 2 2 3
0 0 1 2	1 2 2 2
0 1 1 2	1 1 2 2
0 1 2 2	1 1 1 2
0 1 2 3	1 1 1 1
1 1 2 3	0 1 1 1
1 2 2 3	0 0 1 1
1 2 3 3	0 0 0 1
1 2 3 4	0 0 0 0

We then combine this "master" set of boundaries into sets of lower and upper boundaries, keeping only the boundaries which do not cross one another. An R [5] function called create.bnds has been created to implement this procedure (see Appendix A.2.2). An example of all possible sets of boundaries for $n_T = 4$ can be found in Appendix A.1.

Once all of the possible sets of boundaries are created, we are left to determine which set of boundaries is the "best." In practical testing, you will be given a maximum allowance for Type I and Type II error rates. Thus, we must reduce the possible set of boundaries by first determining which sets of boundaries preserve α, the predetermined maximum allowable Type I error rate. To determine if α is preserved, each set of upper and lower n_T boundaries is considered individually. A given set of bounds is first evaluated using the seqbin

function. Using this, we then determine which of the outcomes are in the rejection region, which are in the acceptance region, and which are on the truncation boundary. Then, assuming that the null hypothesis is true ($\pi = \pi_0$), the outcomes in the rejection region, if they exist, are used to calculate α. For each outcome in the rejection region, we consider the probability of that outcome occurring, $P((n_{out}, y_{out}))$, assuming $\pi = \pi_0$ where n_{out} and y_{out} denote the number of items sampled and the number of defectives, respectively, for a given outcome. $P((n_{out}, y_{out}))$ is found by taking the product of the probability mass function and the proportion of paths staying within the boundaries leading to (n_{out}, y_{out}), as shown by [2]. Then, α is calculated by taking the sum of $P((n_{out}, y_{out}))$ for each outcome in the rejection region. If no outcomes appear in the rejection region, $\alpha = 0$ initially. It is important to note that at this stage, the outcomes on the truncation boundary are not yet considered. Once α has initially been calculated for a given set of bounds, we determine if this initial value of α is indeed bounded by the pre-specified value. All sets of boundaries which preserve the given Type I error rate are retained, and the remaining sets of boundaries are eliminated. An R [5] function called alpha.bounded has been written to determine the sets of boundaries for which the Type I error rate is preserved (see Appendix A.2.3).

From the remaining sets of boundaries, we must then determine which of those sets of bounds also maintain β, the predetermined maximum allowable Type II error rate. To determine if β is maintained, each set of upper and lower

α-bounded boundaries for n_T must be considered. First, a given set of α-bounded bounds is evaluated using the seqbin function. Using this, we then determine which of the outcomes are in the rejection region, which are in the acceptance region, and which are on the truncation boundary. Then, assuming that the null hypothesis is true ($\pi = \pi_0$), the outcomes in the rejection region, if they exist, are used to calculate α as explained above. If no outcomes appear in the rejection region, $\alpha = 0$ initially. Now the outcomes that occur on the truncation boundary need to be considered, and a decision in the test must be made. Beginning with the outcome on the truncation point closest to the rejection region, we add that outcome to the rejection region as long as α remains bounded by our desired significance level. Each successive outcome on the truncation point is added to the rejection region until α is beyond our desired bound. Thus, the remaining truncation outcomes are then added to the acceptance region. Assuming that the alternative hypothesis is true ($\pi = \pi_1$), β is calculated using the sum of $P((n_{out}, y_{out}))$ for all of the (n_{out}, y_{out}) outcomes in the acceptance region. All sets of boundaries which preserve the given Type II error rate are retained, and the remaining sets of boundaries are eliminated. An R [5] function called errors.bounded has been written to determine the sets of boundaries for which both the Type I and II error rates are preserved (see Appendix A.2.4.

Once the sets of boundaries that preserve both α and β have been collected, we begin trying to discover which set of boundaries is the best for given values of n_T, π_0, and π_1. One way to define which set of boundaries is the

best is to identify which set of boundaries minimizes the expected number of observations required to make a decision in the hypothesis test. Since the true value of π is unknown, we use a fine grid of 1001 π-values from 0 to 1 in steps of 0.001 to assist in determining the expected number of total observations required to make a decision in the test, denoted as E(n). Using the value of n at each outcome, n_{out}, and the probability of n_{out} over all outcomes, we are able to determine the expected number of observations to be

$$E(n) = \sum_{\text{all out}} n_{\text{out}} \cdot P(n_{\text{out}}). \tag{3.1}$$

Once E(n) is calculated using π equal to each of the 1001 values, we determine the maximum expected number of observations, max{E(n)}, for each set of α- and β-bounded boundaries. We then determine the best set of boundaries to be that which minimizes max{E(n)}.

3.2 Results for the New Bounds

The method described in section 3.1 has been conducted for n_T = 10, 20, 30, and 40 in both the binomial and hypergeometric cases with all combinations of π_0 = 0.1, 0.2, and 0.3 and π_1 = 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, and 0.9. The determined total number of possible sets of lower and upper boundaries (disregarding conservation of the error rates) is shown in Table 3.2.

Table 3.2: Total Number of Potential Boundaries for n_T

n_T	Number of Boundary Sets
10	4,263
20	58,498
30	277,208
40	844,893

In the case of the binomial distribution, for each combination of n_T, π_0, and π_1 the original Wald boundary has also been calculated as a comparison to determine if this new method is indeed an improvement. R [5] functions written to find the Wald boundaries can be found in Appendices A.2.6 and A.2.7. Because the likelihood ratio in the hypergeometric case cannot be simplified to have parallel upper- and lower-boundary lines as in the binomial case, determining a closed form for the Wald upper and lower boundaries has limitations as seen in (2.9). Thus, the Wald boundaries for the hypergeometric case have not been considered at the present time.

In Tables 3.3 through 3.10, the results for the "best" new boundaries appear for both the binomial and hypergeometric distributions. In the binomial case, the corresponding Wald results also appear. For each π_0 and π_1 combination that has been considered, the value of the minimized max{E(n)} is given along with the corresponding values of α and β. The combinations of π_0 and π_1 which appear as a shaded box are not possible due to the condition that $\pi_0 < \pi_1$. The combinations of π_0 and π_1 which are empty indicate that no boundaries where both α and β are bounded were found.

Table 3.3: Binomial Results for $n_T = 10$

π_0			π_1 0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
	0.1	New					5.9184	4.3633	4.3633	3.3677
		α					0.0414	0.0436	0.0436	0.0100
		β					0.0776	0.0905	0.0400	0.0953
		Wald					5.3634	5.0508	3.6723	3.8750
		α					0.0465	0.0317	0.0163	0.0124
		β					0.0850	0.0469	0.0539	0.0122
	0.2	New						7.3085	5.1506	3.3677
		α						0.0447	0.0404	0.0400
		β						0.0589	0.1000	0.0953
		Wald						6.4211	5.3828	3.4787
		α						0.0375	0.0249	0.0439
		β						0.0741	0.0609	0.0305
	0.3	New							7.0678	4.8007
		α							0.0454	0.0275
		β							0.0770	0.0902
		Wald							5.9209	4.8460
		α							0.0387	0.0398
		β							0.1173	0.0365

Table 3.4: Binomial Results for $n_T = 20$

π_0			π_1 0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
	0.1	New			13.9015	10.9703	9.0002	6.7246	6.7246	5.2201
		α			0.0472	0.0467	0.0475	0.0481	0.0481	0.0100
		β			0.0811	0.0653	0.0641	0.0900	0.0400	0.0961
		Wald			11.9671	8.4377	6.0426	5.4169	3.7788	3.9961
		α			0.0454	0.0376	0.0251	0.0180	0.0154	0.0122
		β			0.0989	0.0862	0.0865	0.0476	0.0539	0.0122
	0.2	New					13.2734	10.7311	9.4808	5.2201
		α					0.0444	0.0435	0.0400	0.0400
		β					0.0744	0.0903	0.0954	0.0961
		Wald					10.2606	7.6652	6.0148	3.5478
		α					0.0420	0.0297	0.0148	0.0435
		β					0.0723	0.0601	0.0616	0.0305
	0.3	New						15.1577	12.0552	7.4444
		α						0.0310	0.0271	0.0270
		β						0.1000	0.0938	0.0954
		Wald						10.5790	7.0062	5.1109
		α						0.0459	0.0417	0.0338
		β						0.0764	0.0678	0.0367

Table 3.5: Binomial Results for $n_T = 30$

π_0			π_1 0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
	0.1	New			18.1587	15.2001	11.9479	8.6750	8.6750	4.0000*
		α			0.0474	0.0496	0.0494	0.0484	0.0484	0.0453
		β			0.0826	0.0626	0.0640	0.0900	0.0400	0.0100
		Wald			13.5398	8.8907	6.1424	5.4412	3.7812	3.9999
		α			0.0441	0.0304	0.0245	0.0179	0.0154	0.0122
		β			0.0874	0.0867	0.0640	0.0476	0.0539	0.0122
	0.2	New				22.0886	18.6028	14.4298	13.0787	7.0768
		α				0.0422	0.0486	0.0486	0.0400	0.0400
		β				0.0860	0.0645	0.0900	0.0925	0.0961
		Wald				15.9520	11.1224	7.8916	6.0881	3.5495
		α				0.0390	0.0361	0.0269	0.0147	0.0435
		β				0.0998	0.0702	0.0603	0.0616	0.0305
	0.3	New					22.8769	19.1538	17.1158	10.0460
		α					0.0471	0.0476	0.0270	0.0270
		β					0.0956	0.0908	0.0978	0.0992
		Wald					17.3569	11.5612	7.2026	5.1275
		α					0.0400	0.0351	0.0402	0.0338
		β					0.1102	0.0760	0.0678	0.0367

Table 3.6: Binomial Results for $n_T = 40$

π_0			π_1 0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
	0.1	New		28.6824	22.4856	19.6224	15.1803	10.6011	10.6011	5.0000*
		α		0.0486	0.0478	0.0489	0.0499	0.0491	0.0491	0.0398
		β		0.0943	0.0786	0.0625	0.0640	0.0900	0.0400	0.1000
		Wald		23.2703	14.2361	9.0032	6.1566	5.4424	3.78122	4.0000
		α		0.0465	0.0337	0.0868	0.0245	0.0179	0.0154	0.0122
		β		0.1058	0.0882	0.0301	0.0865	0.0476	0.0539	0.0122
	0.2	New				28.1028	24.4068	18.2877	16.6321	8.9333
		α				0.0476	0.0463	0.0500	0.0146	0.0400
		β				0.0674	0.0640	0.0900	0.0616	0.0997
		Wald				17.1846	11.3995	7.9315	6.0965	3.5495
		α				0.0467	0.0337	0.0268	0.0147	0.0435
		β				0.0869	0.0704	0.0603	0.0616	0.0305
	0.3	New					30.3477	24.3836	22.9403	12.7527
		α					0.0425	0.0485	0.0270	0.0270
		β					0.0741	0.0900	0.0951	0.0993
		Wald					18.9418	11.9050	7.2356	5.1286
		α					0.0460	0.0317	0.0401	0.0338
		β					0.0768	0.0762	0.0678	0.0367

Table 3.7: Hypergeometric Results for $n_T = 10$

π_0			π_1 0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
	0.1	New					5.9235	4.3668	4.3668	3.3697
		α					0.0408	0.0431	0.0431	0.0099
		β					0.0770	0.0903	0.0399	0.0949
	0.2	New						7.3149	5.1550	3.3697
		α						0.0439	0.0402	0.0398
		β						0.0581	0.0992	0.0949
	0.3	New							7.0736	4.8052
		α							0.0448	0.0273
		β							0.0761	0.0896

Table 3.8: Hypergeometric Results for $n_T = 20$

π_0			π_1 0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
	0.1	New			13.7763	10.8738	9.0133	6.7308	6.7308	2.0000*
		α			0.0493	0.0498	0.0463	0.0471	0.0471	0.0465
		β			0.0781	0.0645	0.0638	0.0898	0.0398	0.1000
	0.2	New					13.2908	10.7414	9.4899	5.2247
		α					0.0430	0.0422	0.0398	0.0398
		β					0.0736	0.0901	0.0941	0.0955
	0.3	New						13.1629	12.0721	7.3560
		α						0.0438	0.0269	0.0268
		β						0.0996	0.0922	0.0994

Table 3.9: Hypergeometric Results for $n_T = 30$

π_0			π_1 0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
	0.1	New			18.2001	14.9808	11.9721	8.6850	8.6850	4.0000*
		α			0.0452	0.0497	0.0477	0.0472	0.0472	0.0429
		β			0.0817	0.0622	0.0637	0.0898	0.0398	0.1000
	0.2	New				21.7086	18.5264	14.4479	13.0612	7.0800
		α				0.0498	0.0481	0.0467	0.0398	0.0398
		β				0.0770	0.0641	0.0898	0.0984	0.09978
	0.3	New					22.9104	19.1758	17.0431	10.0671
		α					0.0448	0.0455	0.0268	0.0268
		β					0.0936	0.0905	0.0985	0.0977

Table 3.10: Hypergeometric Results for $n_T = 40$

π_0			π_1 0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
	0.1	New		28.6847	22.998	19.6704	14.9472	10.5957	10.5957	5.0000*
		α		0.0483	0.0487	0.0463	0.0500	0.0500	0.0500	0.0378
		β		0.0898	0.0778	0.0621	0.0637	0.0898	0.0398	0.1000
	0.2	New				28.1084	24.4261	18.2432	16.6563	8.9443
		α				0.0497	0.0496	0.0486	0.0398	0.0398
		β				0.0659	0.0637	0.0898	0.0960	0.0989
	0.3	New					30.1737	24.3469	22.9849	12.7119
		α					0.0500	0.0487	0.0268	0.0268
		β					0.0694	0.0898	0.0919	0.0993

In Tables 3.3 through 3.10, results marked with an asterisk (*) have multiple sets of boundaries with the same max{E(n)} and β value when testing π_0 vs. π_1. Thus, the result presented in the table is the set of boundaries that has the smallest value of α.

Taking note from the results presented in Tables 3.3 through 3.10, it can be seen that for $n_T = 10$, the new boundaries found for testing $\pi_0 = 0.1$ vs. $\pi_1 = 0.7$, $\pi_0 = 0.1$ vs. $\pi_1 = 0.9$, $\pi_0 = 0.2$ vs. $\pi_1 = 0.8$, $\pi_0 = 0.2$ vs. $\pi_1 = 0.9$, and $\pi_0 = 0.3$ vs. $\pi_1 = 0.9$ all reduced the minimized max{E(n)} when compared to the Wald boundaries for the binomial case. For $n_T = 10$, the new boundaries found for testing $\pi_0 = 0.1$ vs. $\pi_1 = 0.7$ reduces the expected number of observations by 0.6875, which was greatest difference observed.

It was not until close to the end of conducting this research that the Wald boundaries were calculated and analyzed. Only after analyzing this summary information did we notice a discrepancy between the Wald boundaries and the new boundaries. Because we set out to determine all sets of possible boundaries for any given n_T, the Wald boundaries should have been included in our possible set of boundaries with both α and β bounded. Thus, the new "best" boundaries should be no worse than the Wald boundaries. When analyzing the results, we should have seen that if the minimized max{E(n)} of the new boundaries was not smaller than the max{E(n)} of the Wald boundaries, then the minimized max{E(n)} should have at most been equal to that of the Wald boundaries.

After retracing our steps, it was discovered that although our R functions correctly reduce a given collection of lower/upper boundaries to those for which both α and β are bounded, the process in which we generated all possible boundaries was not exhaustive for all n_T. The idea of creating all possible bounds for n_T under the conditions that the lower and upper bounds must both begin at zero or one, the lower and upper bounds must have a non-decreasing step size of at most one, and the lower and upper bounds may touch one another but not cross is correct. Thus, we discovered that not all possible boundaries under the three conditions were generated using the algorithm that we created; only a subset of the possible boundaries was generated. Unfortunately, as n_T grows larger, it appears that more and more boundaries were missed. When creating the method for generating all possible boundaries for n_T, we tested the method by hand for several small n_T values and incorrectly assumed it would work for all n_T values.

Since $n_T = 10$ is the smallest n_T value included in this research, we were able to find new boundaries with smaller minimized max{E(n)} than their corresponding Wald boundaries because enough of the possible boundaries were generated. Thus, we are optimistic that this method actually would sharpen many more of the boundaries for the SPRT if all possible boundaries are generated.

Chapter 4: Simulation Studies

4.1 Simulation Method

To test the performance of the boundaries, a simulation study was performed for each set of new n_T-boundaries that minimized the max{E(n)} for a specified π so that we could observe the value of the required sample size and the decision made in each simulation. Because we have two values being tested of the known unknown parameter π, $\pi = \pi_0$ and $\pi = \pi_1$, we performed each simulation study for a set of bounds under both assumptions individually, similar to what was done in [4].

For each set of lower and upper bounds which minimize the max{E(n)}, the rejection and acceptance regions are determined and decisions are made regarding the outcomes on the truncation boundary. When sampling from an infinite population (or sampling with replacement), sample data is created with the specified probability of success, $\pi = \pi_0$ or $\pi = \pi_1$, by using the binomial random variable generator in R with n_T trials [5]. The total number of defectives at stage n is calculated for $1 \leq n \leq n_T$. When sampling without replacement, a population with a specified size of 1,000 is created with the correct proportion of defective and non-defective items. From the created population we randomly sample n_T items from the population without replacement (resulting in a hypergeometric random variable), and the number of defectives at stage n is calculated for $1 \leq n \leq n_T$. The SPRT is then performing using this generated data and the given set of boundaries. For each simulation setting (a given value of π,

a given set of boundaries, and a given distribution), 10,000 samples were simulated and analyzed with SPRT.

For each set of new boundaries that were determined to minimized the max{E(n)}, simulations were run for both $\pi = \pi_0$ and $\pi = \pi_1$. We also performed simulations for the boundaries where the max{E(n)} was within one of the minimized max{E(n)}. Additionally, when sampling with replacement, the Wald boundaries were also studied via simulation. Since the simulations were conducted before the results were recognized as incomplete, 53 Wald boundaries, 216 minimized max{E(n)} boundaries, and 1,216 boundaries with a max{E(n)} within one of the minimized max{E(n)} were analyzed. The R [5] function created for the simulation study can be found in Appendix A.2.5.

4.2 Simulation Results

Although we know that we do not have all possible boundaries, we also do not have the Wald boundaries with which to compare in the hypergeometric case. Thus, for each combination of π_0 and π_1 we present max{E(n)} (calculated as described in section 3.1), the simulated average required number of observations when $\pi = \pi_0$, denoted as $\hat{E}_0(n)$, and the simulated average required number of observations when $\pi = \pi_1$, denoted as $\hat{E}_1(n)$. These results can be seen in Tables 4.1 through 4.4. For max{E(n)} we notice that for $\pi_0 = 0.1$, as π_1 increases, the value of max{E(n)} either stays the same or decreases; this is consistent with the results found in [3]. Considering $\pi_0 = 0.2$ and the lowest satisfying π_1 value, max{E(n)} starts at a higher value than the lowest

combination of $\pi_0 = 0.1$ and π_1, but the pattern does continue that as π_0 and π_1 become more distant, the value of max{E(n)} either decreases or stays the same. However, in simulation, this pattern also came very close to holding true for $\hat{E}_0(n)$ and $\hat{E}_1(n)$. If we had increased the number of simulations, it is possible that the pattern would have always held for $\hat{E}_0(n)$ and $\hat{E}_1(n)$.

Table 4.1: Hypergeometric Simulation Results for $n_T = 10$

π_0	π_1	max{E(n)}	$\hat{E}_0(n)$	$\hat{E}_1(n)$
0.1	0.6	5.9235	4.7752	4.6734
	0.7	4.3668	3.3625	3.1840
	0.8	4.3668	3.3849	2.8129
	0.9	3.3697	2.1955	3.0085
0.2	0.7	7.3149	6.2719	5.6530
	0.8	5.1550	3.6390	4.5566
	0.9	3.3697	2.3866	2.9850
0.3	0.8	7.0736	5.5370	6.2295
	0.9	4.8052	2.8088	4.4061

Table 4.2: Hypergeometric Simulation Results for $n_T = 20$

π_0	π_1	max{E(n)}	$\hat{E}_0(n)$	$\hat{E}_1(n)$
0.1	0.4	13.7763	12.2465	9.7108
	0.5	10.8738	9.1128	7.5435
	0.6	9.0133	7.3063	6.2372
	0.7	6.7308	5.1811	5.1907
	0.8	6.7308	5.1525	4.7787
	0.9	2.0000*	1.9029	1.9957
0.2	0.6	13.2908	11.1347	10.8846
	0.7	10.7414	8.3162	9.1039
	0.8	9.4899	7.1898	7.9287
	0.9	5.2247	3.6038	4.5873
0.3	0.7	13.1629	11.0000	11.6911
	0.8	12.0721	8.3228	10.8826
	0.9	7.3560	4.1959	6.7192

Table 4.3: Hypergeometric Simulation Results for $n_T = 30$

π_0	π_1	max{E(n)}	$\hat{E}_0(n)$	$\hat{E}_1(n)$
0.1	0.4	18.2001	14.8325	13.8146
	0.5	14.9808	12.3564	9.3850
	0.6	11.9721	9.9179	7.8238
	0.7	8.6850	6.8775	6.5478
	0.8	8.6850	6.8910	5.9896
	0.9	4.0000*	3.1406	3.9953
0.2	0.5	21.7086	19.0467	17.2162
	0.6	18.5264	15.7705	14.0164
	0.7	14.4479	11.905	11.7568
	0.8	13.0612	9.6042	11.2064
	0.9	7.0800	4.7981	6.3865
0.3	0.6	22.9104	20.6262	20.2472
	0.7	19.1758	16.1491	16.8933
	0.8	17.0431	11.1249	15.5119
	0.9	10.0671	5.5498	9.1159

Table 4.4: Hypergeometric Simulation Results for $n_T = 40$

π_0	π_1	max{E(n)}	$\hat{E}_0(n)$	$\hat{E}_1(n)$
0.1	0.3	28.6847	25.4916	22.0905
	0.4	22.4998	18.8053	15.9250
	0.5	19.6704	16.1707	13.0795
	0.6	14.9472	12.3363	9.3464
	0.7	10.5957	8.6470	7.8012
	0.8	10.5957	8.6645	7.1912
	0.9	5.0000*	3.9628	4.9896
0.2	0.5	28.1084	25.0704	22.4594
	0.6	24.4261	20.8494	18.6746
	0.7	18.2432	15.4056	14.3170
	0.8	16.6563	11.9775	14.7202
	0.9	8.9443	6.0116	7.9092
0.3	0.6	30.1737	27.005	25.3294
	0.7	24.3469	21.3139	20.6702
	0.8	22.9849	15.2466	20.2881
	0.9	12.7119	6.9240	11.4694

In Tables 4.2 through 4.4, results marked with an asterisk (*) in the max{E(n)} column have multiple sets of boundaries with the same max{E(n)} when testing π_0 vs. π_1. Thus, the result presented in the table is for one such set of boundaries.

Since we had the available Wald boundaries for comparison in the binomial case, we also considered max{E(n)}, $\hat{E}_0(n)$, and $\hat{E}_1(n)$ for each combination of π_0 and π_1 using the Wald boundaries, as seen in Tables 4.5 through 4.8. In Tables 4.7 and 4.8, results marked with an asterisk (*) in the max{E(n)} column have multiple sets of boundaries with the same max{E(n)} when testing π_0 vs. π_1. Thus, the result presented in the table is for one such set of boundaries.

Table 4.5: Binomial Simulation Results for $n_T = 10$

π_0	π_1	max{E(n)}	$\hat{E}_0(n)$	$\hat{E}_1(n)$	Wald max{E(n)}	Wald $\hat{E}_0(n)$	Wald $\hat{E}_1(n)$
0.1	0.6	5.9184	4.7578	4.6637	5.3634	4.0088	4.1648
	0.7	4.3633	3.3324	3.2164	5.0508	3.6937	3.6030
	0.8	4.3633	3.3536	2.7948	3.6723	2.4633	2.8029
	0.9	3.3677	2.2035	3.0484	3.8750	2.4292	2.4408
0.2	0.7	7.3085	6.2659	5.6650	6.4211	4.7574	5.2695
	0.8	5.1506	3.6512	4.5939	5.3828	3.1551	4.4190
	0.9	3.3677	2.4059	3.0185	3.4787	2.4989	2.5743
0.3	0.8	7.0678	5.5660	6.2276	5.9209	3.9921	4.9784
	0.9	4.8007	2.7889	4.4095	4.8460	3.0866	3.9172

Table 4.6: Binomial Simulation Results for $n_T = 20$

π_0	π_1	max{E(n)}	$\hat{E}_0(n)$	$\hat{E}_1(n)$	Wald max{E(n)}	Wald $\hat{E}_0(n)$	Wald $\hat{E}_1(n)$
0.1	0.4	13.9015	12.1465	9.9026	11.9671	9.3658	9.0282
	0.5	10.9703	9.1120	7.5400	8.4377	6.0968	5.8014
	0.6	9.0002	7.3055	6.2129	6.0426	4.1488	4.2644
	0.7	6.7246	5.1944	5.1775	5.4169	3.7271	3.5657
	0.8	6.7246	5.2175	4.7963	3.7788	2.4501	2.8090
	0.9	5.2201	3.2995	4.6401	3.9961	2.4436	2.4404
0.2	0.6	13.2734	11.1300	10.9368	10.2606	7.1108	7.5998
	0.7	10.7311	8.3850	9.0972	7.6652	4.8856	5.5753
	0.8	9.4808	7.1704	8.1960	6.0148	3.2472	4.4810
	0.9	5.2201	3.5953	4.6090	3.5479	2.5255	2.5746
0.3	0.7	15.1577	11.4480	13.8293	10.5790	6.8894	8.5362
	0.8	12.0552	8.2872	10.7970	7.0062	4.1630	5.2008
	0.9	7.4444	4.1881	6.7286	5.1109	3.1001	3.9475

Table 4.7: Binomial Simulation Results for $n_T = 30$

π_0	π_1	max{E(n)}	$\hat{E}_0(n)$	$\hat{E}_1(n)$	Wald max{E(n)}	Wald $\hat{E}_0(n)$	Wald $\hat{E}_1(n)$
0.1	0.4	18.1587	14.9030	13.8616	13.5398	9.6654	9.4107
	0.5	15.2001	12.5394	9.4200	8.8907	6.0493	5.8684
	0.6	11.9479	9.8374	7.8029	6.1424	4.1344	4.2086
	0.7	8.6750	6.8629	6.5122	5.4412	3.7383	3.5784
	0.8	8.6750	6.8209	6.0007	3.7812	2.4638	2.8200
	0.9	4.0000*	3.2063	4.0035	4.0000	2.4322	2.4498
0.2	0.5	22.0886	19.1325	18.6147	15.9520	11.0599	12.2338
	0.6	18.6028	15.8078	13.9482	11.1224	7.2013	7.7357
	0.7	14.4298	11.7950	11.7403	7.8916	4.9505	5.6244
	0.8	13.0787	9.6233	11.2948	6.0881	3.2038	4.4769
	0.9	7.0768	4.8042	6.1967	3.5495	2.4802	2.5681
0.3	0.6	22.8769	20.6655	20.1917	17.3569	12.4320	13.6627
	0.7	19.1538	16.2480	16.9735	11.5612	6.9737	8.5800
	0.8	17.1158	11.1155	15.4351	7.2026	4.1674	5.2306
	0.9	10.0460	5.5777	9.2890	5.1275	3.0758	3.9457

Table 4.8: Binomial Simulation Results for $n_T = 40$

π_0	π_1	max{E(n)}	$\hat{E}_0(n)$	$\hat{E}_1(n)$	Wald max{E(n)}	Wald $\hat{E}_0(n)$	Wald $\hat{E}_1(n)$
0.1	0.3	28.6824	25.7380	22.0599	23.2703	17.5602	17.521
	0.4	22.4856	18.6465	16.1252	14.2361	9.8535	9.4997
	0.5	19.6224	16.0586	13.1422	9.0032	6.1039	5.9216
	0.6	15.1803	12.7834	9.3408	6.1566	4.1556	4.2586
	0.7	10.6011	8.7007	7.7807	5.4424	3.7168	3.6251
	0.8	10.6011	8.5831	7.2095	3.7812	2.4409	2.8022
	0.9	5.0000*	3.6924	4.9796	4.0000	2.4264	2.4216
0.2	0.5	28.1028	24.9114	22.4476	17.1846	11.3336	12.4192
	0.6	24.4068	20.7344	18.7275	11.3995	7.3563	7.7136
	0.7	18.2877	15.3981	14.3096	7.9315	4.8828	5.6380
	0.8	16.6321	11.9965	15.1101	6.0965	3.2199	4.5035
	0.9	8.9333	6.0203	7.8245	3.5495	2.4921	2.5740
0.3	0.6	30.3477	27.2511	26.4759	18.9418	12.5169	14.0898
	0.7	24.3836	21.0709	20.7718	11.9050	6.9735	8.7327
	0.8	22.9403	15.2968	20.5287	7.2356	4.2092	5.2435
	0.9	12.7527	6.9587	11.8276	5.1286	3.1058	3.9234

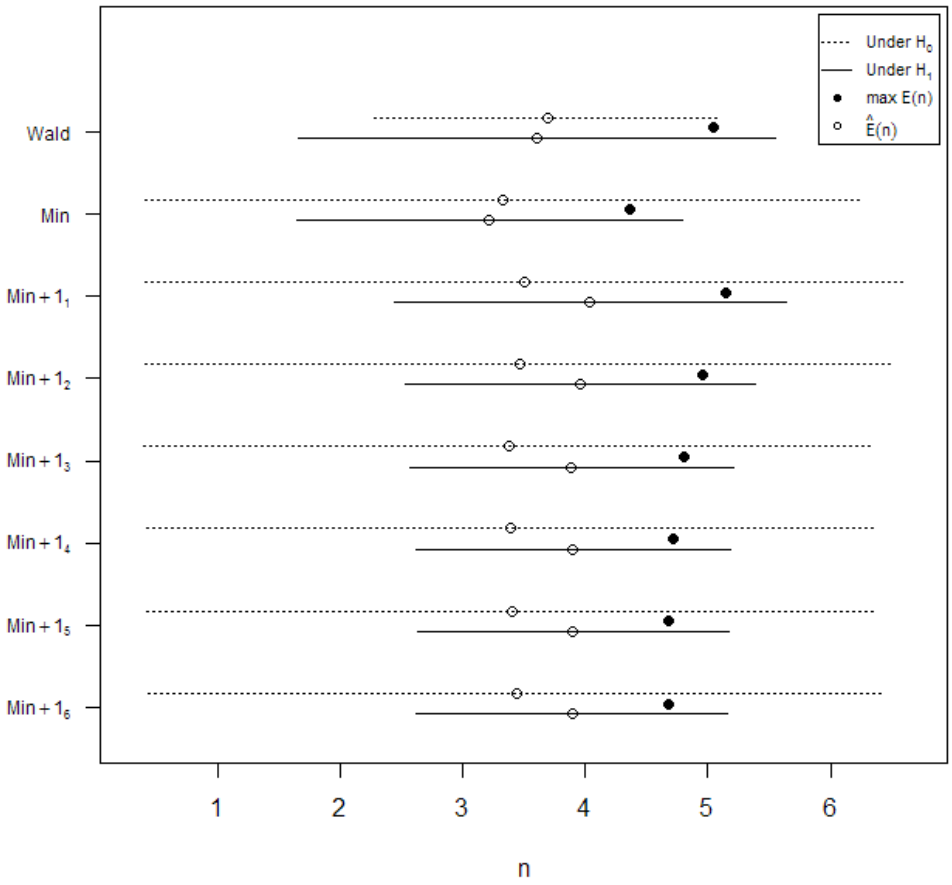
We observed that the same pattern described in [3] was exhibited in the binomial case for both the new boundaries and the Wald boundaries, as it was in the hypergeometric case. After closer observation, we also found that for a few of the new boundaries, $\hat{E}_0(n)$, and $\hat{E}_1(n)$ were smaller than that of their Wald counterparts. These instances are highlighted in Tables 4.5 through 4.8. To be exact, $\hat{E}_0(n)$ was found to be better on four occurrences, and $\hat{E}_1(n)$ was found to be better on three occurrences, all occurring when $n_T = 10$. These instances correspond to the new boundaries that were found to be better than the Wald boundaries, described in section 3.2.

For an additional qualification of what could be considered as the best boundary, we have also considered the boundaries that in simulation minimize

the variability in required sample size, similar to [1]. Since only five new sets of boundaries were found to have minimized max{E(n)} in comparison to the Wald boundaries, we only present further results for the five cases where an improvement was found. However, many more simulation results are available from the author.

Figures 4.1 through 4.5 detail the results for the combinations of π_0 and π_1 for which an improvement in maximum expected sample size was found (with the new boundaries labeled as "Min"). For each set of boundaries, the horizontal dotted line extends one standard deviation above and below $\hat{E}_0(n)$, which is indicated by the open circle. Similarly, the solid horizontal line extends one standard deviation above and below $\hat{E}_1(n)$, which is indicated by the open circle. Additionally, the max{E(n)} is presented as a solid circle. Since we also conducted simulations for the sets of boundaries whose max{E(n)} is within one of the minimized max{E(n)} value, we present results for those observations as well. These boundaries are labeled as "Min +1" in the figures, and the subscript serves as a signifying index for each considered set of boundaries for the hypotheses being tested.

Figure 4.1: Binomial $n_T = 10$ Simulation with $\pi_0 = 0.1$ and $\pi_1 = 0.7$



In Figure 4.1, we notice that under the null hypothesis ($\pi_0 = 0.1$), based upon the simulation performed, all new boundaries considered tend to require a smaller sample size to make a decision in the test than Wald's boundaries; however, less variation is observed in the new boundaries under the alternative hypothesis ($\pi_1 = 0.7$). To the contrary, under the alternative hypothesis, the use of Wald's boundaries tends to require a smaller sample size than the new boundaries (except for the new "best" boundary), and the null case exhibits less

variation for the Wald boundaries. Additionally, all new max{E(n)} (except for Min $+1_1$) are less than max{E(n)} for Wald.

Figure 4.2 demonstrates that although max{E(n)} has been reduced with the new boundaries, under simulation Wald's boundaries result in more consistency in that it has small variation under both the null ($\pi_0 = 0.1$) and alternative hypotheses ($\pi_1 = 0.9$). In simulation using the new "best" bounds and the bounds whose max{E(n)} are within one, under the null hypothesis all but one (Min $+1_7$) has less variation than under Wald's simulation.

Figure 4.2: Binomial $n_T = 10$ Simulation with $\pi_0 = 0.1$ and $\pi_1 = 0.9$

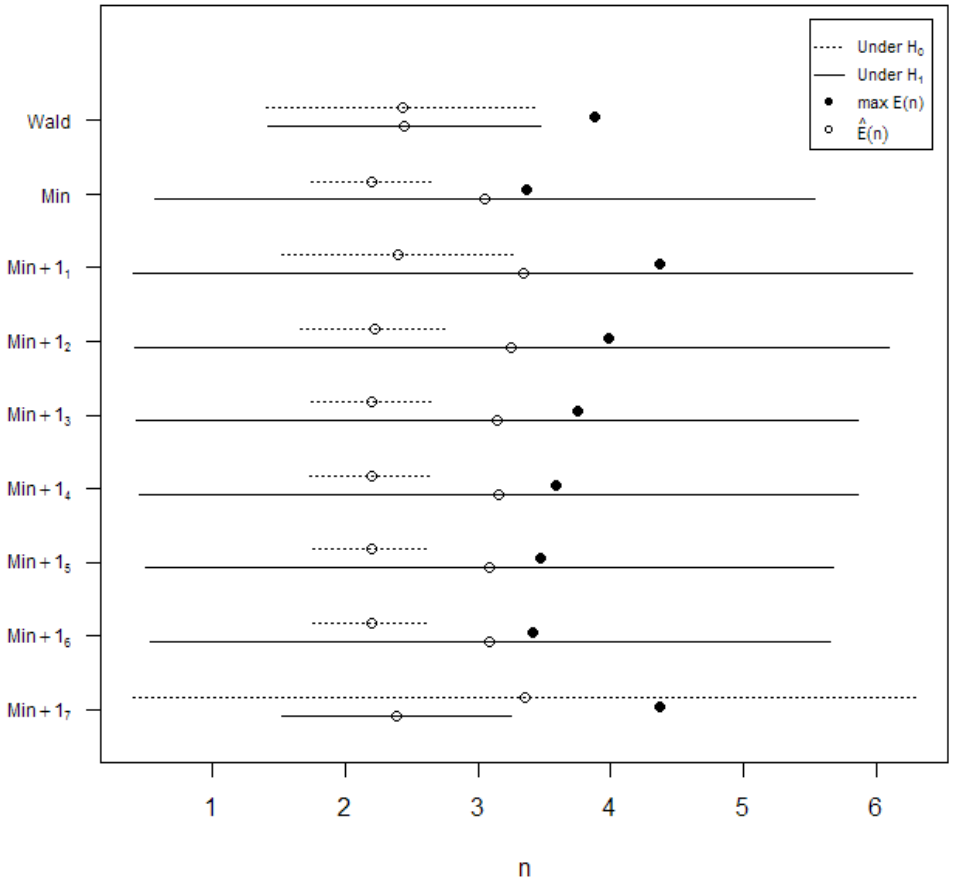


Figure 4.3 shows that although max{E(n)} is reduced, Wald appears again to be more consistent under both the null ($\pi_0 = 0.2$) and alternative ($\pi_1 = 0.8$) hypotheses. Also, max{E(n)} for the new bounds is the only max{E(n)} value which is less than Wald's. Figures 4.4 and 4.5 demonstrate that under the alternative hypothesis ($\pi_1 = 0.9$), much more variability is shown in the new "best" bounds than in Wald's boundaries.

Figure 4.3: Binomial $n_T = 10$ Simulation with $\pi_0 = 0.2$ and $\pi_1 = 0.8$

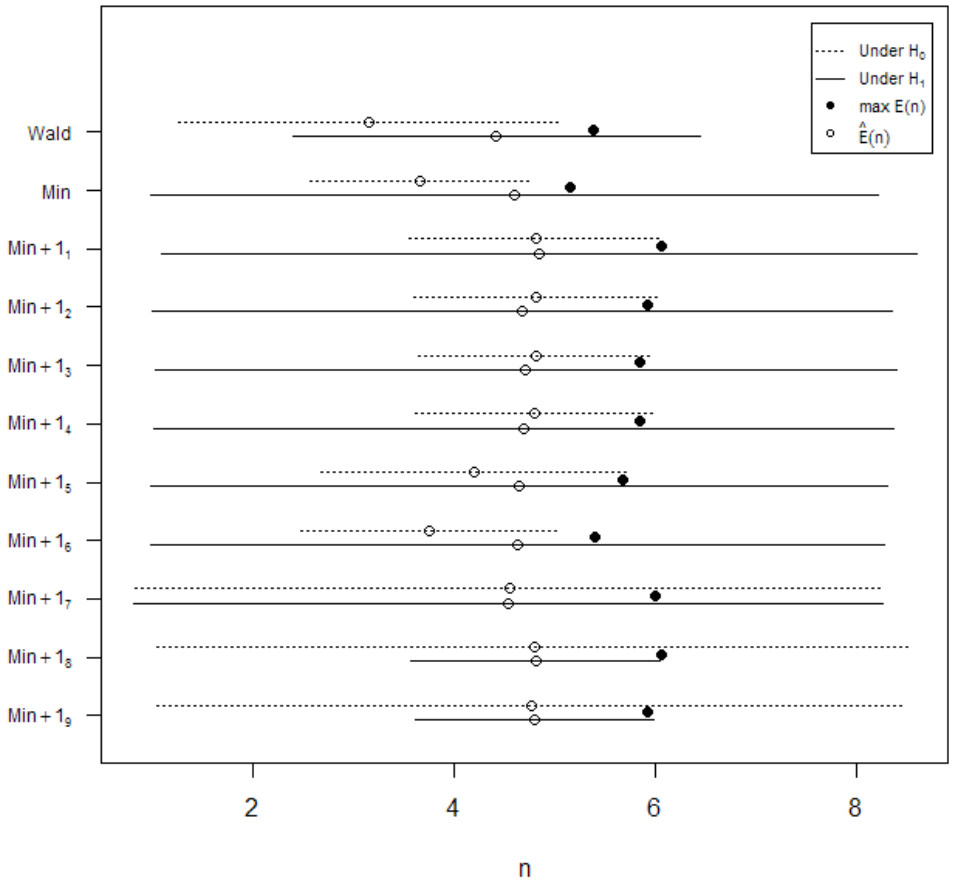


Figure 4.4: Binomial $n_T = 10$ Simulation with $\pi_0 = 0.2$ and $\pi_1 = 0.9$

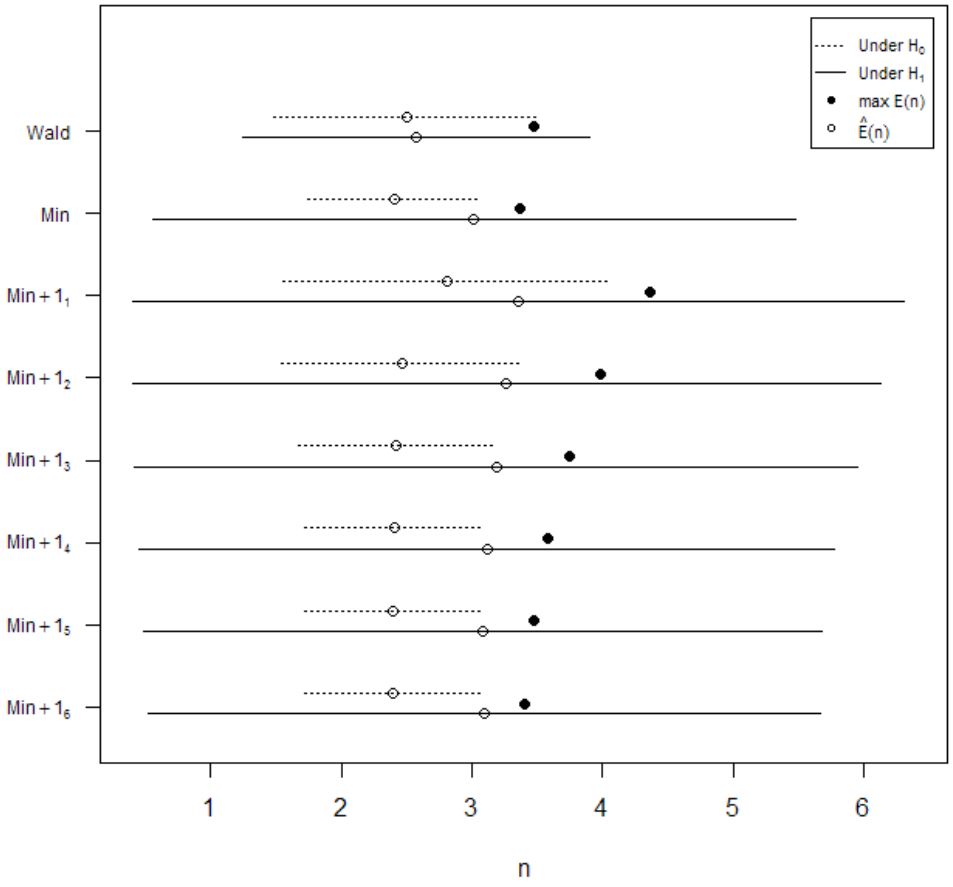
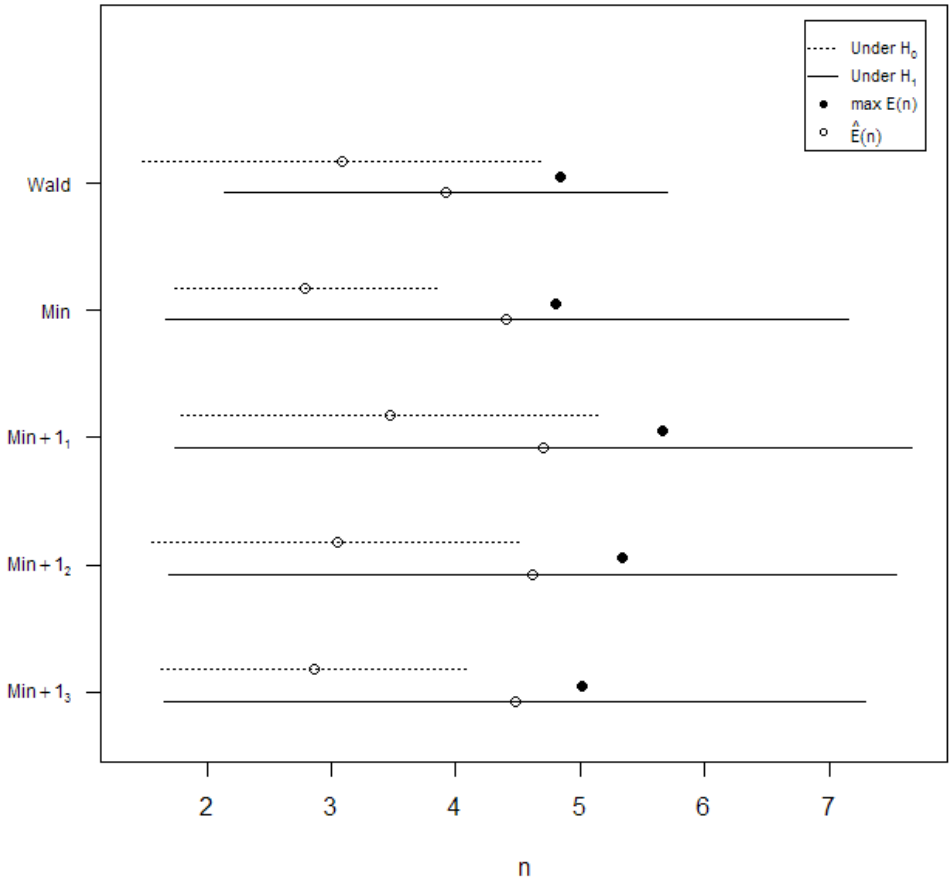


Figure 4.5: Binomial $n_T = 10$ Simulation with $\pi_0 = 0.3$ and $\pi_1 = 0.9$



Chapter 5: Conclusion and Future Work

For this thesis, we set out to determine a possible method to reduce the expected number of observations needed in a sequential probability ratio test. The goal was to generate all possible boundaries for a given value of n_T that preserve a given set of error rates (α,β). We conducted this research on binary data for binomial and hypergeometric random variables. In the binomial case, we were able to compare the newly created boundaries to the boundaries established by [6]. In some cases, we were able to find boundaries that reduced the expected number of observations for making a decision in the test from that which is required by the standard Wald boundaries. Although we believe that the overall method that we intended to use is correct, we discovered that there was a problem in one of the algorithms developed to implement this method. Additionally, we were able to conduct simulation studies on the boundaries to estimate max{E(n)} and the amount of variability of the required sample size in practice, under both the null hypothesis and the alternative hypothesis.

In the future, the first thing that needs to be determined is a correct algorithm for generating all possible boundaries for n_T. Although the idea of considering all possible boundaries with the conditions of beginning at zero or one, have a step size of at most one, and being allowed to touch one another but not cross is correct, the method that was used to implement it appears to be incomplete and not exhaustive. The overall method that has been developed in this research for sharpening the boundaries is hopeful, so the generation of all

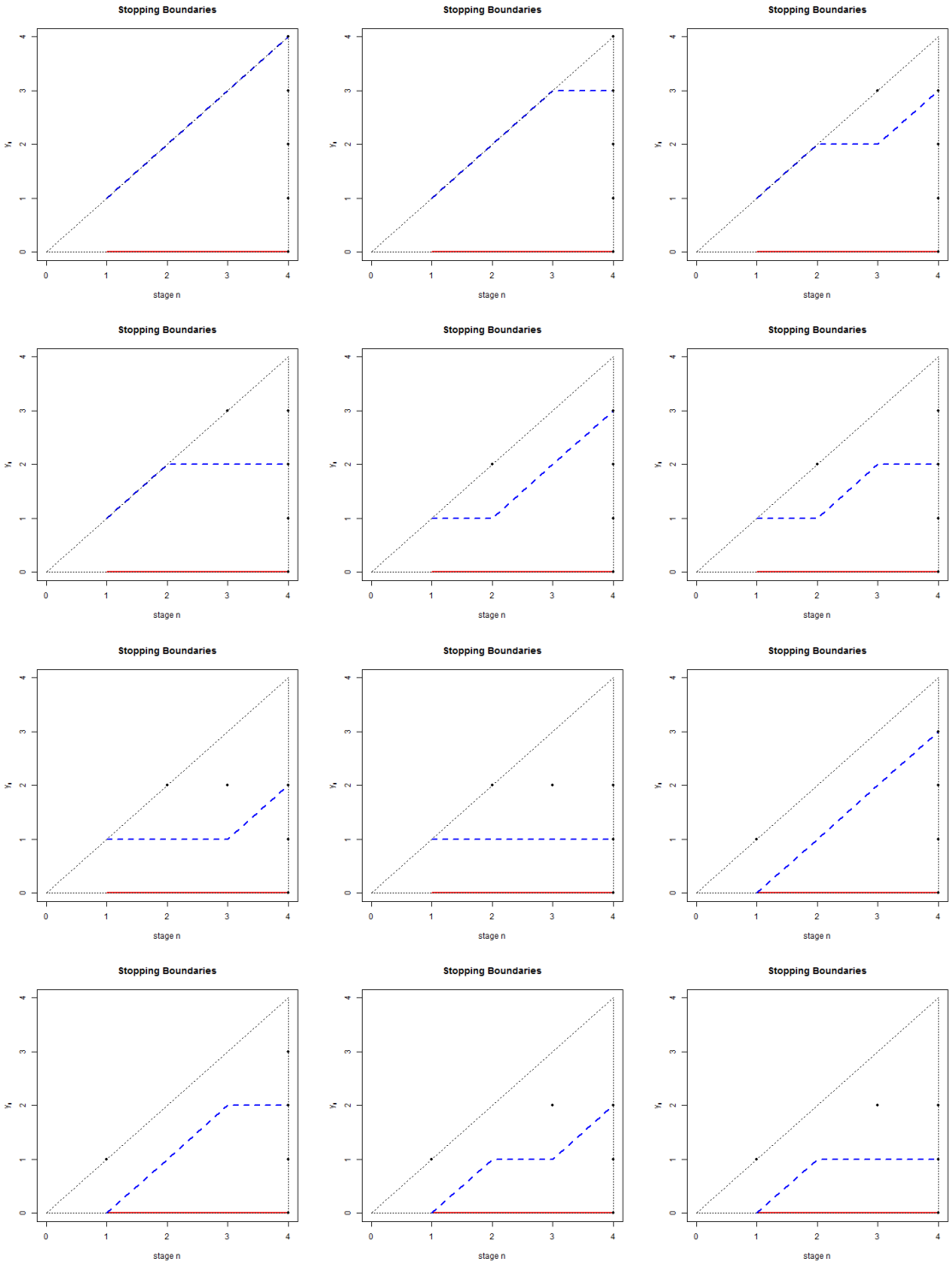
possible boundaries should be a top priority. Additionally, the efficiency of the correct R functions that have been created (alpha.bounded and errors.bounded) needs to be improved. At this time, we were only able to consider n_T as large as 40 because some of these functions take days to run on a standard computer. Now knowing that not even all of the possible boundaries were considered, it is unimaginable how long it would have taken to run had an exhaustive list of boundaries been used.

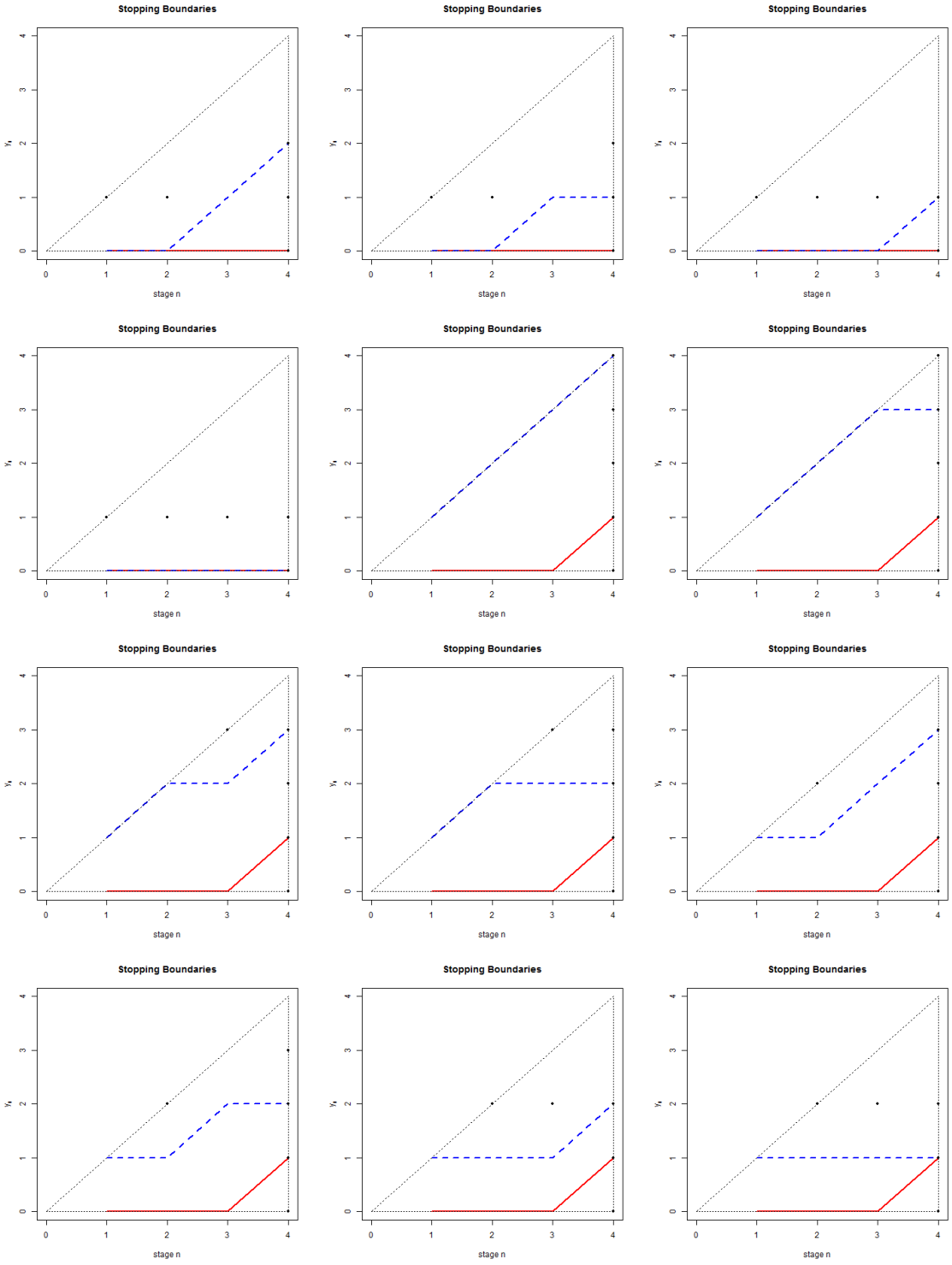
Alternative areas to explore in this research are also available. Since all data that we have considered has been binary, we could explore testing data that is not binary. Additionally, we could consider testing composite alternative hypotheses ($<$, $>$, $\neq$) as opposed to the simple alternative that was considered in this research. Also, since little to no research has been conducted on the SPRT in the hypergeometric case, there is much left to consider, including the Wald boundaries. We could also take direction from [3] and explore testing in the situation of correlated binary data.

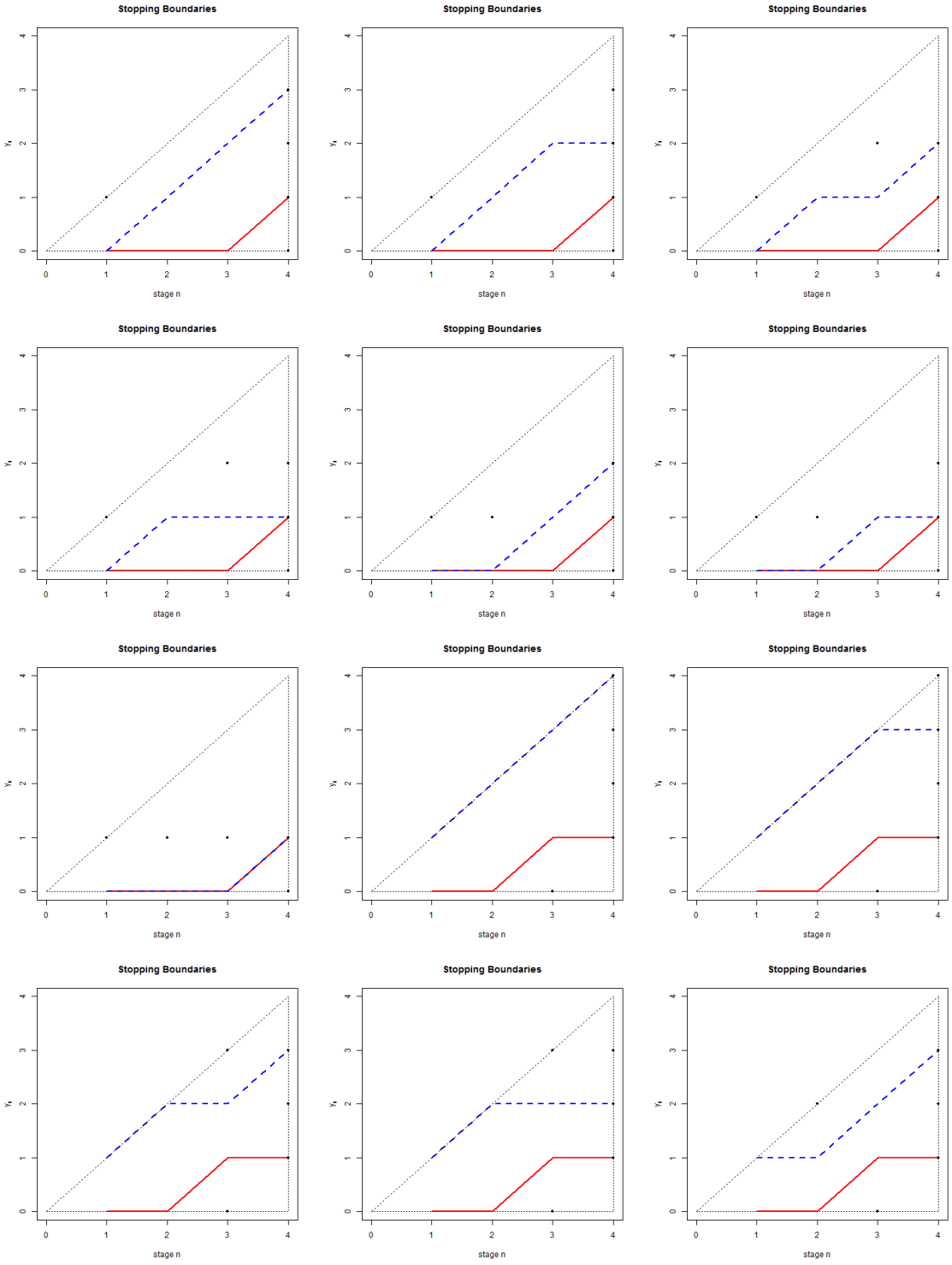
In conclusion, we have explored the sequential probability ratio test beginning with Wald's first introduction of the test. We also examined expansions on the SPRT that other authors have considered. Utilizing the research presented in [2], we have presented and attempted to develop a potential new method for sharpening the boundaries of the sequential probability ratio test.

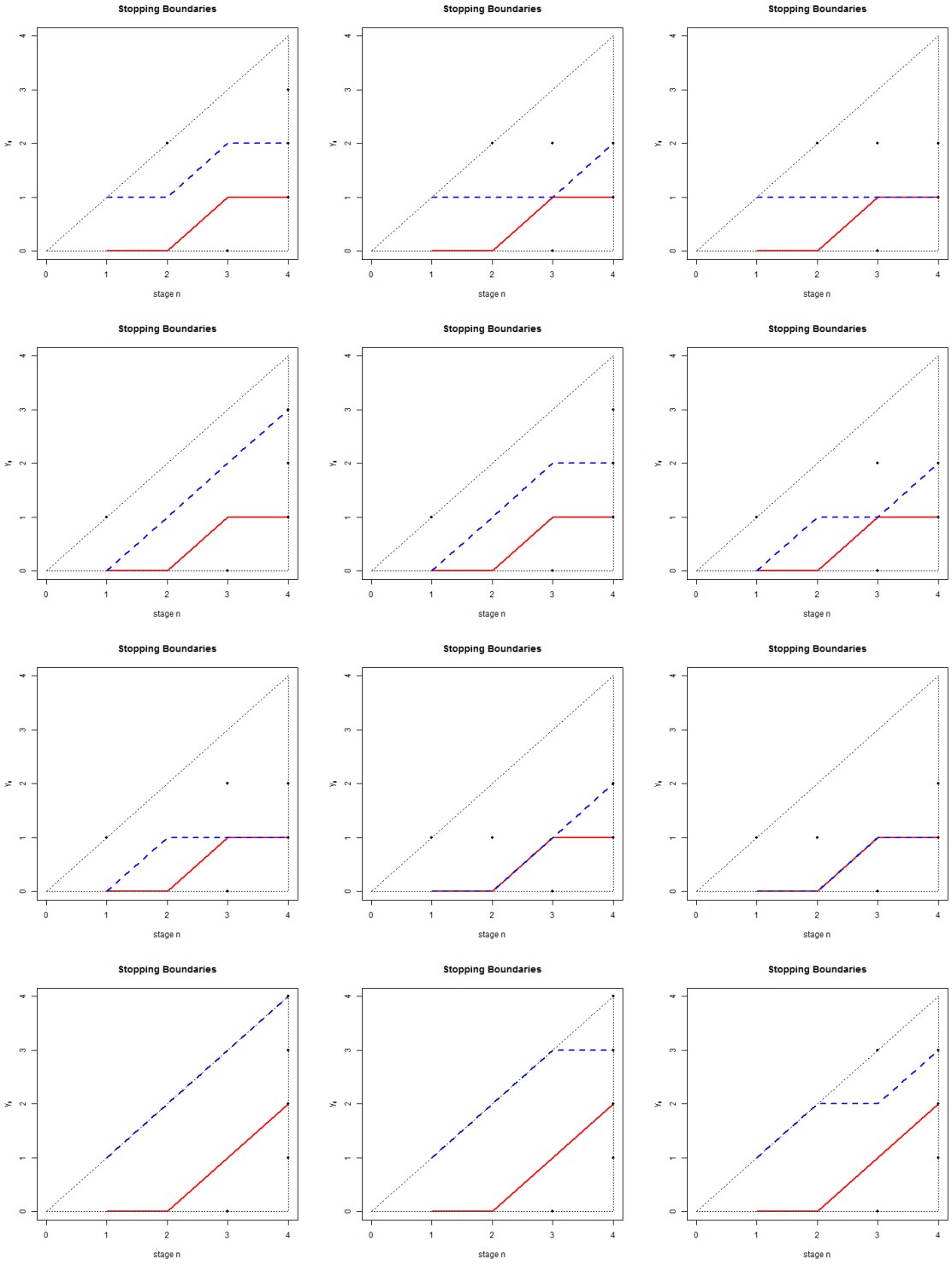
APPENDIX

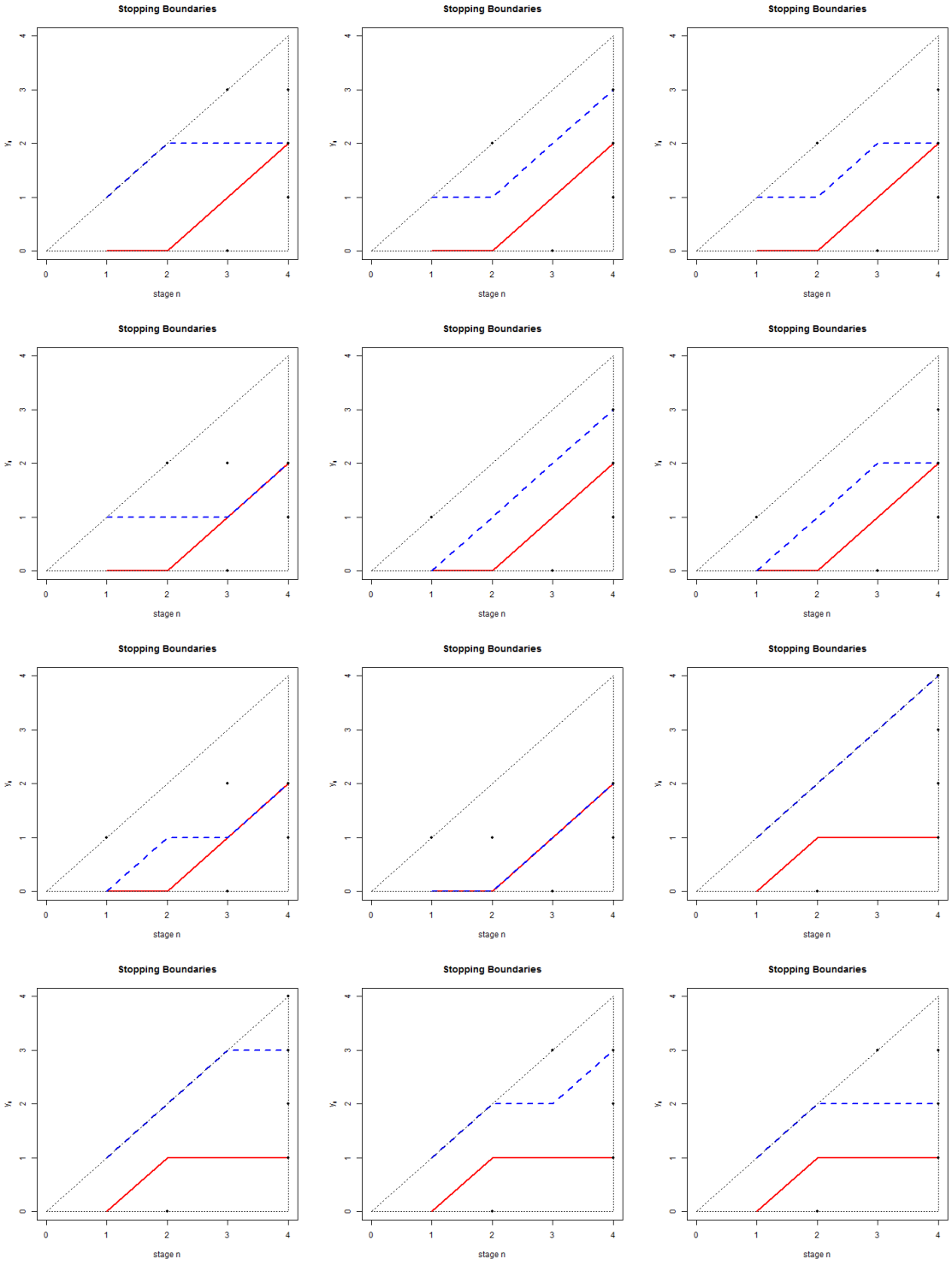
A.1 Boundaries for $n_T = 4$

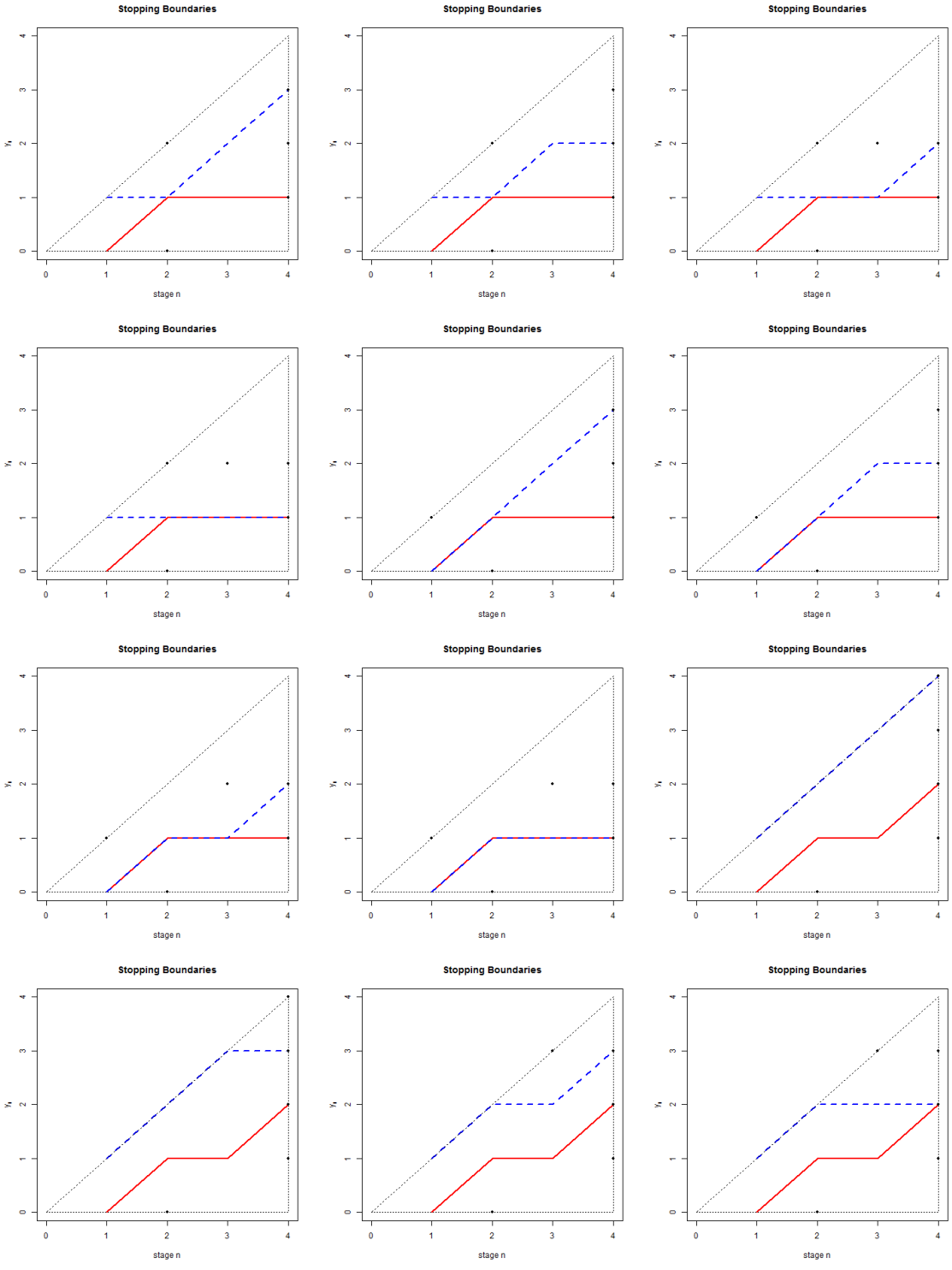


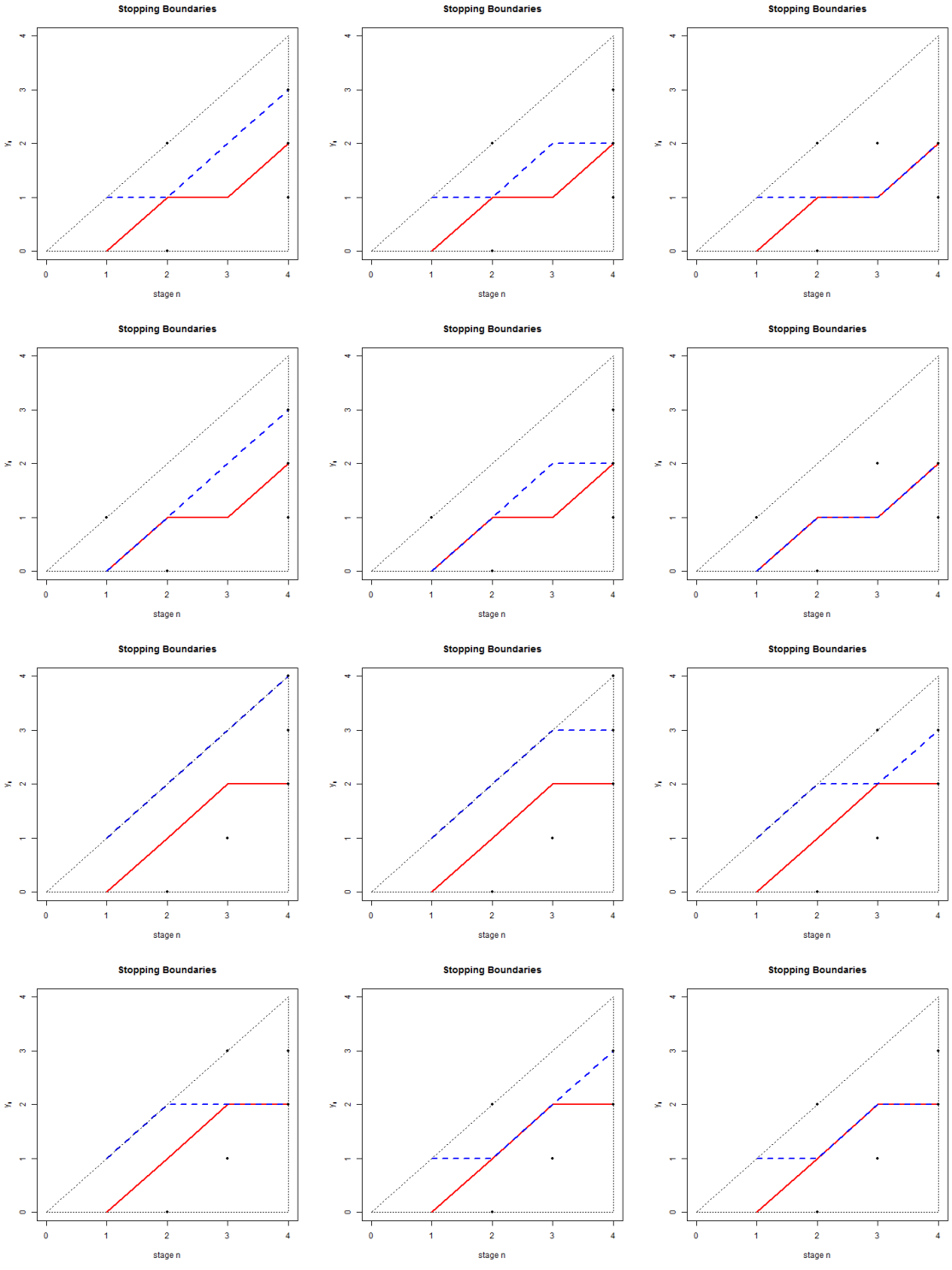

Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n

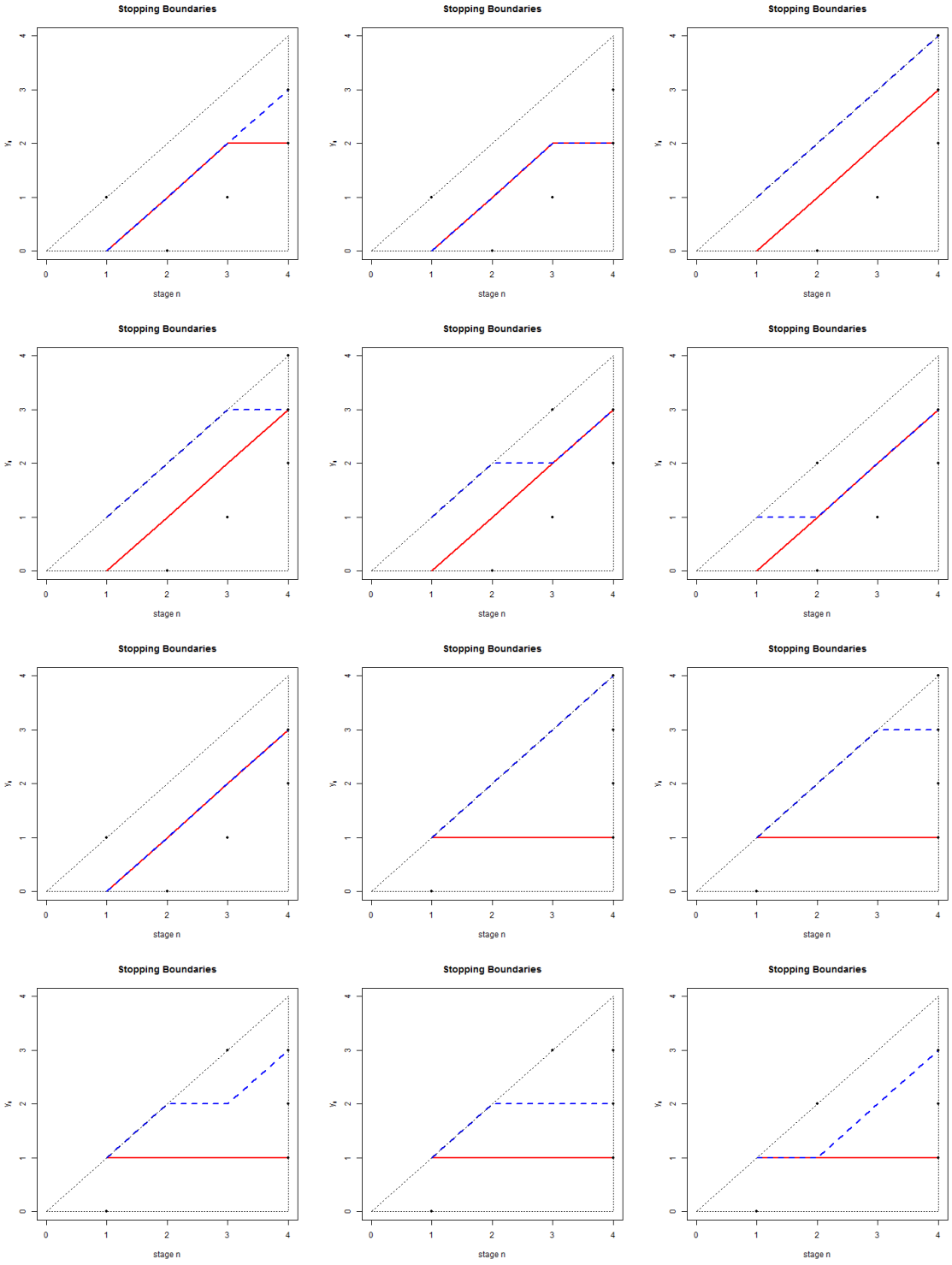

Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n

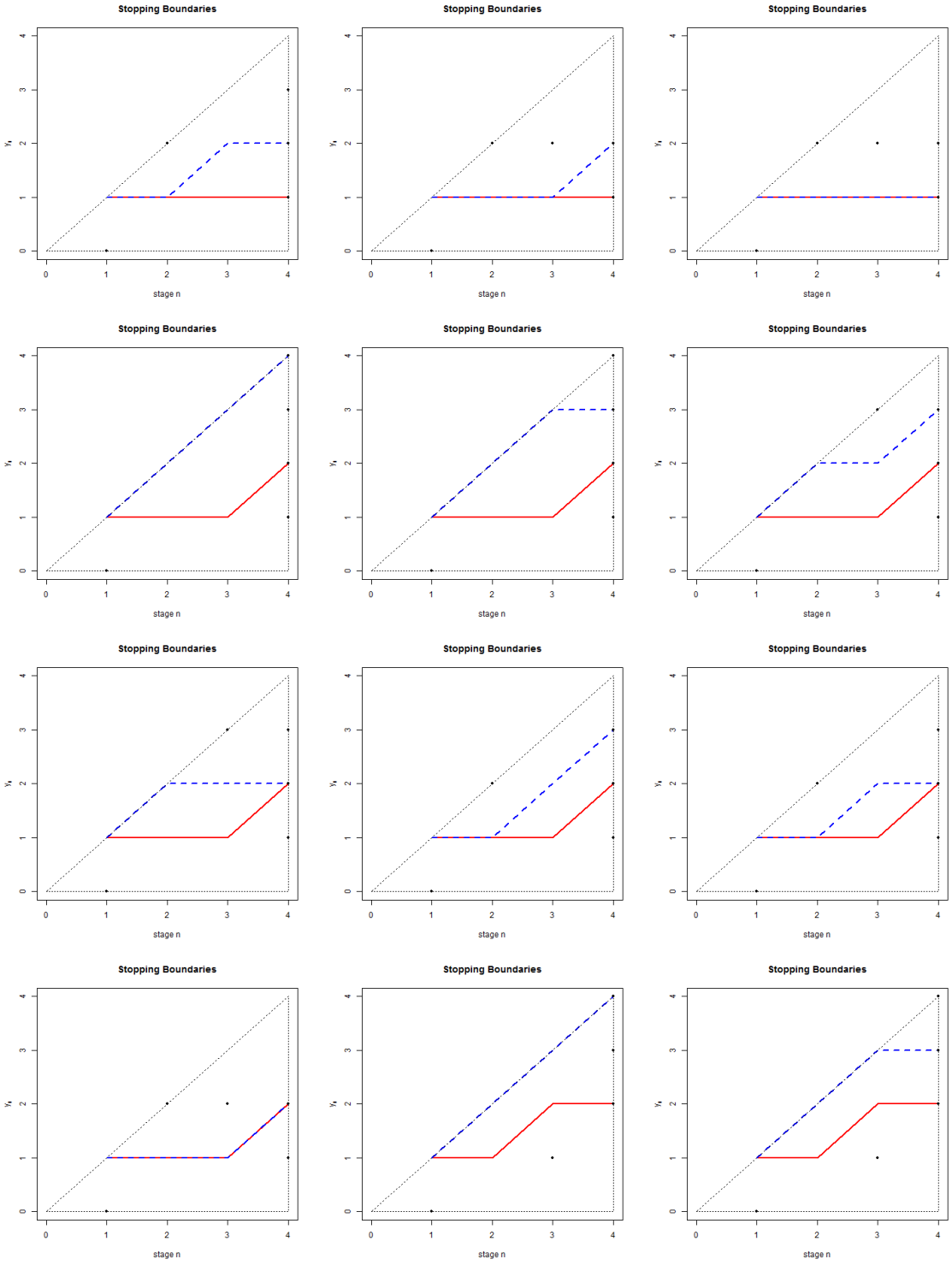

Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n

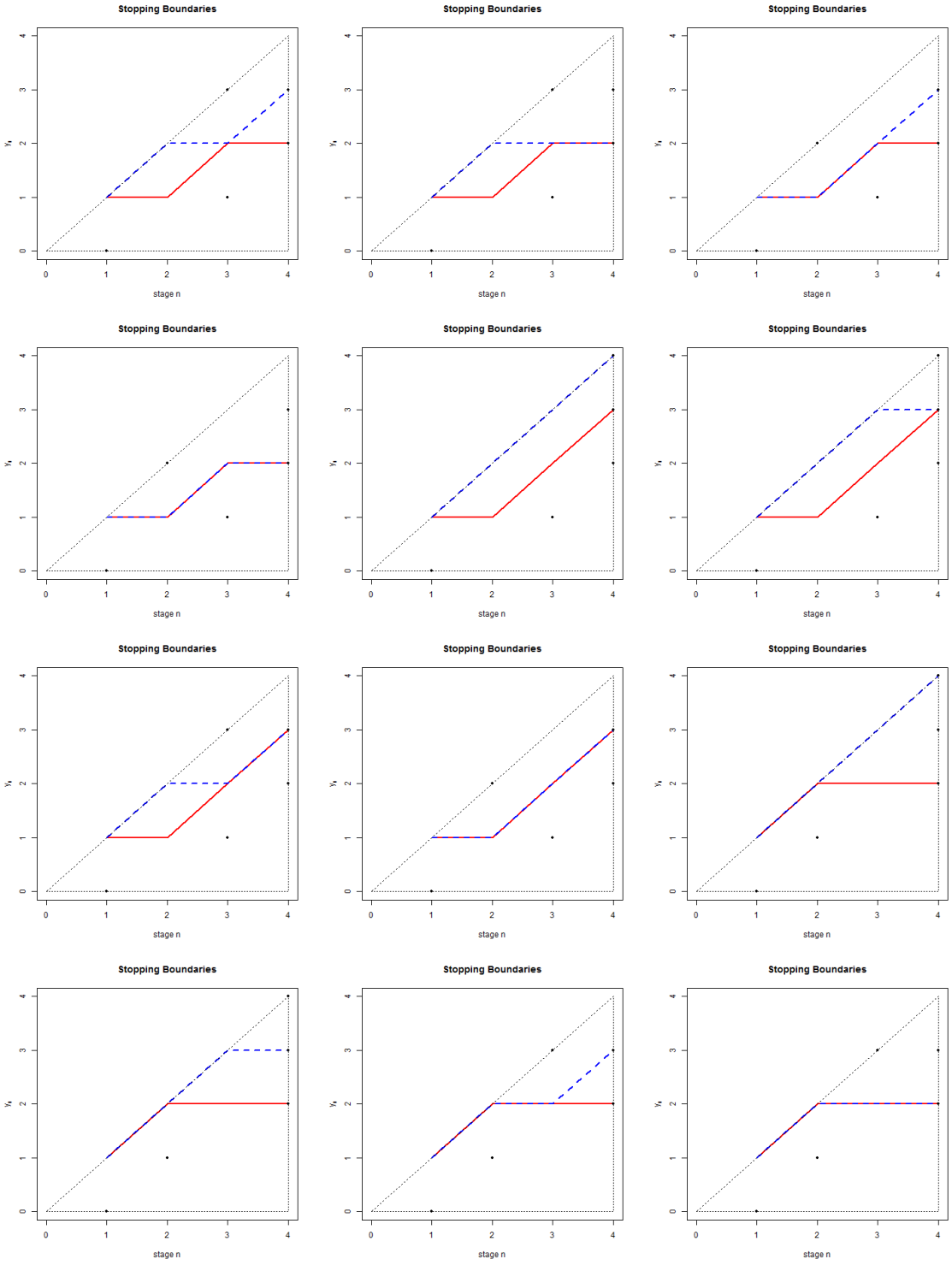

Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n

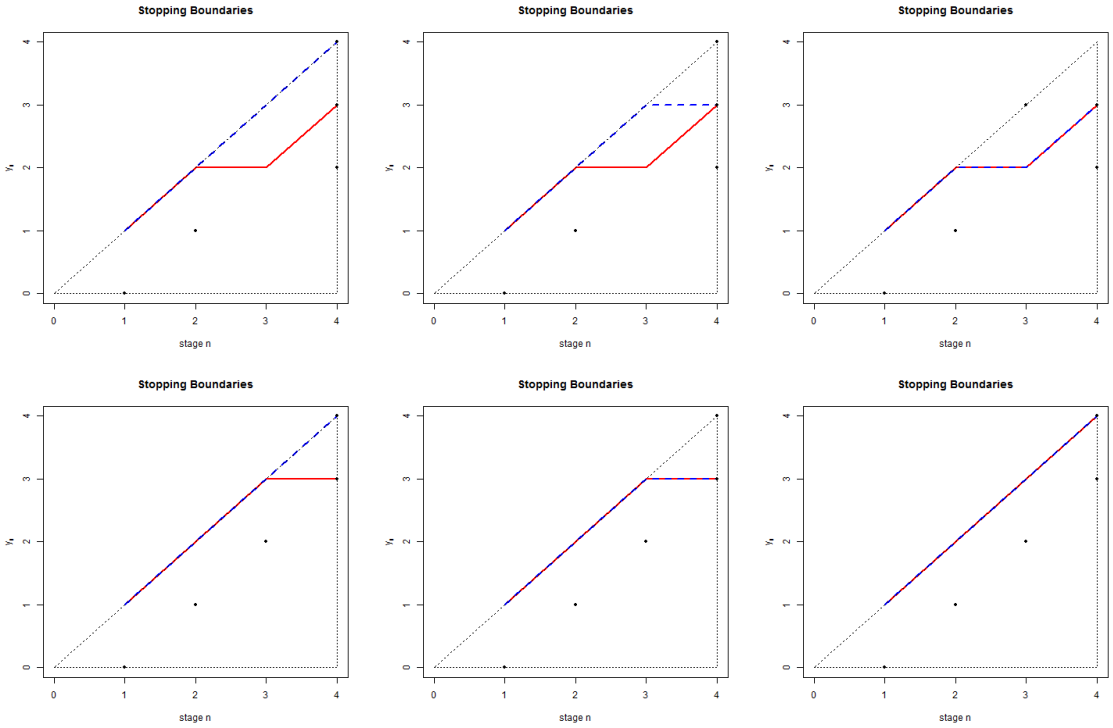

Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n


Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n


Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n


Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n


Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n


Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n
Stopping Boundaries
stage n

A.2 R Code

A.2.1 `check.step()`

Purpose of Function:
Check the step size of the boundaries to ensure that the step sizes are non-decreasing and are at most one.

Input:

`test`	Boundaries which are to be tested.

Output:

`flag`	Value of 1 indicates that the set of boundaries does not meet the step size requirement. Value of 0 indicates that the set of boundaries does meet the step size requirement.

Function:

```
check.step<-function(test)
{
i<-1
flag<-0

while(flag==0 && i < length(test))
{
      if((test[i+1]-test[i])>1 | (test[i+1]-test[i])<0)
            flag<-1
      i<-i+1
}
flag
}
```

A.2.2 `create.bnds()`

Purpose of Function:
Create all possible sets of lower and upper bounds (for a specified n_T) which either begin at zero or one, have steps that are non-decreasing and of size at most one, and (possibly) touch one another but do not cross.

Input:

`nT` Value of the maximum allowable sample size.

Output:
List containing the following matrices, rows of which correspond to one another:

`lbnd.mat` Matrix of all possible lower boundaries for specified n_T.

`ubnd.mat` Matrix of all possible upper boundaries for specified n_T.

Function:

```
create.bnds<-function(nT)
{
### A matrix to hold the upper and lower bounds
bnd.mat<-matrix(0,nrow=1,ncol=nT)

### Bnd.now begins as (0,0,0,...,0)
bnd.now<-rep(0,nT)

### Designates that bnd.now is no more than (1,2,3,...,nT)
while(sum(bnd.now==(1:nT)) != nT)

{
flag<-0
i<-0

while(flag==0)
{
bnd.new<-bnd.now
bnd.new[nT-i]<-bnd.new[nT-i]+1   ### Adds 1 to the nT-i column

if(check.step(bnd.new)==0)
{
      ### Combines bounds that are non-decreasing with step- size
            of at most 1
      bnd.mat<-rbind(bnd.mat,bnd.new)
      bnd.now<-bnd.new
      flag<-1
} # end of if(check.step(bnd.new)==0)
i<-i+1
} # end of while(flag==0)
} # end of while(sum(bnd.now==(1:nT)) != nT)

### Bnd.now begins as (1,2,3,..,nT)
bnd.now<-(1:nT)
```

```
### Designates that bnd.now is no less than (0,0,0,...,0)
while(sum(bnd.now) !=0)
{
flag<-0
i<-0

while(flag==0)
{
bnd.new<-bnd.now
bnd.new[nT-i]<-bnd.new[nT-i]-1   ### Subtracts 1 from the nT-i
      column

if(check.step(bnd.new)==0)
{
      ### Combines bounds that are non-decreasing with step-size
            of at most 1
      bnd.mat<-rbind(bnd.mat,bnd.new)
      bnd.now<-bnd.new
      flag<-1
} # end of if(check.step(bnd.new)==0)

i<-i+1

} # end of while(flag==0)
} # end of while(sum(bnd.now) != 0)



### Combines all bounds found by adding/subtracting 1 to column
      nT-i and deletes duplicates
bnd.list<-bnd.mat[!duplicated(bnd.mat),]
row.names(bnd.list)<-1:nrow(bnd.list)

lbnd.mat<-matrix(NA,ncol=nT)
ubnd.mat<-matrix(NA,ncol=nT)

for(i in 1:nrow(bnd.list))
{
      for(j in 1:nrow(bnd.list))

      {
            if(sum(bnd.list[i,]<=bnd.list[j,])==nT)
            {
                  ### Creates matrix of lower bounds
                  lbnd.mat<-rbind(lbnd.mat,bnd.list[i,])

                  ### Creates matric of upper bounds
                  ubnd.mat<-rbind(ubnd.mat,bnd.list[j,])
            } # end of if(sum(bnd.list[i,]<=bnd.list[j,])==nT)
      } # end of for(j in 1:nrow(bnd.list))
} # end of for(i in 1:nrow(bnd.list))
```

```
lbnd.mat<-lbnd.mat[-1,]
ubnd.mat<-ubnd.mat[-1,]

out<-list(lbnd.mat, ubnd.mat)
names(out)<-c("lbnd.mat","ubnd.mat")
out
}
```

A.2.3 `alpha.bounded()`

Purpose of Function:
Reduce the list of all possible sets of lower and upper bounds (for a specified n_T) such that the Type I error rate, α, is preserved.

Input:

`nT`	Value of the maximum allowable sample size.
`lbnd.mat`	Matrix of lower boundaries to be tested.
`ubnd.mat`	Matrix of upper boundaries to be tested.
`p0`	Value of null proportion to be tested.
`target.a`	Value of target Type I error rate. Default value is "0.05".
`dist`	Distribution under which to test the boundaries. Default value is "binom".

Output:
List containing the following matrices, rows of which correspond to one another:

`lbnds.a.bounded`	Matrix of lower boundaries which preserve α.
`ubnds.a.bounded`	Matrix of upper boundaries which preserve α.

Function:

```
alpha.bounded<-
function(nT,lbnd.mat,ubnd.mat,p0,target.a=0.05,dist="binom")
{

numbounded<-0
alphabounded<-rep(NA,nrow(lbnd.mat))
lbnds.bounded<-matrix(NA,nrow=1,ncol=nT)
ubnds.bounded<-matrix(NA,nrow=1,ncol=nT)

#### the following will be done for all possible bounds for nT
for(j in 1:nrow(lbnd.mat))
{
#### run seqbin for current set of bounds
lbnd.now<-lbnd.mat[j,]
ubnd.now<-ubnd.mat[j,]
out<-seqbin.nT(nT,lbnd.now,ubnd.now,plotit=F,dist=dist)
outcomes<-out$outcomes
probs<-out$probs

trunc<-outcomes[,1]==nT
lower<-rep(NA,nrow(outcomes))
upper<-rep(NA,nrow(outcomes))

#### determine which outcomes are in the rejection/acceptance
     regions and which are at the truncation
for(i in 1:nrow(outcomes))
{
```

```
      lower[i]<-outcomes[i,2] < lbnd.now[outcomes[i,1]]
      upper[i]<-outcomes[i,2] > ubnd.now[outcomes[i,1]]
      if(lower[i]==T)
           trunc[i]<-F
      if(upper[i]==T)
           trunc[i]<-F
} # end of for(i in 1:nrow(outcomes))

lowlength<-sum(lower)
trlength=sum(trunc)
uplength=sum(upper)

#### use values in rejection region (if they exist) to calculate
      alpha
if(uplength>0)
      alpha<-sum(probs[nrow(outcomes):(nrow(outcomes)-
      uplength+1),(p0*1000+1)]) else alpha<-0

#### determine if alpha is bounded
if(alpha>target.a)
      alphabounded[j]<-F else alphabounded[j]<-T

#### save the bounds for which alpha is bounded
if(alphabounded[j]==T)
{
      lbnds.bounded<-rbind(lbnds.bounded,lbnd.mat[j,])
      ubnds.bounded<-rbind(ubnds.bounded,ubnd.mat[j,])
      numbounded<-numbounded+1
} # end of if(alphabounded[j]==T)
} # end of for(j in 1:nrow(lbnd.mat))

#### return the list of boundaries for which alpha is bounded
out=list(lbnds.bounded[-1,],ubnds.bounded[-1,])
names(out)<-c("lbnds.a.bounded", "ubnds.a.bounded")
out
} # end of alpha.bounded
```

A.2.4 `errors.bounded()`

Purpose of Function:
Reduce a list sets of lower and upper bounds (for a specified n_T) such that not only the Type I error rate, α, is preserved, but also the Type II error rate, β.

Input:

`nT`	Value of the maximum allowable sample size.
`lbnd.mat`	Matrix of lower boundaries to be tested.
`ubnd.mat`	Matrix of upper boundaries to be tested.
`p0`	Value of null proportion to be tested.
`p1`	Value of alternative proportion to be tested.
`target.a`	Value of target Type I error rate. Default value is "0.05".
`target.b`	Value of target Type I error rate. Default value is "0.10".
`dist`	Distribution under which to test the boundaries. Default value is "binom".

Output:
List containing the following objects (rows of which correspond):

`lbnds`	Matrix of possible lower boundaries with both α and β bounded.
`ubnds`	Matrix of possible upper boundaries with both α and β bounded.
`alphas`	Vector containing α for each set of boundaries.
`betas`	Vector containing β for each set of boundaries.
`maxEn`	Vector containing max{E(n)} for each set of boundaries.

Function:

```
errors.bounded<-
function(nT,lbnd.mat,ubnd.mat,p0,p1,target.a=0.05,target.b=0.10,d
ist="binom")
{
numbounded<-nrow(lbnd.mat)

betas<-rep(NA,numbounded)
alphas<-rep(NA,numbounded)
bbounded<-rep(NA,numbounded)
maxEn<-rep(NA,numbounded)

#### the following will be done for all bounds for nT for which
      alpha is bounded
for(j in 1:numbounded)
{
lbnd.now<-lbnd.mat[j,]
ubnd.now<-ubnd.mat[j,]

#### run seqbin for current set of bounds
out<-seqbin.nT(nT,lbnd.now,ubnd.now,plotit=F,dist=dist)
En<-out$outcomes[,1]%*%out$probs
maxEn[j]<-max(En)    #### find the max[(E(n)] for this set of
```

```
                              boundaries
outcomes<-out$outcomes
probs<-out$probs

trunc<-outcomes[,1]==nT
lower<-rep(NA,nrow(outcomes))
upper<-rep(NA,nrow(outcomes))

#### determine which outcomes are in the rejection/acceptance
     regions and which are at the truncation
for(i in 1:nrow(outcomes))
{
     lower[i]<-outcomes[i,2] < lbnd.now[outcomes[i,1]]
     upper[i]<-outcomes[i,2] > ubnd.now[outcomes[i,1]]
     if(lower[i]==T) trunc[i]<-F
     if(upper[i]==T) trunc[i]<-F
} # end of for(i in 1:nrow(outcomes))

lowlength<-sum(lower)
trlength=sum(trunc)
uplength=sum(upper)

if(uplength>0)
{
     abounded<-T
     i<-nrow(outcomes)-uplength+1

     #### add points on truncation to rejection region, as long
     as alpha stays bounded
     while((abounded == T) && (i > lowlength))
     {
          if(sum(probs[nrow(outcomes):i,(p0*1000+1)])
          <= target.a)
          {
               alpha<-sum(probs[nrow(outcomes):i,(p0*1000+1)])
               i<-i-1
          } else abounded<-F          # end of
          if(sum(probs[nrow(outcomes):i,(p0*1000+1)])
          <= target.a)
     } # end of while((abounded == T) && (i > lowlength))
} # end of if(uplength>0)

if(uplength==0)
{
     abounded<-T
     i<-nrow(outcomes)

     #### add points on truncation to rejection region, as long
     as alpha stays bounded
     while((abounded == T) && (i > lowlength))
     {
```

```
            if(sum(probs[nrow(outcomes):i,(p0*1000+1)])
            <= target.a)
            {
                  alpha<-sum(probs[nrow(outcomes):i,(p0*1000+1)])
                  i<-i-1
            } else abounded<-F    # end of
            if(sum(probs[nrow(outcomes):i,(p0*1000+1)])
            <= target.a)
      } # end of while((abounded == T) && (i > lowlength))
} # end of if(uplength==0)

#### calculate beta using all point in acceptance region and
      remaining truncation points
beta<-sum(probs[1:i,(p1*1000+1)])
#### determine if beta is bounded; if yes, save results
if(beta<=target.b)
{
      bbounded[j]<-T
      betas[j]<-beta
      alphas[j]<-alpha
} else bbounded[j]<-F # end of if(beta<=target.b)
} # end of for(j in 1:numbounded)

#### save results for only the sets that have both alpha and beta
      bounded
if(sum(is.na(alphas))>0)
{
      lbnds.bounded.out<-lbnd.mat[-which(is.na(alphas)),]
      ubnds.bounded.out<-ubnd.mat[-which(is.na(alphas)),]
      betas.out<-betas[-which(is.na(alphas))]
      alphas.out<-alphas[-which(is.na(alphas))]
      maxEn.out<-maxEn[-which(is.na(alphas))]
} # end of if(sum(is.na(alphas))>0)

#### save results for all sets of bounds (because alpha and beta
      are bounded for all in this case)
if(sum(is.na(alphas))==0)
{
      lbnds.bounded.out<-lbnd.mat
      ubnds.bounded.out<-ubnd.mat
      betas.out<-betas
      alphas.out<-alphas
      maxEn.out<-maxEn
} # end of if(sum(is.na(alphas))==0)

#### return results for boundaries that have both error rates
      bounded
out=list(lbnds.bounded.out,ubnds.bounded.out,alphas.out,betas.out
,maxEn.out)
names(out)<-c("lbnds","ubnds","alphas","betas","maxEn")
out
} # end of errors.bounded
```

A.2.5 `sim.sprt()`

Purpose of Function:
Conduct a specified number of simulations on a given set of boundaries to determine the decision made in the test and the required sample size for each simulation.

Input:

`nT`	Value of the maximum allowable sample size.
`p`	Value of proportion of defectives in the population.
`p0`	Value of null to be tested.
`lbnd`	Lower boundary to be used.
`ubnd`	Upper boundary to be used.
`sim.size`	Value of the number of simulations to conduct. Default value is "10000".
`dist`	Distribution under which to simulate the data. Default value is "binom".
`pop.size`	Value of size of the population. Default value is "1000". This argument is only used when dist="hyper".

Output:
List containing the following objects:

`decision`	Vector of decisions made for each run of the simulation. Value = 1 if decision was made to fail to reject the null hypothesis, and value = 0 if decision was made to reject the null hypothesis.
`n.stop`	Vector of the required sample size to make a decision in the test for each run of the simulation.

Function:

```
sim.sprt<-
     function(nT,p,p0,lbnd,ubnd,sim.size=10000,dist="binom",
     pop.size=1000)
{
decision<-rep(NA,sim.size) ########## accept = 1; reject = 0
n.stop<-rep(NA,sim.size)

#### get seqbin results
out<-seqbin.nT(nT,lbnd,ubnd,plotit=F,dist=dist)
outcomes<-out$outcomes
probs<-out$probs

trunc<-outcomes[,1]==nT
lower<-rep(NA,nrow(outcomes))
upper<-rep(NA,nrow(outcomes))

#### determine which outcomes are in the rejection/acceptance
     regions and which are at the truncation
for(i in 1:nrow(outcomes))
```

```
{
      lower[i]<-outcomes[i,2] < lbnd[outcomes[i,1]]
      upper[i]<-outcomes[i,2] > ubnd[outcomes[i,1]]
      if(lower[i]==T)
            trunc[i]<-F
      if(upper[i]==T)
            trunc[i]<-F
} # end of for(i in 1:nrow(outcomes))

lowlength<-sum(lower)
trlength=sum(trunc)
uplength=sum(upper)

#### use values in rejection region (if they exist) to calculate
      alpha
if(uplength>0)
{
      abounded<-T
      i<-nrow(outcomes)-uplength+1

      #### add points on truncation to rejection region, as long
           as alpha stays bounded
      while((abounded == T) && (i > lowlength))
      {
            if(sum(probs[nrow(outcomes):i,(p0*1000+1)]) <= 0.05)
            {
                  alpha<-sum(probs[nrow(outcomes):i,(p0*1000+1)])
                  i<-i-1
            } else abounded<-F
            # end of if(sum(probs[nrow(outcomes):i,(p0*1000+1)])
            <= target.a)

      } # end of while((abounded == T) && (i > lowlength))

} # end of if(uplength>0)

if(uplength==0)
{
      abounded<-T
      i<-nrow(outcomes)

      #### add points on truncation to rejection region, as long
           as alpha stays bounded
      while((abounded == T) && (i > lowlength))
      {
            if(sum(probs[nrow(outcomes):i,(p0*1000+1)]) <= 0.05)
            {
                  alpha<-sum(probs[nrow(outcomes):i,(p0*1000+1)])
                  i<-i-1
            } else abounded<-F
            # end of if(sum(probs[nrow(outcomes):i,(p0*1000+1)])
            <= target.a)
```

```
      } # end of while((abounded == T) && (i > lowlength))
} # end of if(uplength==0)

for(j in 1:sim.size)
{
      if(dist=="binom")
      {
            #### create sample data with specified probability of
            success
            x<-cumsum(rbinom(nT,1,p))  #### cumsum calculates "yn"
      }     # end of if(dist=="binom")
      if(dist=="hyper")
      {
            #### create a "population" with the correct proportion
            of defectives/nondefectives
            population<-c(rep(1,p*pop.size),rep(0,(1-p)*pop.size))
            #### randomly sample from the population without
            replacement (making it hypergeometric)
            x<-cumsum(sample(population,nT,replace=F))
            #### cumsum calculates "yn"
      } # end of if(dist=="hyper")

      done<-0
      k<-1

      #### continue "sampling" until a decision is made
      while(done==0 && k<nT)
      {
            #### determine if in the acceptance region
            if(x[k] < lbnd[k])
            {
                  done<-1
                  n.stop[j]<-k
                  decision[j]<-1
            } # end of if(x[k] < lbnd[k])

            #### determine if in the rejection region
            if(x[k] > ubnd[k])
            {
                  done<-1
                  n.stop[j]<-k
                  decision[j]<-0
            } # end of if(x[k] > ubnd[k])
            k<-k+1
      } # end of while(done==0 && k<nT)


      ##### make a decision if sampling reached maximum sample
      size
      if(is.na(n.stop[j]))
      {
            n.stop[j]<-nT
```

```
            if(x[nT]>=outcomes[i+1,2]) decision[j]<-0
            if(x[nT]<outcomes[i+1,2]) decision[j]<-1
      } # end of if(is.na(n.stop[j]))
} # end of for(j in 1:sim.size)


#### return decision and needed sample size for all sim.size
simulations
out<-list(decision,n.stop)
names(out)<-c("decision","n.stop")
out

} # end of sim.sprt
```

A.2.6 `Wald.bnd()`

Purpose of Function:
Calculate the Wald boundaries in the binomial case.

Input:

`p0`	Value of null proportion.
`p1`	Value of alternative proportion.
`alpha`	Value of maximum allowable Type I error rate, α. Default value is "0.05".
`beta`	Value of maximum allowable Type II error rate, β. Default value is "0.10".

Output:
List containing the following objects:

`l.int`	Value of the intercept for the lower boundary.
`u.int`	Value of the intercept for the upper boundary.
`slope`	Value of the slope for the parallel boundaries.

Function:

```
Wald.bnd<-function(p0, p1, alpha=0.05, beta=0.10)
{
A<-(1-beta)/alpha
B<-beta/(1-alpha)
#### slope for the parallel boundaries
slope<--log((1-p1)/(1-p0))/log((p1/p0)*((1-p0)/(1-p1)))
#### intercept for lower boundary
l.int<-log(B)/log((p1/p0)*((1-p0)/(1-p1)))
#### intercept for upper boundary
u.int<-log(A)/log((p1/p0)*((1-p0)/(1-p1)))
out<-c(l.int, u.int, slope)
names(out)<-c("l.int", "u.int", "slope")
out
}
```

A.2.7 `Wald.step.bnd()`

Purpose of Function:
Convert the Wald boundaries to a step boundary.

Input:

`l.int`	Value of the intercept for the lower boundary to be converted.
`u.int`	Value of the intercept for the upper boundary to be converted.
`slope`	Value of the slope for the parallel boundaries to be converted.
`nT`	Value of the maximum allowable sample size.

Output:
List containing the following objects:

`lbnd`	The converted Wald lower bound as a step boundary.
`ubnd`	The converted Wald upper bound as a step boundary.

Function:

```
Wald.step.bnd<-function(l.int, u.int, slope,nT)
{
n<-1:nT
lbnd<-ceiling(l.int+slope*n)
ubnd<-floor(u.int+slope*n)
out<-list(lbnd,ubnd)
names(out)<-c("lbnd", "ubnd")
out
}
```

REFERENCES

[1] Iglewicz, Boris (1973). An Alternative Measure of the Efficiency of Sequential Testing Procedures. *Technometrics*. 15(3):571-574.

[2] Ignatova, Iliana, Roland Deutsch, and Don Edwards (2011). Closed sequential and multistage inference with binary response. Submitted for publication.

[3] Mingoti, Sueli (2003). A note on the sample size required in sequential tests for the generalized binomial distribution. *Journal of Applied Statistics*. 30(8):873-879.

[4] McWilliams, Thomas P. (2004). The Design of Truncated Sequential Test Plans Based on Attributes Data. *Communications in Statistics: Simulation and Computation*. 33(3):843-849.

[5] R Development Core Team (2011). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.

[6] Wald, Abraham (1947). *Sequential Analysis*. New York: John Wiley and Sons.