THE EFFECT OF SAMPLE SIZE RE-ESTIMATION ON TYPE I ERROR RATES WHEN COMPARING TWO BINOMIAL PROPORTIONS

by

DANNI CONG

B.S., Harbin Institute of Technology, 2007
M.S., University of Science and Technology of China, 2010
M.S., Kansas State University, 2015

A REPORT

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Statistics
College of Arts and Sciences

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2016

Approved by:

Major Professor
Dr. Christopher I. Vahl

Abstract

Estimation of sample size is an important and critical procedure in the design of clinical trials. A trial with inadequate sample size may not produce a statistically significant result. On the other hand, having an unnecessarily large sample size will definitely increase the expenditure of resources and may cause a potential ethical problem due to the exposure of unnecessary number of human subjects to an inferior treatment. A poor estimate of the necessary sample size is often due to the limited information at the planning stage. Hence, the adjustment of the sample size mid-trial has become a popular strategy recently. In this work, we introduce two methods for sample size re-estimation for trials with a binary endpoint utilizing the interim information collected from the trial: a blinded method and a partially unblinded method. The blinded method recalculates the sample size based on the first stage's overall event proportion, while the partially unblinded method performs the calculation based only on the control event proportion from the first stage. We performed simulation studies with different combinations of expected proportions based on fixed ratios of response rates. In this study, equal sample size per group was considered. The study shows that for both methods, the type I error rates were preserved satisfactorily.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my deepest appreciation to all those who provided me the possibility to complete this report.

I would like to express my sincere gratitude to my advisor Dr. Christopher I. Vahl, for the continuous support of my master study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this report. I could not have imagined having a better advisor and mentor for my master study.

I would like to thank my committee members, Dr. Haiyan Wang and Dr. Cen Wu for their support and guidance. I would also like to thank Ms. Jo Ann Blackburn and Ms. Bonnie Messmer, for all the help in my school life.

I would like to thank all of my friends for all the fun we have had in the last two years, and for incenting me to strive towards my goal.

A special thanks to my families, to my dearest parents, my beloved husband, and my lovely little girl. Thank you all for the sacrifices that you’ve made on my behalf. Your support and your prayer for me were what sustained me thus far.

Chapter 1 - Introduction

1.1 The importance of sample size

Sample size estimation plays a crucial role for a successful clinical trial. Conducting a clinical trial usually consumes a great deal of time and resources. Hence it is of great importance that the design of the trial should give a good chance of successfully detecting a treatment effect if one exists. Generally, the larger the number of patients involved in a trial, the higher the chance to identify a significant difference between treatments. However, if the study involves unnecessarily large number of participants, it will not only increase the research budget, but also cause potential ethical problems. Therefore, it is often necessary to take an interim look at the data to check how good the sample size estimation at the planning stage was, and whether the study can be terminated early or more patients need to be recruited.

1.2 Blinding & unblinding

For studies with a continuous endpoints, the sample size calculation typically requires the knowledge of the treatment effects, which is the difference between the different group means, and the nuisance parameter, which is the standard deviation or variance. For a study with binary endpoints, the sample size calculation needs information on the treatment effect, which is the difference between the proportions of events (such as the disappearance of tumor) that have occurred in different treatment groups, as well as either the proportion of events in the control group or the proportion of events among all participants without regard to treatment group (Proschan, 2005). During an ongoing trial, investigators may want to re-estimate the sample size based on the information from interim data and adjust it accordingly if necessary. Here, we consider the case of comparing two treatments based on a binary response variable.

When performing sample size re-estimation, people often take an interim look at the data through three different approaches. In terms of the degree of blinding involved, the three methods can be termed: ‘complete unblinding’, ‘partial unblinding’, and ‘no unblinding’. The ‘partial unblinding’ means that researchers know only the participants’ group membership (group 1 or group 2, say), but don’t know the treatments corresponding to these groups (Gould, 2001). The ‘no unblinding’ method ignores that the participants come from different treatment groups and looks at the overall event proportions.

Sample size re-estimation in clinical trials has a long history which can date back to Stein (Stein, 1945), who developed a method about the t-test that can be used in both one-sample and two-sample trials whose power is independent of variance. There were several approaches of sample size re-estimation strategies that have been developed in the literature with different two-stage schemes. Gould & Shih (1992) and Wittes & Brittain (1990) have discussed methods of blinded sample size re-estimation which assumed that the real treatment effect is not exposed to the decision makers who do the sample size re-estimation. They found that the blinded methods were reasonably comparable in performance with methods that use unblinding estimates. In a simulation study that investigated the performance of both blinded and unblinded sample size re-estimation methods, Wittes, Schabenberger, Zucker, Brittain, & Proschan (1999) observed some slight violations of the type I error when the sample size was relatively small. Others have shown that when the sample size is re-calculated with an unblinded variance estimate, an inflation of the type I error rate may occur (Wittes & Brittain, 1990; Birkett & Day, 1994). Therefore, in our study, we consider only the ‘partial unblinding’ and ‘no unblinding’ methods when performing sample size re-estimation. We expect that the type I error rate can be preserved well since the blinding assessment does not provide information about true treatment effects.

1.3 Power & type I error rate

For a trial with a binary endpoint, the sample size depends not only on the information of treatment effects (i.e., a clinically relevant difference) and the overall event rate or event rate in control group, but also on the value of the significance level and power (Friede & Kieser, 2004).

In hypothesis testing, two mutually exclusive statements (H_0 and H_A) are evaluated in terms of a population parameter. There are four possible results for hypothesis testing (Table 1.1): H_0 is correctly not rejected when it is true, H_0 is wrongly rejected when it is actually true, H_0 is wrongly not rejected when it is actually false, and H_0 is correctly rejected when it is false.

Table 1.1 Possible results in hypothesis testing.

Hypothesis testing	H_0 is not rejected	H_0 is rejected
H_0 is true	Correct Probability = 1- α	Type I error Probability = α
H_0 is false	Type II error Probability = β	Correct Probability = 1- β

A type I error occurs when incorrectly rejecting a true H_0. The probability of committing this kind of error is α. We call α the significance level or type I error rate. A type II error occurs when failing to reject H_0 when it is false. The probability of committing a type II error is β. The power of a test is then the probability that it correctly rejects a false null hypothesis, i.e. the complement of a type II error (i.e. 1- β). A trial designer generally wants the power to be as large as reasonably possible.

1.4 Test of statistical superiority

In clinical studies, the randomized clinical trial (RCT) is generally considered as the best method to compare effects of therapies (Armitage, Berry, & Matthews, 2008). Often the aim of an RCT is to investigate whether a new therapy is superior to a control therapy (i.e. either an approved therapy or a placebo). In this situation, we refer to the study as a superiority trial. Let p_T be the proportion of successful outcomes for treatment group (i.e. the proportion of patients who are receiving the new intervention and get cured) and let p_C be the proportion of successful outcomes for control group (i.e. the proportion of patients who are receiving the control treatment and get cured). To test whether the effect of new treatment is superior to that of control, the following hypotheses are usually considered:

$$H_O: \ p_T \leq p_C + \delta$$

$$H_A: \ p_T > p_C + \delta.$$

Here δ ($\delta > 0$) is the superiority margin (i.e. the clinically meaningful difference between the two proportions). If H_O is rejected in favor of H_A, the new treatment is assumed to be clinically superior to the control. When $\delta = 0$, the hypotheses become:

$$H_O: \ p_T \leq p_C$$

$$H_A: \ p_T > p_C.$$

This is often referred to as a test of statistical superiority and the treatment is considered to be statistically superior to control when H_O is rejected.

Besides the superiority test, sometimes the investigators are also interested in the testing for difference (i.e. $p_T \neq p_C$), non-inferiority (i.e. p_T - p_C is greater than or equal to the non-inferiority margin), and equivalence (i.e. $| p_T - p_C|$ is a clinically unimportant). In our study, we will focus on superiority trials only.

1.5 Example

In this work, we perform the simulation studies with different combinations of expected proportions, based on fixed ratios of response rates. Now let's take one of those scenarios as an example to illustrate the research problem. We consider a trial to compare a new drug (test drug) to an approved standard drug (control drug) with respect to the proportion of patients who have been cured (or whose symptoms have improved) over a certain time period. The proportion expected for the control group (i.e. the proportion of patients who are taking the standard drug and get cured) is assumed to be 0.60. The goal of the sample-size estimation is to find the sample size needed to achieve 80% power to detect a 25% increase for the test group over control (i.e. p_T / p_C = 1.25). Based on these assumptions, the initial sample size was calculated to be 120/arm, or 240 patients in total. After 120 patients (60/arm, or half of the initial sample size) have been evaluated, we stop the trial and re-calculate the sample size using the accumulated information so far. If the initial sample size is large enough, the trial will continue to complete with the remaining planned number of patients. Otherwise, the sample size will be increased and the trial will continue until enough patients have been recruited to provide the required power (80%). The methods of sample size evaluation, the corresponding test statistics, and a simulation study will be discussed in detail in the following chapters.

Chapter 2 - Statistical Methods

2.1 Initial sample size estimation

We consider a clinical trial that compares two treatments based on a binary outcome. We denote the proportions of treatment success by p_T for the treatment group and p_C for the control group, respectively. At the planning stage, we estimate the initial sample size for the trial based on our assumptions. Conventionally, point estimates of p_T and p_C are assumed – here they are denoted as $p_T^*$and p_C^*, respectively. As before, let's assume that the proportion of successes for the control group is 0.60 (i.e. p_C^*=0.60) and that we desire 80% power to detect a 25% increase for the test group over control (i.e. $p_T^* / p_C^* = 1.25$). Therefore, the expected proportion of events for the test group is 0.75 (i.e. p_T^*=0.75). The initial sample size depends on several quantities: the overall proportion of successes disregarding treatment groups, i.e., $p^* = (p_T^* + p_C^*)/2$, the difference between treatment groups (p_T^*- p_C^*), the type I error rate (e.g., $\alpha = 0.05$), and the desired power (e.g. $1- \beta = 0.80$). For a test of statistical superiority, the sample size (n) required for achieving $100(1- \beta)\%$ power is calculated by the following formula (c.f., Proschan, 2005):

$$n = \frac{\left(z_\alpha\sqrt{2p^*(1-p^*)} + z_\beta\sqrt{p_C{}^*(1-p_C{}^*) + p_T{}^*(1-p_T{}^*)}\right)^2}{(p_T{}^* - p_C{}^*)^2}. \qquad (1)$$

Based on the assumptions given above, we need 120 patients per arm (or 240 in total) to detect a 25% increase of the events rates in the test group, with the 80% power at 5% level of significance.

2.2 Two methods of sample size re-estimation

Investigators sometimes look at the interim results in the trial to see whether they need to change the sample size or not. In our study, the total initial sample size is $N = 2n$ (n patients in each treatment group). We will conduct an interim analysis when the outcome data are available from n patients ($n/2$ from each group), i.e. when the trial is halfway completed. Since unblinding the data may cause bias and hence inflate the type I error rate, we conduct two different methods for performing the sample size re-estimation: blinded method, and partially unblinded. The partially unblinded method uses only the observed event rates in the control group to re-calculate the sample size, while the blinded method looks at the observed overall event rate without regard to treatment group.

To illustrate these two methods, consider the example above used to explain the initial sample size estimation. It was initially estimated that 240 patients in total are needed for the trial. After the trial is halfway completed, we analysis the outcome data from the 120 patients (60 for each group). We denote the count of the successes up to this point (i.e. the number of patients cured) as x_{11} and x_{21} for test and control groups respectively. For the partially unblinded method, look at the control data only and suppose that $x_{21} = 29$ was observed. Hence the control proportion is:

$$p_C = \frac{x_{21}}{n/2} = \frac{29}{60} = 0.483$$

and $$p_T = 1.25 * p_C = 1.25 * 0.483 = 0.604$$

hence $$p = \frac{p_C + p_T}{2} = \frac{0.483 + 0.604}{2} = 0.544.$$

The formula for the new sample size is:

$$n^* = \frac{\left(z_\alpha\sqrt{2p(1-p)}+z_\beta\sqrt{p_C(1-p_C)+p_T(1-p_T)}\right)^2}{(p_T - p_C)^2}. \qquad (2)$$

Based on interim data, $n^* = 209$ per treatment arm.

For the blinded method, we look at the entire data via the overall event rate. Suppose the observed counts of the successes for test and control groups combined are $x_{11} + x_{21} = 63$, respectively. Hence the overall proportion is:

$$p = \frac{x_{11}+x_{21}}{n} = \frac{63}{120} = 0.525$$

hence $$p_C = 2 * \frac{p}{2.25} = 2 * \frac{0.525}{2.25} = 0.467$$

and $$p_T = 1.25 * p_C = 1.25 * 0.467 = 0.583.$$

Therefore, using equation (2), the new sample size is $n^* = 226$ per treatment arm.

If the initial sample size is large enough to provide the desired power (i.e. $n \geq n^*$), the trial will continue until all the planned number of patients (n) are recruited. Otherwise, if $n < n^*$, we will increase the sample size and the trial will continue until enough patients (n^*) have been recruited that the desired power is achieved. Therefore, the final new sample size for each arm of the trial is

$$n_{new} = Max(n, n^*).$$

We can see that compared to the originally planned 240 patients, the new sample sizes calculated by both methods have increased (418 for partially unblinded method and 452 for blinded method). In the simulation study, we will use the same calculation procedures to re-estimate the sample sizes for all the scenarios with different combinations of expected proportions.

2.3 Two - proportion Z-test

In this work, we want to test whether there is a superiority of the success rates of the test group against that of the control group, and the following hypotheses are considered:

$$H_O:\ p_T \leq p_C$$

$$H_A:\ p_T > p_C.$$

The test statistic for testing the null hypothesis is:

$$Z = \frac{p_T - p_C}{\sqrt{p * (1-p) * (\frac{1}{n_1} + \frac{1}{n_2})}} = \frac{p_T - p_C}{\sqrt{p * (1-p) * \frac{2}{n_{new}}}} \qquad (3)$$

where n_1 and n_2 are the sample sizes of test and control groups, respectively, with $n_1 = n_2 = n_{new}$, and p is the overall proportion of successes. If we denote the count of the successes (i.e. the number of patients cured) as x_1 and x_2, for test and control groups, respectively, we have:

$$p = \frac{x_1 + x_2}{n_1 + n_2} = \frac{x_1 + x_2}{2 * n_{new}}$$

Since the superiority test is a right-tailed Z-test, an extreme value on the right side of the sampling distribution would cause the rejection of the null hypothesis. At the 5% significance level ($\alpha = 0.05$), we reject the null hypothesis H_0 if $Z \geq Z_\alpha = Z_{0.05} = 1.645$, and we fail to reject H_0 when $Z < 1.645$ (Figure 2.1).

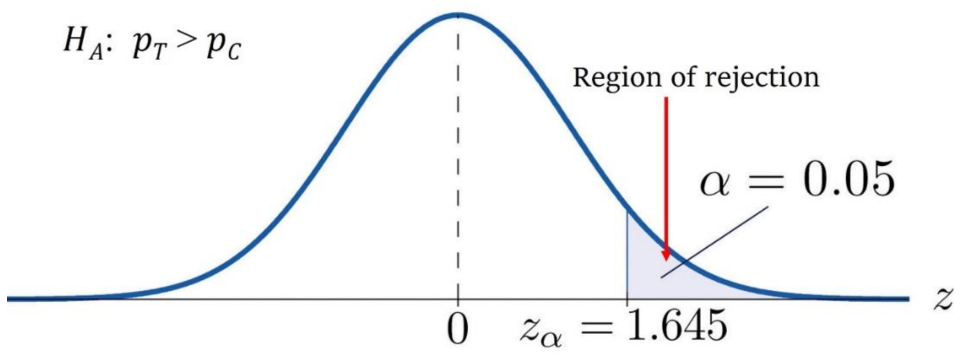

Figure 2.1 Rejection region of the superiority test at 5% significance level

Chapter 3 - Simulation Design

In this work, the simulation studies are conducted to generate data, calculate the sample size, and evaluate the type I error rate for the superiority test. The simulations were performed within SAS/STAT® software version 9.4 (SAS Institute, 2013).

Simulations were performed with different combinations of expected proportions and two different ratios of response rates given in Table 3.1.

Table 3.1 Scenarios for the simulation studies.

Scenario	Expected proportion of successes for control group, p_C^*	Expected proportion of successes for test group, p_T^*	p_T^* / p_C^*
1	0.4	0.5	1.25
2	0.6	0.75	1.25
3	0.7	0.875	1.25
4	0.4	0.46	1.15
5	0.6	0.69	1.15
6	0.7	0.805	1.15

Six scenarios were considered: 3 expected proportions of successes for the control group (p_C^* = 0.4, 0.6 and 0.7) combined with 2 ratios of response rates (p_T^* / p_C^* = 1.15 and 1.25). These assumptions are used for the calculation of the initial sample sizes. Within each scenario, we set the true success rate for the control group at p_C = 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, and 0.8, and each value was used to generate the binary data through the simulation process. The set of the combinations was chosen to encompass the range of values typically encountered in a real trial.

Since we have 96 settings for the simulation study (6 scenarios × 8 true p_C's × 2 methods), macros were created to efficiently perform the simulations with the specified setting of parameters. The complete SAS code for implementing the simulations is given in Appendix B.

For each selected setting, the simulation process is as follows:

1. Calculate the initial sample size (n) by Equation (2), based on the pre-specified expected effect: p_C^* and p_T^* / p_C^*.
2. Choose a proportion of n to determine the time for the interim analysis, in our study, we perform the sample size re-estimation when $n/2$ of the patients are recruited into the study.
3. Begin the trial by simulating the binary data using the half of the original planned sample size: x_{11}, x_{21} ~ Bin ($n/2$, p_C).
4. Use the interim data to re-calculate the sample size (n^*) by Equation (3), and use the maximum of n and n^* as the final new sample size (n_{new}).
5. Resume the trial to obtain the binary data for the remaining observations: x_{12} , x_{22} ~ Bin ($n_{\text{new}} - n/2$, p_C).
6. Calculate the Z-statistics (by Equation (1)) based on the simulated data and conduct the superiority test. The estimated type I error rate is:

$$Type\ I\ \widehat{error}\ rate = \frac{The\ number\ of\ rejections\ of\ H_0}{The\ number\ of\ simulation\ runs} \tag{4}$$

The simulation contains 5000 runs for each setting.

Chapter 4 - Simulation Results

Results from all simulations are given in Appendix A (Tables A.1 to A.6). From Table A.1 to A.6, we can see that for each selected setting of p_C^* and p_T^* / p_C^*, when the true response rates in control group (p_C) is much lower than anticipated (p_C^*), and hence the true overall success rate, p, is lower than p^*, the interim re-estimation yields an impractically large increase in the sample size; the mean of n_{new} is as large as 7 to 40 times the original planned sample size. In these cases, 100% of the sample sizes were increased. As the true p_C increased to no larger than p_C^*, the new sample size required to achieve the desired power is decreased, but was still larger than the original planned sample size (n). And when the true p_C is greater than p_C^*, the re-estimation procedure may call for a reduction of the sample size (results are not shown in the tables because we used the maximum of n and n^*) as the final sample size. This pattern is seen in the results from both methods.

The trends of the mean of the re-estimated sample sizes along with different levels of true p_C for 6 scenarios are shown in Figures 4.1 to Figure 4.6. Within each figure, the trend line in red is for the sample sizes that are re-estimated by the blinded method, and the trend line in blue is for the sample sizes that are re-estimated by the partially unblinded method. The dash line shows the initial sample size (n) that was calculated at the design stage.

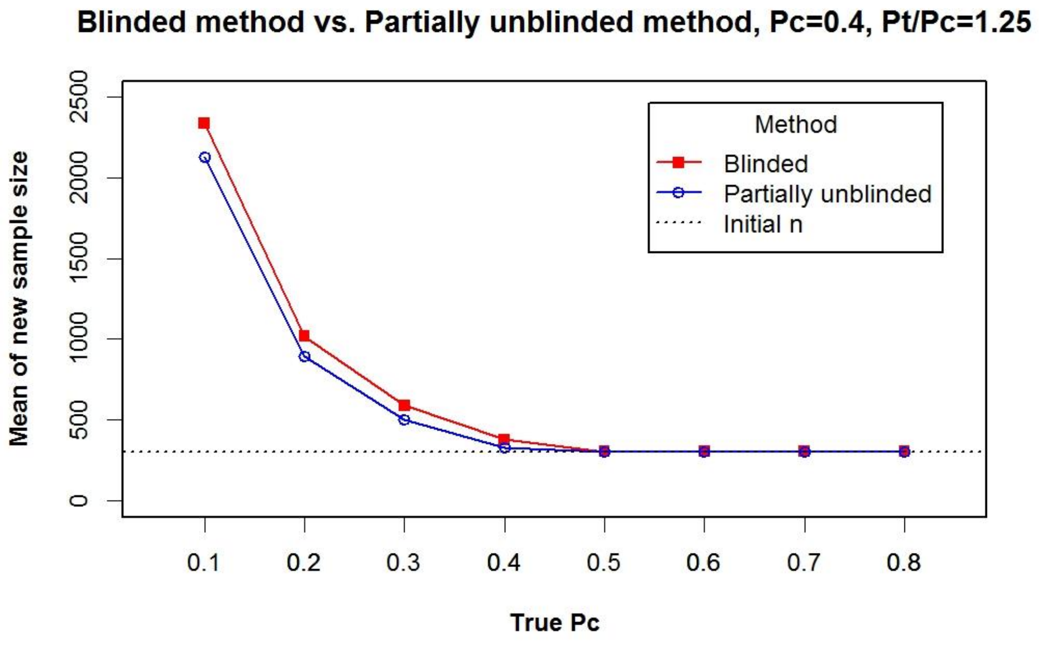

Figure 4.1 The mean of the new sample size for p_C^* = 0.4 and p_T^* / p_C^* = 1.25

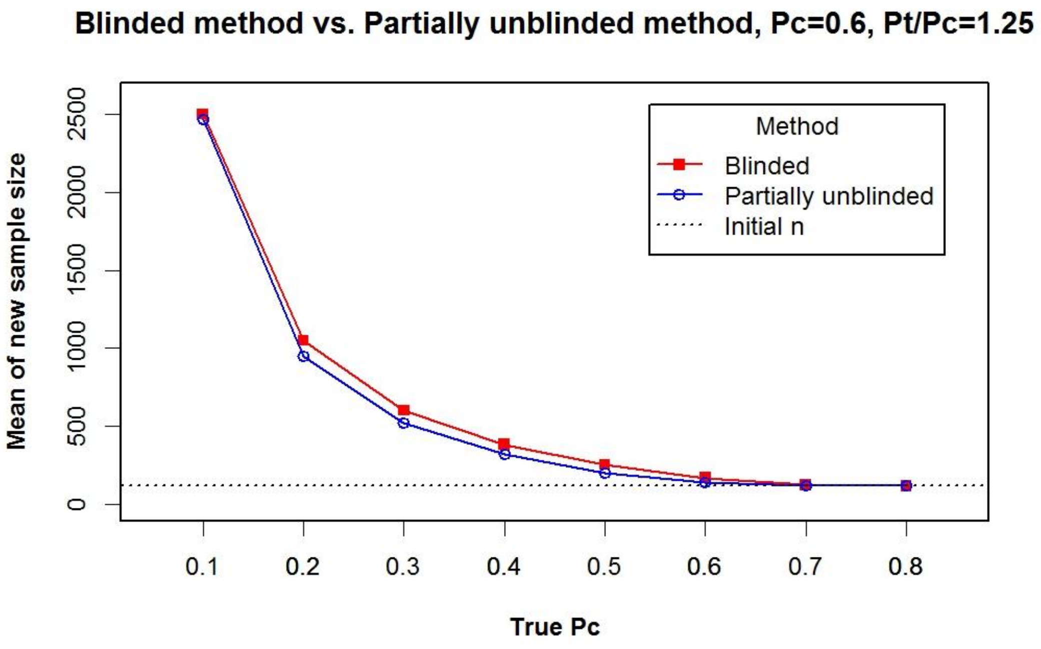

Figure 4.2 The mean of the new sample size for p_C^* = 0.6 and p_T^* / p_C^* = 1.25

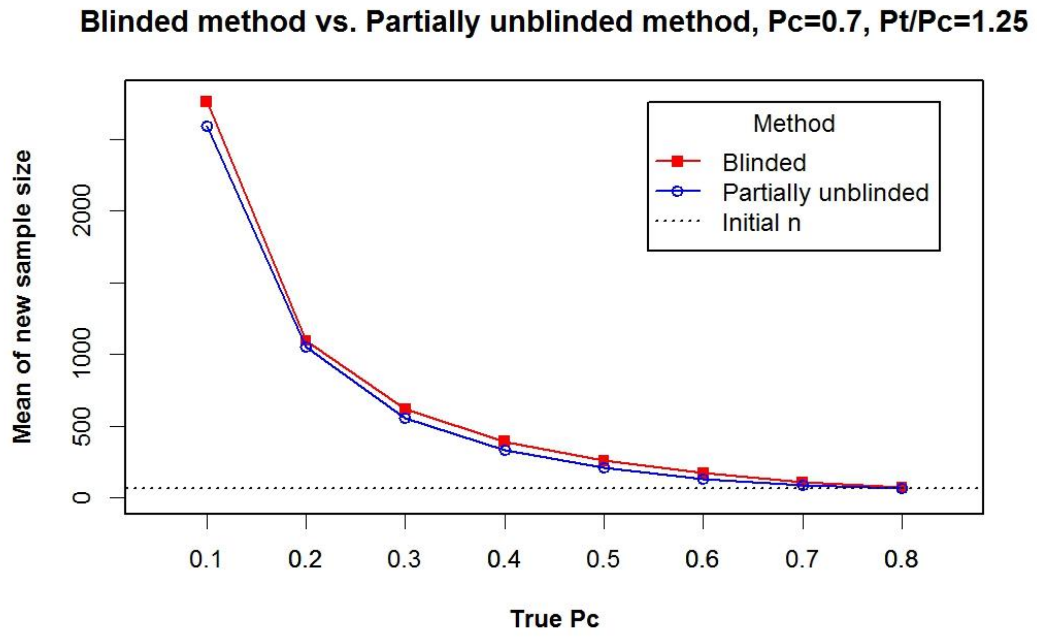

Figure 4.3 The mean of the new sample size for p_C^* = 0.7 and p_T^* / p_C^* = 1.25

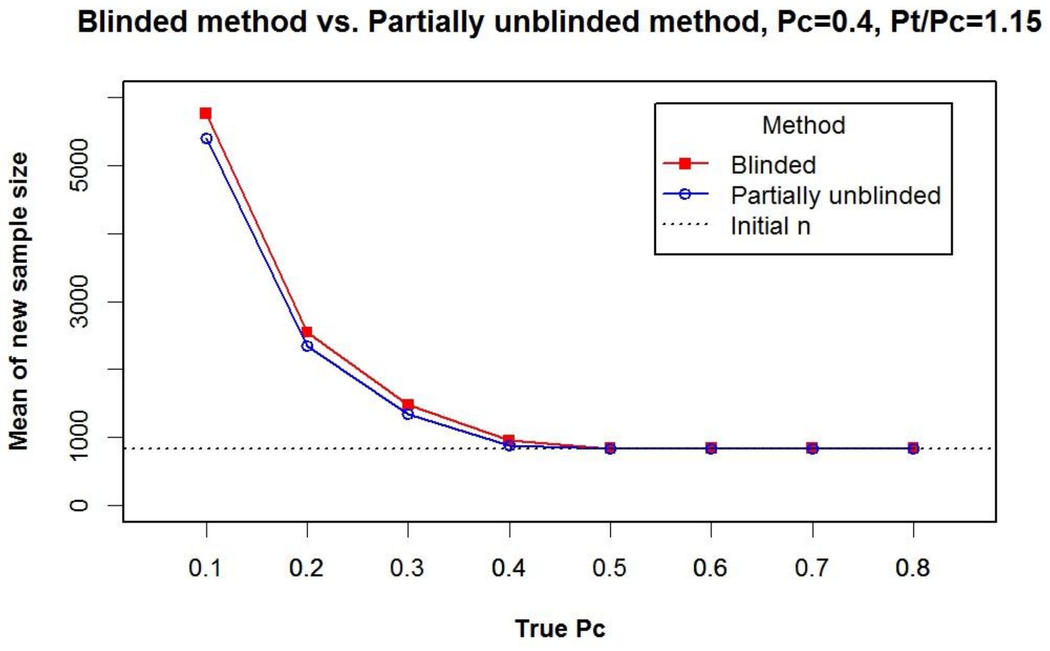

Figure 4.4 The mean of the new sample size for p_C^* = 0.4 and p_T^* / p_C^* = 1.15

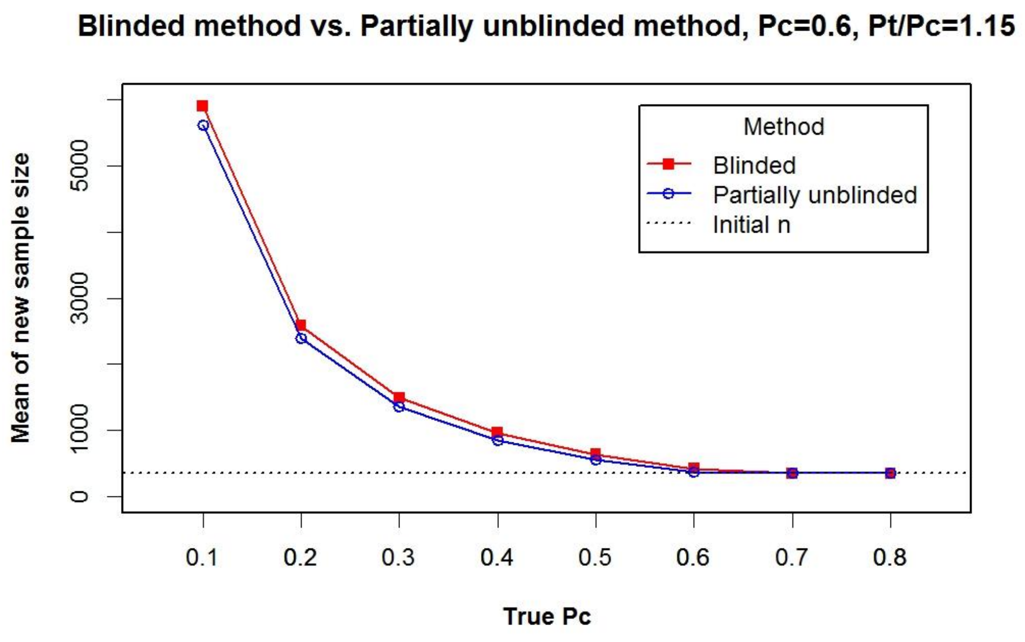

Figure 4.5 The mean of the new sample size for p_C^* = 0.6 and p_T^* / p_C^* = 1.15

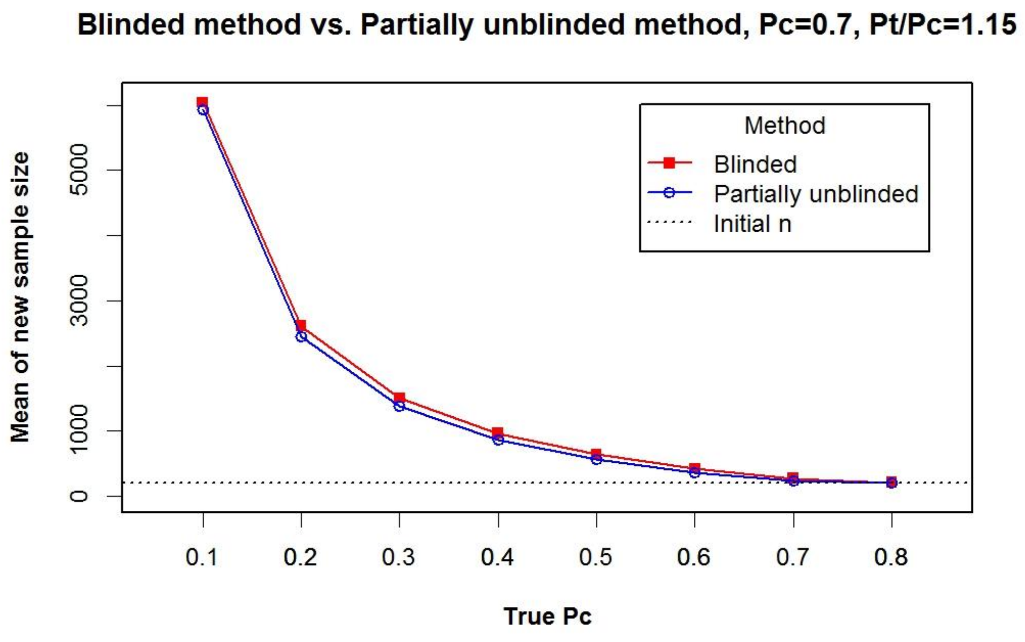

Figure 4.6 The mean of the new sample size for p_C^* = 0.7 and p_T^* / p_C^* = 1.15

From Figures 4.1 to 4.6, we can see that the mean of the new sample size estimated by the blinded method is always higher than the mean of the new sample size estimated by the partially unblinded method. This may be due to the fact that when we use the blinded method, we look at only the overall data, and in most of the cases, it results a lower overall success rate (the SAS output is not shown). The final sample size was reported to be negatively correlated with the overall success rate (Gould, 1992). Hence the blinded method yields a larger sample size than the partially unblinded method.

Now consider the effect of the sample size re-estimation process on the type I error rate where the true type I error rates were estimated using Equation (4). The margin of error (MoE) was computed for 95% confidence limits of the desired type I error rate (0.05) and the values 0.05 $\pm$ MoE were added to the figures as dashed lines. Estimated type I error rates within the dashed lines are said to hold their nominal level. Estimated type I error rates below both dashed lines are said to be conservative (which decreases power) and those above both dashed lines are said to be liberal (i.e. have inflated type I error rates). The results of the type I error rates are shown in the Tables A.1 to A.6, and also in Figures 4.7 to 4.12. Figures 4.7 to 4.12 show the observed type I error rates for different settings. Within each figure, the trend line in red is the observed type I error rates for the blinded method, and the trend line in blue is the observed type I error rates for the partially unblinded method. The dash lines are confidence limits described above.

The simulation results show that for 88 of the 96 parameter the observed type I error rates were contained within the 95% confidence limits around 0.05. The other 8 are very slightly higher or lower than the nominal level. Hence, it appears that the sample size re-estimation in the interim stage has no meaningful effect on the type I error rate. The figures show also show that there is no difference between the type I error rates of two re-estimation methods.

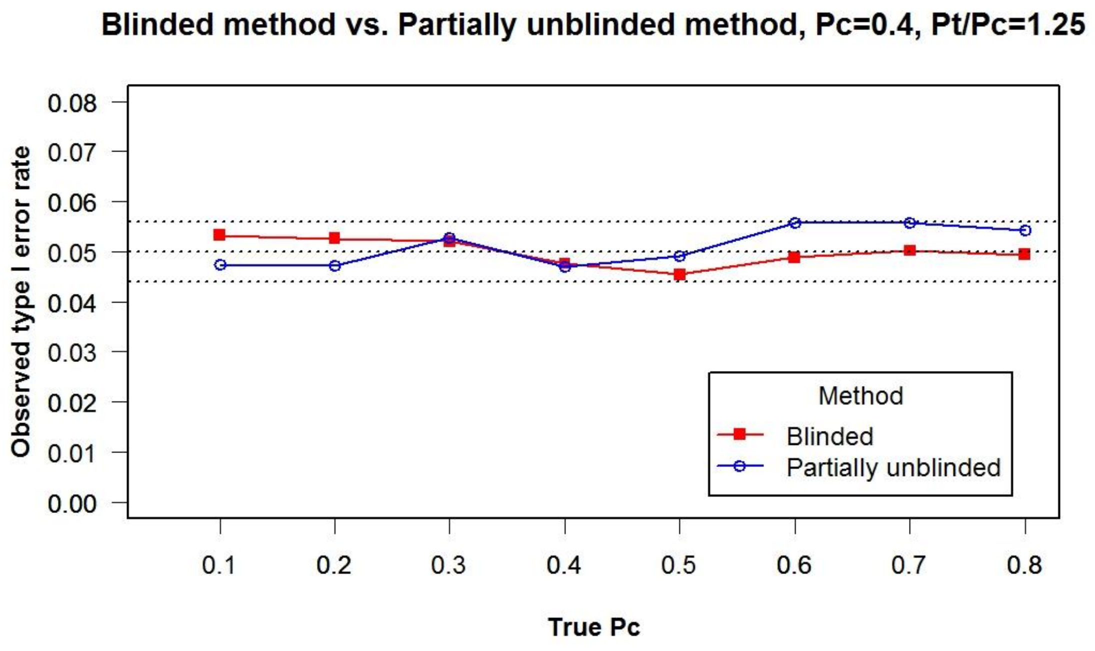

Figure 4.7 The observed type I error rate for p_C^* = 0.4 and p_T^* / p_C^* = 1.25

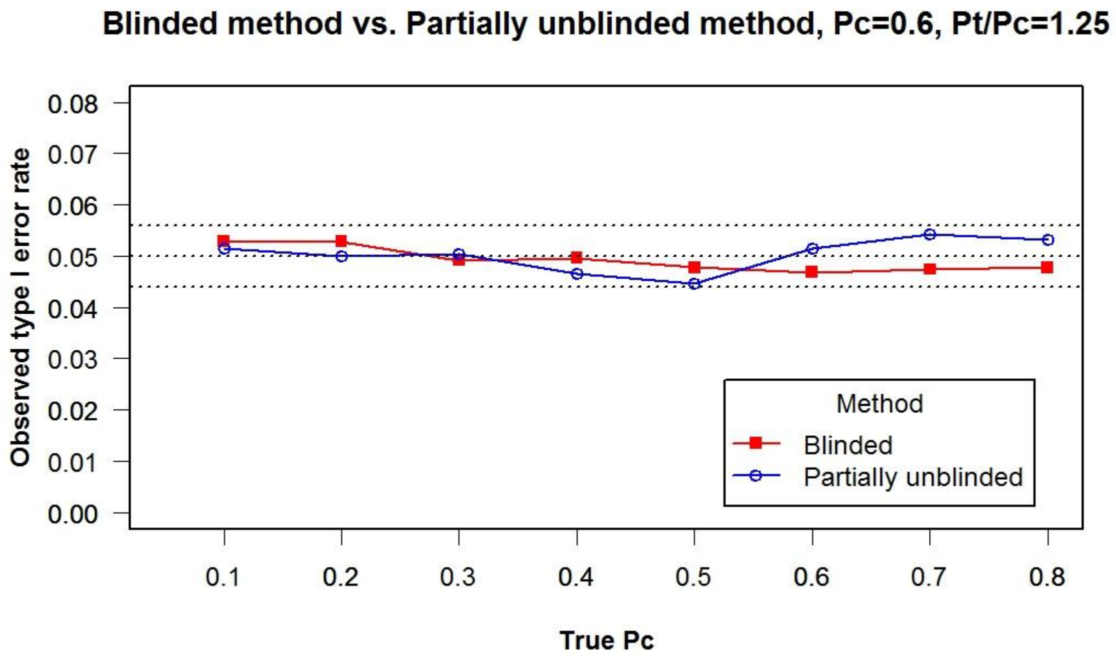

Figure 4.8 The observed type I error rate for p_C^* = 0.6 and p_T^* / p_C^* = 1.25

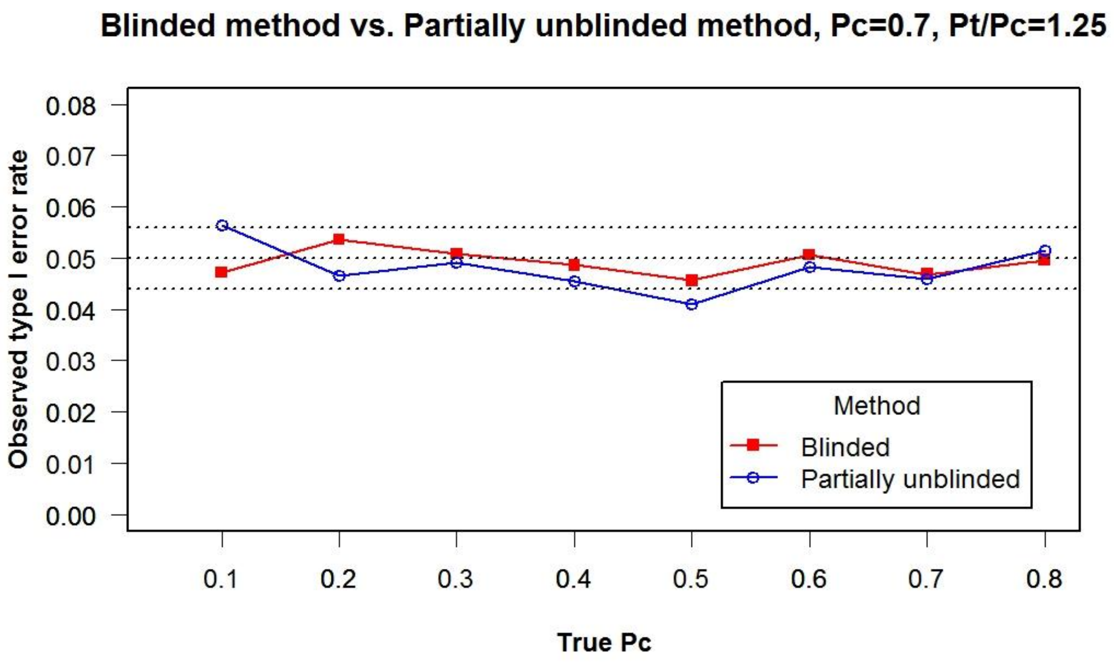

Figure 4.9 The observed type I error rate for p_C^* = 0.7 and p_T^* / p_C^* = 1.25

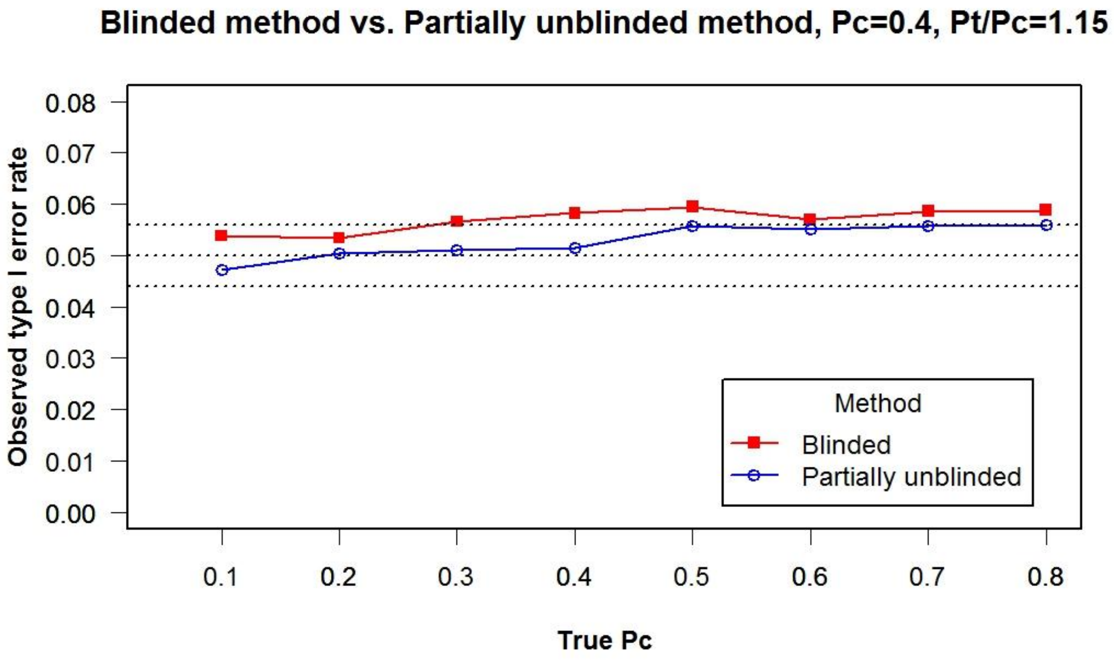

Figure 4.10 The observed type I error rate for p_C^* = 0.4 and p_T^* / p_C^* = 1.15

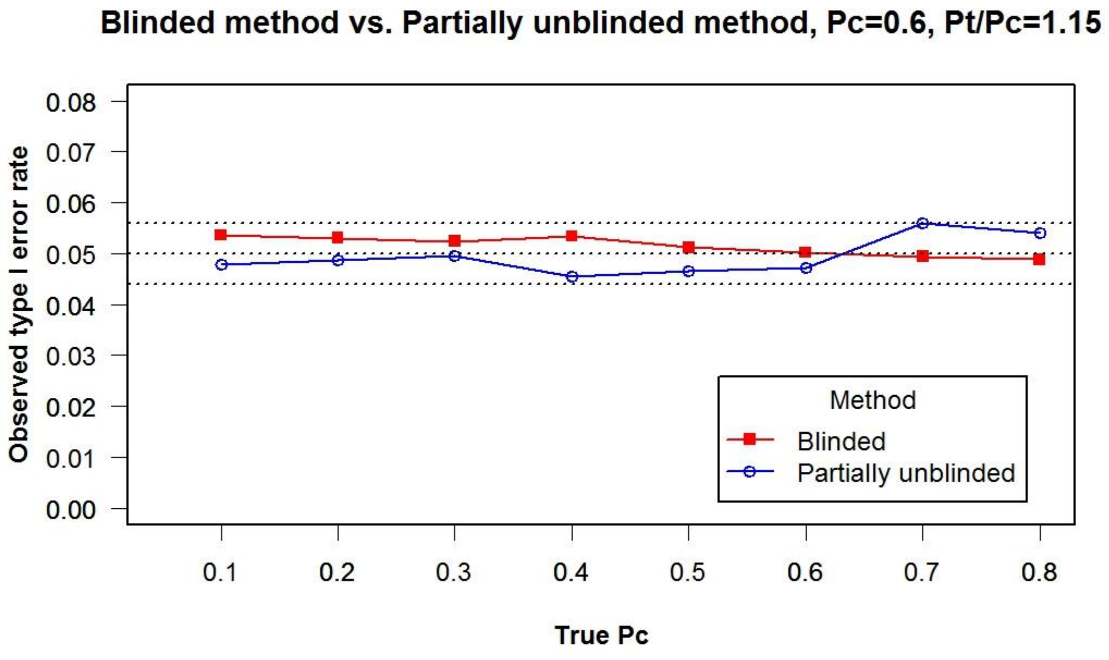

Figure 4.11 The observed type I error rate for p_C^* = 0.6 and p_T^* / p_C^* = 1.15

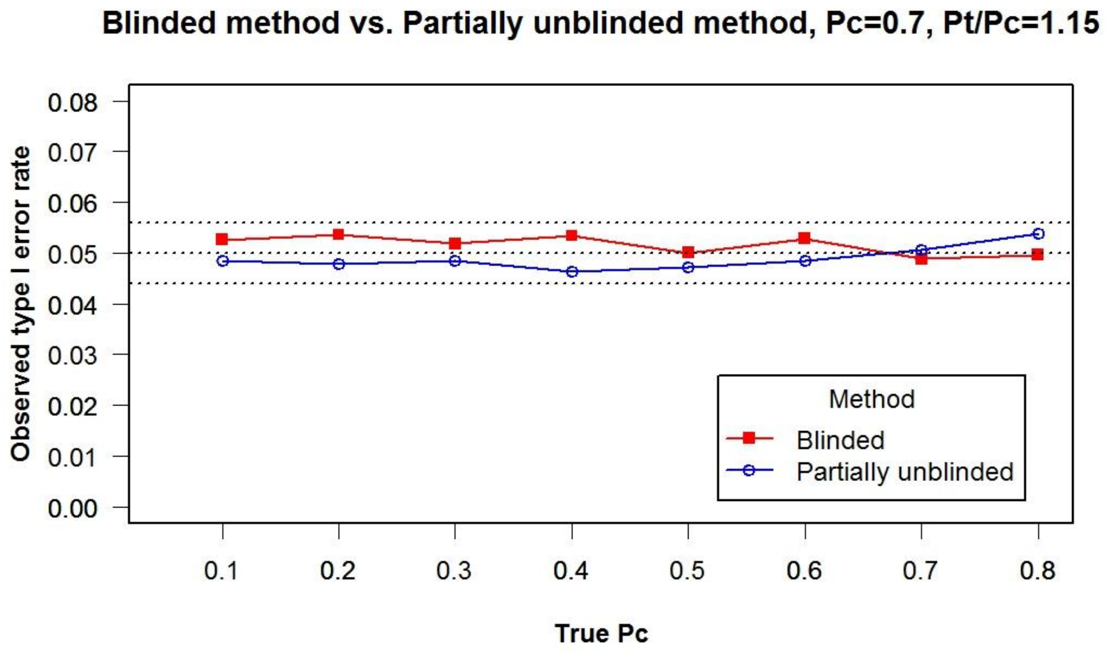

Figure 4.12 The observed type I error rate for p_C^* = 0.7 and p_T^* / p_C^* = 1.15

Chapter 5 - Conclusion and Discussion

In this study, we investigated two methods for sample size re-estimation for clinical trials with a binary endpoint: one blinded method and one partially unblinded method. We performed a simulation study to estimate the type I error rates associated with sample size re-estimation for hypotheses of statistical superiority under the assumption of fixed ratios of success rates together with different combinations of expected success rates and true success rates. The simulation results show that the appropriateness of the initial sample size estimation depends on how close the assumed response rate, p_C^*, is to the true response rate, p_C. When the true p_C is lower than p_C^*, the sample size needs to be increased, whereas when the true p_C is higher than p_C^*, there could be a reduction in the sample size without a loss of desired power. Since in the real trial, the control success rate is often overestimated in the design stage (Proschan, 2005), the sample size without re-estimation is typically not adequate resulting in an underpowered trial. Therefore, the interim re-estimation and adjustment of the sample size is very helpful to make sure the trial is successful. The re-estimated final sample size is sometimes considerably larger than is practical. In these cases, researchers often set an upper limit for the final sample size (for example, $n_{\text{new}} \leq 2n$). When the number of patients recruited attains that limit, the researcher may consider terminating the trial and accept lower power (Gould, 1992; Gould, 1995). In our study, we did not use this constraint.

The results of the simulation also show that the mean of the new sample size estimated by the blinded method is higher than the mean of the new sample size estimated by the partially unblinded method. One possible reason for that is that when we look at the overall data without regard to treatment group, it mostly results a lower overall interim success rate and final sample sizes are negatively correlated with this overall success rate (Gould, 1992).

The two sample-size re-estimation methods studied here both appear to preserve the type I error rate well. Previous studies have shown that recalculating the sample size with the unblinded data may inflate the type I error rate (Wittes & Brittain, 1990; Birkett & Day, 1994). However, the information we used to perform the sample size re-estimation does not provide any information about the true treatment effects, and intuitively, should not affect the type I error rate.

Additionally, we choose the "halfway" point in the trial as the interim time for the sample size re-estimation. This was because if the re-estimation is conducted too early, we may not have adequate information and the estimation may still be imprecise; otherwise, if the re-estimation is performed too late, it may not be very useful to the conduct of the trial. In future work, we suggest trying different interim times for the sample size re-estimation to see what effect the timing of the interim examination has on the sample size re-estimation process, especially if a reduction in sample size is allowed.

Preserving the power of a trial to test the specified H_A is another usual reason for conducting sample size re-estimation. Some studies conducted the interim power evaluations to investigate the effect of sample size re-estimation process on the power (Gould, 1992; Gould, 2001; Kieser & Friede, 2003). The blinded approach appears to preserve the power well. In our design, we chose not to evaluate the power because the sample size was never decreased by the re-estimation process. Nonetheless, an interesting future project would be to study the actual gain in power by re-estimating the sample size with interim data.

References

Armitage, P., Berry, G., & Matthews, J. N. S. (2008). *Statistical methods in medical research*. John Wiley & Sons.

Birkett, M. A., & Day, S. J. (1994). Internal pilot studies for estimating sample size. *Statistics in medicine*, *13*(23-24), 2455-2463.

Friede, T., & Kieser, M. (2004). Sample size recalculation for binary data in internal pilot study designs. *Pharmaceutical Statistics*, *3*(4), 269-279.

Gould, A. L. (1992). Interim analyses for monitoring clinical trials that do not materially affect the type I error rate. *Statistics in medicine*, *11*(1), 55-66.

Gould, A. L. (1995). Planning and revising the sample size for a trial. *Statistics in Medicine*, *14*(9), 1039-1051.

Gould, A. L. (2001). Sample size re-estimation: recent developments and practical considerations. *Statistics in Medicine*, *20*(17-18), 2625-2643.

Kieser, M., & Friede, T. (2003). Simple procedures for blinded sample size adjustment that do not affect the type I error rate. *Statistics in medicine*, *22* (23), 3571-3581.

Lawrence Gould, A., & Shih, W. J. (1992). Sample size re-estimation without unblinding for normally distributed outcomes with unknown variance. *Communications in Statistics-Theory and Methods*, *21*(10), 2833-2853.

Proschan, M. A. (2005). Two-stage sample size re-estimation based on a nuisance parameter: a review. *Journal of biopharmaceutical statistics*, *15*(4), 559-574.

SAS/STAT® software, Version 9.4. Copyright © 2013. SAS Institute Inc., Cary, NC, USA.

Stein, C. (1945). A two-sample test for a linear hypothesis whose power is independent of the variance. *The Annals of Mathematical Statistics*, *16*(3), 243-258.

Wittes, J., & Brittain, E. (1990). The role of internal pilot studies in increasing the efficiency of clinical trials. *Statistics in medicine*, *9*(1-2), 65-72.

Wittes, J., Schabenberger, O., Zucker, D., Brittain, E., & Proschan, M. (1999). Internal pilot studies I: type I error rate of the naive t-test. *Statistics in medicine, 18*(24), 3481-3491.

Appendix A - Tables of Simulation Results

Table A.1 Simulation results for scenario 1 -- $p_C^* = 0.4$ and $p_T^* / p_C^* = 1.25$

tpc	Method	Count_nnew increase	% n increase	Type1_err_rate	Mean of nnew
0.1	pool	5000	100	0.0532	2337
0.2	pool	5000	100	0.0526	1019
0.3	pool	5000	100	0.052	591
0.4	pool	4800	96	0.0476	379
0.5	pool	200	4	0.0454	306
0.6	pool	0	0	0.0488	305
0.7	pool	0	0	0.0502	305
0.8	pool	0	0	0.0494	305
0.1	ctrl	5000	100	0.0474	2127
0.2	ctrl	5000	100	0.0472	893
0.3	ctrl	4985	99.7	0.0528	503
0.4	ctrl	2648	52.96	0.047	331
0.5	ctrl	37	0.74	0.0492	305
0.6	ctrl	0	0	0.0558	305
0.7	ctrl	0	0	0.0558	305
0.8	ctrl	0	0	0.0542	305

Note. The variables denote:

tpc: true event rates for the control group
Method: “pool” -- blinded method; “ctrl” -- partially unblinded method
Count_nnew increase: the number of times that the sample size is increased
% n increase: the percentage of times that the sample size is increased
Type1_err_rate: the type I error rate
Mean of nnew: the mean of n_{new}

Table A.2 Simulation results for scenario 2 -- $p_C^* = 0.6$ and $p_T^* / p_C^* = 1.25$

tpc	Method	Count_nnew increase	% n increase	Type1_err_rate	Mean of nnew
0.1	pool	5000	100	0.0528	2501
0.2	pool	5000	100	0.0528	1049
0.3	pool	5000	100	0.0492	602
0.4	pool	5000	100	0.0496	384
0.5	pool	4999	99.98	0.0478	255
0.6	pool	4710	94.2	0.0468	169
0.7	pool	1218	24.36	0.0474	124
0.8	pool	3	0.06	0.0478	120
0.1	ctrl	4990	99.8	0.0514	2469
0.2	ctrl	5000	100	0.05	951
0.3	ctrl	5000	100	0.0504	524
0.4	ctrl	4994	99.88	0.0466	321
0.5	ctrl	4613	92.26	0.0446	203
0.6	ctrl	2173	43.46	0.0514	138
0.7	ctrl	164	3.28	0.0542	121
0.8	ctrl	1	0.02	0.0532	120

Note. The variables denote:

tpc: true event rates for the control group
Method: “pool” -- blinded method; “ctrl” -- partially unblinded method
Count_nnew increase: the number of times that the sample size is increased
% n increase: the percentage of times that the sample size is increased
Type1_err_rate: the type I error rate
Mean of nnew: the mean of n_{new}

Table A.3 Simulation results for scenario 3 -- p_C^* = 0.7 and p_T^* / p_C^* = 1.25

tpc	Method	Count_nnew increase	% n increase	Type1_err_rate	Mean of nnew
0.1	pool	4992	99.84	0.0472	2762
0.2	pool	5000	100	0.0536	1094
0.3	pool	5000	100	0.0508	617
0.4	pool	5000	100	0.0486	391
0.5	pool	5000	100	0.0456	258
0.6	pool	4998	99.96	0.0506	170
0.7	pool	4711	94.22	0.0468	109
0.8	pool	1910	38.2	0.0496	73
0.1	ctrl	4865	97.3	0.0564	2590
0.2	ctrl	4997	99.94	0.0466	1050
0.3	ctrl	5000	100	0.0492	554
0.4	ctrl	4998	99.96	0.0454	334
0.5	ctrl	4944	98.88	0.041	209
0.6	ctrl	4305	86.1	0.0482	129
0.7	ctrl	2178	43.56	0.046	84
0.8	ctrl	305	6.1	0.0514	68

Note. The variables denote:

tpc: true event rates for the control group
Method: "pool" -- blinded method; "ctrl" -- partially unblinded method
Count_nnew increase: the number of times that the sample size is increased
% n increase: the percentage of times that the sample size is increased
Type1_err_rate: the type I error rate
Mean of nnew: the mean of n_{new}

Table A.4 Simulation results for scenario 4 -- p_C^* = 0.4 and p_T^* / p_C^* = 1.15

tpc	Method	Count_nnew increase	% n increase	Type1_err_rate	Mean of nnew
0.1	pool	5000	100	0.0538	5760
0.2	pool	5000	100	0.0534	2546
0.3	pool	5000	100	0.0566	1482
0.4	pool	4785	95.7	0.0582	953
0.5	pool	0	0	0.0594	840
0.6	pool	0	0	0.057	840
0.7	pool	0	0	0.0586	840
0.8	pool	0	0	0.0588	840
0.1	ctrl	5000	100	0.0472	5395
0.2	ctrl	5000	100	0.0504	2342
0.3	ctrl	5000	100	0.051	1342
0.4	ctrl	2386	47.72	0.0514	878
0.5	ctrl	0	0	0.0558	840
0.6	ctrl	0	0	0.0552	840
0.7	ctrl	0	0	0.0558	840
0.8	ctrl	0	0	0.056	840

Note. The variables denote:

***tpc**:* true event rates for the control group
***Method**:* “pool” -- blinded method; “ctrl” -- partially unblinded method
Count_nnew increase: the number of times that the sample size is increased
***% n increase**:* the percentage of times that the sample size is increased
***Type1_err_rate**:* the type I error rate
Mean of nnew: the mean of n_{new}

Table A.5 Simulation results for scenario 5 -- $p_C^* = 0.6$ and $p_T^* / p_C^* = 1.15$

tpc	Method	Count_nnew increase	% n increase	Type1_err_rate	Mean of nnew
0.1	pool	5000	100	0.0536	5898
0.2	pool	5000	100	0.053	2580
0.3	pool	5000	100	0.0524	1497
0.4	pool	5000	100	0.0534	959
0.5	pool	5000	100	0.0512	639
0.6	pool	4758	95.16	0.0502	425
0.7	pool	63	1.26	0.0494	349
0.8	pool	0	0	0.0488	349
0.1	ctrl	5000	100	0.0478	5615
0.2	ctrl	5000	100	0.0486	2390
0.3	ctrl	5000	100	0.0496	1361
0.4	ctrl	5000	100	0.0454	854
0.5	ctrl	4974	99.48	0.0466	553
0.6	ctrl	2295	45.9	0.0472	375
0.7	ctrl	7	0.14	0.056	349
0.8	ctrl	0	0	0.054	349

Note. The variables denote:

***tpc**:* true event rates for the control group
***Method**:* "pool" -- blinded method; "ctrl" -- partially unblinded method
***Count_nnew increase**:* the number of times that the sample size is increased
***% n increase**:* the percentage of times that the sample size is increased
***Type1_err_rate**:* the type I error rate
***Mean of nnew**:* the mean of n_{new}

Table A.6 Simulation results for scenario 6 -- p_C^* = 0.7 and p_T^* / p_C^* = 1.15

tpc	Method	Count_nnew increase	% n increase	Type1_err_rate	Mean of nnew
0.1	pool	5000	100	0.0526	6039
0.2	pool	5000	100	0.0536	2608
0.3	pool	5000	100	0.0518	1507
0.4	pool	5000	100	0.0534	965
0.5	pool	5000	100	0.05	642
0.6	pool	5000	100	0.0528	425
0.7	pool	4746	94.92	0.0488	274
0.8	pool	223	4.46	0.0496	209
0.1	ctrl	5000	100	0.0484	5922
0.2	ctrl	5000	100	0.0478	2444
0.3	ctrl	5000	100	0.0484	1381
0.4	ctrl	5000	100	0.0464	864
0.5	ctrl	5000	100	0.0472	558
0.6	ctrl	4904	98.08	0.0484	355
0.7	ctrl	2311	46.22	0.0506	231
0.8	ctrl	31	0.62	0.0538	208

Note. The variables denote:

tpc: true event rates for the control group
Method: "pool" -- blinded method; "ctrl" -- partially unblinded method
Count_nnew increase: the number of times that the sample size is increased
% n increase: the percentage of times that the sample size is increased
Type1_err_rate: the type I error rate
Mean of nnew: the mean of n_{new}

Appendix B - Simulation SAS Code

Below is the SAS code used in the simulation studies.

```
/*Scenario 1 -- pc=0.4,pt/pc=1.25*/

%let group=group1;
%let groupname='group1';
%let filepath='/home/dannicong0/www/';
%let rtffile='/home/dannicong0/www/group1.rtf';

libname dannic &filepath;

%macro simulation(truepc=,try=);
data D1;
  pc=0.4;
  pt=0.5;
  p=(pc+pt)/2;
  n=((1.645*sqrt(2*p*(1-p))+0.84*sqrt(pc*(1-pc)+pt*(1-pt)))/(pt-pc))**2; /*the initial
sample size for each arm: n=305*/
run;

data simu1;
  call streaminit(52571);
  do sim=1 to 5000;
  n1=153;
  n2=153;
    x11=rand('binomial',&truepc,n1);
    x21=rand('binomial',&truepc,n2);
    output;
  end;
run;
/*method 1-fix ratio: 1.25, use phatctrlpool*/
data simu2;
  set simu1;
  phatpool=(x11+x21)/(n1+n2);
  pchat=2*phatpool/2.25;
  pthat=1.25*pchat;
  diff=pthat-pchat;
run;

data simu3;
  set simu2;
  nstar=((1.645*sqrt(2*phatpool*(1-phatpool))+0.84*sqrt(pchat*(1-pchat)+pthat*(1-
pthat)))/diff)**2;
  nnew1=ceil(max(nstar,305));
run;

data addition1;
  set simu3;
  ad1=nnew1-153;
run;
/*Average of new sample size*/
ods output summary=means1;
proc means data = simu3;
  var nnew1;
run;

PROC SQL;
create table pool_mean as
SELECT &truepc as tpc,'pool' as method, nnew1_Mean as mean
```

```
FROM means1;
QUIT;

/*generate new data for method 1*/

data simu21;
set addition1;
  call streaminit(52477);
  x12=rand('binomial', &truepc,ad1);
  x22=rand('binomial', &truepc,ad1);
  x1=x11+x12;
  x2=x21+x22;
run;

/*Calculate Z-statistic*/
data simu211;
  set simu21;
  p1hat=x1/nnew1;
  p2hat=x2/nnew1;
  phat=(x1+x2)/(2*nnew1);
  z=(p1hat-p2hat)/sqrt(phat*(1-phat)*2/nnew1);
run;

data simu2111;
  set simu211;
  if z GT 1.645 then y=1;
  if z LT 1.645 then y=0;
  if nnew1>305 then flag_nnew1=1;
  else if nnew1<=305 then flag_nnew1=0;
  keep sim y flag_nnew1;
run;

Proc summary data = simu2111;
 var y flag_nnew1;
 output out=summation_pool sum=;
run;

data summation_pool;
       set summation_pool;
       rename _freq_=freq flag_nnew1=count_nnew;
       keep _freq_ y flag_nnew1;
run;

PROC SQL;
create table pool_type1 as
SELECT &truepc as tpc,'pool' as method, *, y/freq as type1_err_rate
FROM summation_pool;
QUIT;

/*method 2- fixed ratio:1.25, use phatctrl*/
data simu4;
  set simu1;
  pchat=x21/n2;
  pthat=1.25*pchat;
  phatpool=(pchat+pthat)/2;
  diff=pthat-pchat;
run;
data simu5;
  set simu4;
  nstar=((1.645*sqrt(2*phatpool*(1-phatpool))+0.84*sqrt(pchat*(1-pchat)+pthat*(1-
pthat)))/diff)**2;
  nnew2=ceil(max(nstar,305));
run;
```

```
data addition2;
  set simu5;
  ad2=nnew2-153;
run;

/*Average of new sample size*/
ods output summary=means2;
proc means data = simu5;
  var nnew2;
run;

PROC SQL;
create table ctrl_mean as
SELECT &truepc as tpc,'ctrl' as method, nnew2_Mean as mean
FROM means2;
QUIT;

/*generate new data for method 2*/

data simu22;
set addition2;
  call streaminit(52101);
  x12=rand('binomial', &truepc,ad2);
  x22=rand('binomial',&truepc,ad2);
  x1=x11+x12;
  x2=x21+x22;
run;

/*Calculate Z-statistic*/
data simu221;
  set simu22;
  p1hat=x1/nnew2;
  p2hat=x2/nnew2;
  phat=(x1+x2)/(2*nnew2);
  z=(p1hat-p2hat)/sqrt(phat*(1-phat)*2/nnew2);
run;

data simu2211;
  set simu221;
  if z GT 1.645 then y=1;
  if z LT 1.645 then y=0;
  if nnew2>305 then flag_nnew2=1;
  else if nnew2<=305 then flag_nnew2=0;
  keep sim y flag_nnew2;
  run;

Proc summary data = simu2211;
 var y flag_nnew2;
 output out=summation_ctrl sum=;
run;
data summation_ctrl;
       set summation_ctrl;
       rename _freq_=freq flag_nnew2=count_nnew;
       keep _freq_ y flag_nnew2;
run;

PROC SQL;
create table ctrl_type1 as
SELECT &truepc as tpc, 'ctrl' as method, *, y/freq as type1_err_rate
FROM summation_ctrl;
QUIT;
```

```
data result;
       set pool_type1 ctrl_type1;
run;
data means;
       set pool_mean ctrl_mean;
run;
data dannic.result_&try;
set result;
run;
data dannic.means_&try;
set means;
run;

%mend simulation;
options nomprint;
%simulation(truepc=0.1, try=m1);
%simulation(truepc=0.2, try=m2);
%simulation(truepc=0.3, try=m3);
%simulation(truepc=0.4, try=m4);
%simulation(truepc=0.5, try=m5);
%simulation(truepc=0.6, try=m6);
%simulation(truepc=0.7, try=m7);
%simulation(truepc=0.8, try=m8);

data dannic.type1err_temp_&group;
set dannic.result_m1 dannic.result_m2 dannic.result_m3 dannic.result_m4
       dannic.result_m5 dannic.result_m6 dannic.result_m7 dannic.result_m8;
run;
data dannic.means_temp_&group;
set dannic.means_m1 dannic.means_m2 dannic.means_m3 dannic.means_m4
       dannic.means_m5 dannic.means_m6 dannic.means_m7 dannic.means_m8;
run;

proc sql;
create table dannic.type1err_&group as
select &groupname as grpnm, *
from dannic.type1err_temp_&group;
quit;
proc sql;
create table dannic.means_&group as
select &groupname as grpnm, *
from dannic.means_temp_&group;
quit;

ods rtf file=&rtffile;
proc print data = dannic.type1err_&group;run;
proc print data=dannic.means_&group;run;
ods rtf close;


/*Scenario 2 -- pc=0.6,pt/pc=1.25*/

%let group=group2;
%let groupname='group2';
%let filepath='/home/dannicong0/www/';
%let rtffile='/home/dannicong0/www/group2.rtf';

libname dannic &filepath;

%macro simulation(truepc=,try=);
data D1;
  pc=0.6;
  pt=0.75;
  p=(pc+pt)/2;
  n=((1.645*sqrt(2*p*(1-p))+0.84*sqrt(pc*(1-pc)+pt*(1-pt)))/(pt-pc))**2; /*the initial
sample size for each arm: n=120*/
run;

data simu1;
  call streaminit(52571);
  do sim=1 to 5000;
  n1=60;
  n2=60;
    x11=rand('binomial',&truepc,n1);
    x21=rand('binomial',&truepc,n2);
    output;
  end;
run;

/*method 1-fix ratio: 1.25, use phatctrlpool*/
data simu2;
  set simu1;
  phatpool=(x11+x21)/(n1+n2);
  pchat=2*phatpool/2.25;
  pthat=1.25*pchat;
  diff=pthat-pchat;
run;

data simu3;
  set simu2;
  nstar=((1.645*sqrt(2*phatpool*(1-phatpool))+0.84*sqrt(pchat*(1-pchat)+pthat*(1-
pthat)))/diff)**2;
  nnew1=ceil(max(nstar,120));
run;

data addition1;
  set simu3;
  ad1=nnew1-60;
run;

/*Average of new sample size*/
ods output summary=means1;
proc means data = simu3;
  var nnew1;
run;

PROC SQL;
create table pool_mean as
SELECT &truepc as tpc,'pool' as method, nnew1_Mean as mean
FROM means1;
QUIT;

/*generate new data for method 1*/
data simu21;
set addition1;
  call streaminit(52477);
  x12=rand('binomial', &truepc,ad1);
  x22=rand('binomial', &truepc,ad1);
  x1=x11+x12;
  x2=x21+x22;
run;

/*Calculate Z-statistic*/
data simu211;
  set simu21;
  p1hat=x1/nnew1;
  p2hat=x2/nnew1;
  phat=(x1+x2)/(2*nnew1);
  z=(p1hat-p2hat)/sqrt(phat*(1-phat)*2/nnew1);
run;

data simu2111;
  set simu211;
  if z GT 1.645 then y=1;
  if z LT 1.645 then y=0;
  if nnew1>120 then flag_nnew1=1;
  else if nnew1<=120 then flag_nnew1=0;
  keep sim y flag_nnew1;
run;

Proc summary data = simu2111;
 var y flag_nnew1;
 output out=summation_pool sum=;
run;

data summation_pool;
       set summation_pool;
       rename _freq_=freq flag_nnew1=count_nnew;
       keep _freq_ y flag_nnew1;
run;

PROC SQL;
create table pool_type1 as
SELECT &truepc as tpc,'pool' as method, *, y/freq as type1_err_rate
FROM summation_pool;
QUIT;

/*method 2- fixed ratio:1.25, use phatctrl*/
data simu4;
  set simu1;
  pchat=x21/n2;
  pthat=1.25*pchat;
  phatpool=(pchat+pthat)/2;
  diff=pthat-pchat;
run;

data simu5;
  set simu4;
  nstar=((1.645*sqrt(2*phatpool*(1-phatpool))+0.84*sqrt(pchat*(1-pchat)+pthat*(1-
pthat)))/diff)**2;
  nnew2=ceil(max(nstar,120));
run;

data addition2;
  set simu5;
  ad2=nnew2-60;
run;

/*Average of new sample size*/
ods output summary=means2;
proc means data = simu5;
  var nnew2;
run;

PROC SQL;
create table ctrl_mean as
SELECT &truepc as tpc,'ctrl' as method, nnew2_Mean as mean
FROM means2;
QUIT;
```

```
/*generate new data for method 2*/

data simu22;
set addition2;
  call streaminit(52101);
  x12=rand('binomial', &truepc,ad2);
  x22=rand('binomial',&truepc,ad2);
  x1=x11+x12;
  x2=x21+x22;
run;

/*Calculate Z-statistic*/
data simu221;
  set simu22;
  p1hat=x1/nnew2;
  p2hat=x2/nnew2;
  phat=(x1+x2)/(2*nnew2);
  z=(p1hat-p2hat)/sqrt(phat*(1-phat)*2/nnew2);
run;

data simu2211;
  set simu221;
  if z GT 1.645 then y=1;
  if z LT 1.645 then y=0;
  if nnew2>120 then flag_nnew2=1;
  else if nnew2<=120 then flag_nnew2=0;
  keep sim y flag_nnew2;
  run;

Proc summary data = simu2211;
var y flag_nnew2;
output out=summation_ctrl sum=;
run;

data summation_ctrl;
       set summation_ctrl;
       rename _freq_=freq flag_nnew2=count_nnew;
       keep _freq_ y flag_nnew2;
run;

PROC SQL;
create table ctrl_type1 as
SELECT &truepc as tpc, 'ctrl' as method, *, y/freq as type1_err_rate
FROM summation_ctrl;
QUIT;

data result;
       set pool_type1 ctrl_type1;
run;

data means;
       set pool_mean ctrl_mean;
run;

data dannic.result_&try;
set result;
run;

data dannic.means_&try;
set means;
run;
```

```
%mend simulation;
options nomprint;
%simulation(truepc=0.1, try=m1);
%simulation(truepc=0.2, try=m2);
%simulation(truepc=0.3, try=m3);
%simulation(truepc=0.4, try=m4);
%simulation(truepc=0.5, try=m5);
%simulation(truepc=0.6, try=m6);
%simulation(truepc=0.7, try=m7);
%simulation(truepc=0.8, try=m8);

data dannic.type1err_temp_&group;
set dannic.result_m1 dannic.result_m2 dannic.result_m3 dannic.result_m4
     dannic.result_m5 dannic.result_m6 dannic.result_m7 dannic.result_m8;
run;
data dannic.means_temp_&group;
set dannic.means_m1 dannic.means_m2 dannic.means_m3 dannic.means_m4
     dannic.means_m5 dannic.means_m6 dannic.means_m7 dannic.means_m8;
run;

proc sql;
create table dannic.type1err_&group as
select &groupname as grpnm, *
from dannic.type1err_temp_&group;
quit;
proc sql;
create table dannic.means_&group as
select &groupname as grpnm, *
from dannic.means_temp_&group;
quit;

ods rtf file=&rtffile;
proc print data = dannic.type1err_&group;run;
proc print data=dannic.means_&group;run;
ods rtf close;


/*Scenario 3 -- pc=0.7,pt/pc=1.25*/

%let group=group3;
%let groupname='group3';
%let filepath='/home/dannicong0/www/';
%let rtffile='/home/dannicong0/www/group3.rtf';

libname dannic &filepath;

%macro simulation(truepc=,try=);
data D1;
  pc=0.7;
  pt=0.875;
  p=(pc+pt)/2;
  n=((1.645*sqrt(2*p*(1-p))+0.84*sqrt(pc*(1-pc)+pt*(1-pt)))/(pt-pc))**2; /*the initial
sample size for each arm: n=67*/
run;

data simu1;
  call streaminit(52571);
  do sim=1 to 5000;
  n1=34;
  n2=34;
    x11=rand('binomial',&truepc,n1);
    x21=rand('binomial',&truepc,n2);
    output;
  end;
run;

/*method 1-fix ratio: 1.25, use phatctrlpool*/
data simu2;
  set simu1;
  phatpool=(x11+x21)/(n1+n2);
  pchat=2*phatpool/2.25;
  pthat=1.25*pchat;
  diff=pthat-pchat;
run;

data simu3;
  set simu2;
  nstar=((1.645*sqrt(2*phatpool*(1-phatpool))+0.84*sqrt(pchat*(1-pchat)+pthat*(1-
pthat)))/diff)**2;
  nnew1=ceil(max(nstar,67));
run;

data addition1;
  set simu3;
  ad1=nnew1-34;
run;

/*Average of new sample size*/
ods output summary=means1;
proc means data = simu3;
  var nnew1;
run;

PROC SQL;
create table pool_mean as
SELECT &truepc as tpc,'pool' as method, nnew1_Mean as mean
FROM means1;
QUIT;

/*generate new data for method 1*/
data simu21;
set addition1;
  call streaminit(52477);
  x12=rand('binomial', &truepc,ad1);
  x22=rand('binomial', &truepc,ad1);
  x1=x11+x12;
  x2=x21+x22;
run;

/*Calculate Z-statistic*/
data simu211;
  set simu21;
  p1hat=x1/nnew1;
  p2hat=x2/nnew1;
  phat=(x1+x2)/(2*nnew1);
  z=(p1hat-p2hat)/sqrt(phat*(1-phat)*2/nnew1);
run;

data simu2111;
  set simu211;
  if z GT 1.645 then y=1;
  if z LT 1.645 then y=0;
  if nnew1>67 then flag_nnew1=1;
  else if nnew1<=67 then flag_nnew1=0;
  keep sim y flag_nnew1;
run;
```

```
Proc summary data = simu2111;
 var y flag_nnew1;
 output out=summation_pool sum=;
run;

data summation_pool;
       set summation_pool;
       rename _freq_=freq flag_nnew1=count_nnew;
       keep _freq_ y flag_nnew1;
run;

PROC SQL;
create table pool_type1 as
SELECT &truepc as tpc,'pool' as method, *, y/freq as type1_err_rate
FROM summation_pool;
QUIT;

/*method 2- fixed ratio:1.25, use phatctrl*/
data simu4;
  set simu1;
  pchat=x21/n2;
  pthat=1.25*pchat;
  phatpool=(pchat+pthat)/2;
  diff=pthat-pchat;
run;

data simu5;
  set simu4;
  nstar=((1.645*sqrt(2*phatpool*(1-phatpool))+0.84*sqrt(pchat*(1-pchat)+pthat*(1-
pthat)))/diff)**2;
  nnew2=ceil(max(nstar,67));
run;

data addition2;
  set simu5;
  ad2=nnew2-34;
run;

/*Average of new sample size*/
ods output summary=means2;
proc means data = simu5;
  var nnew2;
run;

PROC SQL;
create table ctrl_mean as
SELECT &truepc as tpc,'ctrl' as method, nnew2_Mean as mean
FROM means2;
QUIT;


/*generate new data for method 2*/
data simu22;
set addition2;
  call streaminit(52101);
  x12=rand('binomial', &truepc,ad2);
  x22=rand('binomial',&truepc,ad2);
  x1=x11+x12;
  x2=x21+x22;
run;

/*Calculate Z-statistic*/
```

```
data simu221;
  set simu22;
  p1hat=x1/nnew2;
  p2hat=x2/nnew2;
  phat=(x1+x2)/(2*nnew2);
  z=(p1hat-p2hat)/sqrt(phat*(1-phat)*2/nnew2);
run;

data simu2211;
  set simu221;
  if z GT 1.645 then y=1;
  if z LT 1.645 then y=0;
  if nnew2>67 then flag_nnew2=1;
  else if nnew2<=67 then flag_nnew2=0;
  keep sim y flag_nnew2;
  run;

Proc summary data = simu2211;
 var y flag_nnew2;
 output out=summation_ctrl sum=;
run;

data summation_ctrl;
       set summation_ctrl;
       rename _freq_=freq flag_nnew2=count_nnew;
       keep _freq_ y flag_nnew2;
run;

PROC SQL;
create table ctrl_type1 as
SELECT &truepc as tpc, 'ctrl' as method, *, y/freq as type1_err_rate
FROM summation_ctrl;
QUIT;

data result;
       set pool_type1 ctrl_type1;
run;

data means;
       set pool_mean ctrl_mean;
run;

data dannic.result_&try;
set result;
run;

data dannic.means_&try;
set means;
run;

%mend simulation;
options nomprint;
%simulation(truepc=0.1, try=m1);
%simulation(truepc=0.2, try=m2);
%simulation(truepc=0.3, try=m3);
%simulation(truepc=0.4, try=m4);
%simulation(truepc=0.5, try=m5);
%simulation(truepc=0.6, try=m6);
%simulation(truepc=0.7, try=m7);
%simulation(truepc=0.8, try=m8);

data dannic.type1err_temp_&group;
set dannic.result_m1 dannic.result_m2 dannic.result_m3 dannic.result_m4
```

```
      dannic.result_m5 dannic.result_m6 dannic.result_m7 dannic.result_m8;
run;
data dannic.means_temp_&group;
set dannic.means_m1 dannic.means_m2 dannic.means_m3 dannic.means_m4
      dannic.means_m5 dannic.means_m6 dannic.means_m7 dannic.means_m8;
run;

proc sql;
create table dannic.type1err_&group as
select &groupname as grpnm, *
from dannic.type1err_temp_&group;
quit;
proc sql;
create table dannic.means_&group as
select &groupname as grpnm, *
from dannic.means_temp_&group;
quit;

ods rtf file=&rtffile;
proc print data = dannic.type1err_&group;run;
proc print data=dannic.means_&group;run;
ods rtf close;



/*Scenario 4 -- pc=0.4,pt/pc=1.15*/

%let group=group4;
%let groupname='group4';
%let filepath='/home/dannicong0/www/';
%let rtffile='/home/dannicong0/www/group4.rtf';

libname dannic &filepath;

%macro simulation(truepc=,try=);
data D1;
  pc=0.4;
  pt=0.46;
  p=(pc+pt)/2;
  n=((1.645*sqrt(2*p*(1-p))+0.84*sqrt(pc*(1-pc)+pt*(1-pt)))/(pt-pc))**2; /*the initial
sample size for each arm: n=840*/
run;

data simu1;
  call streaminit(52571);
  do sim=1 to 5000;
  n1=420;
  n2=420;
    x11=rand('binomial',&truepc,n1);
    x21=rand('binomial',&truepc,n2);
    output;
  end;
run;

/*method 1-fix ratio: 1.15, use phatctrlpool*/
data simu2;
  set simu1;
  phatpool=(x11+x21)/(n1+n2);
  pchat=2*phatpool/2.15;
  pthat=1.15*pchat;
  diff=pthat-pchat;
run;
```

```
data simu3;
  set simu2;
  nstar=((1.645*sqrt(2*phatpool*(1-phatpool))+0.84*sqrt(pchat*(1-pchat)+pthat*(1-
pthat)))/diff)**2;
  nnew1=ceil(max(nstar,840));
run;

data addition1;
  set simu3;
  ad1=nnew1-420;
run;


/*Average of new sample size*/
ods output summary=means1;
proc means data = simu3;
  var nnew1;
run;

PROC SQL;
create table pool_mean as
SELECT &truepc as tpc,'pool' as method, nnew1_Mean as mean
FROM means1;
QUIT;

/*generate new data for method 1*/
data simu21;
set addition1;
  call streaminit(52477);
  x12=rand('binomial', &truepc,ad1);
  x22=rand('binomial', &truepc,ad1);
  x1=x11+x12;
  x2=x21+x22;
run;

/*Calculate Z-statistic*/
data simu211;
  set simu21;
  p1hat=x1/nnew1;
  p2hat=x2/nnew1;
  phat=(x1+x2)/(2*nnew1);
  z=(p1hat-p2hat)/sqrt(phat*(1-phat)*2/nnew1);
run;

data simu2111;
  set simu211;
  if z GT 1.645 then y=1;
  if z LT 1.645 then y=0;
  if nnew1>840 then flag_nnew1=1;
  else if nnew1<=840 then flag_nnew1=0;
  keep sim y flag_nnew1;
run;

Proc summary data = simu2111;
 var y flag_nnew1;
 output out=summation_pool sum=;
run;

data summation_pool;
       set summation_pool;
       rename _freq_=freq flag_nnew1=count_nnew;
       keep _freq_ y flag_nnew1;
run;
```

```
PROC SQL;
create table pool_type1 as
SELECT &truepc as tpc,'pool' as method, *, y/freq as type1_err_rate
FROM summation_pool;
QUIT;

/*method 2- fixed ratio:1.15, use phatctrl*/
data simu4;
  set simu1;
  pchat=x21/n2;
  pthat=1.15*pchat;
  phatpool=(pchat+pthat)/2;
  diff=pthat-pchat;
run;

data simu5;
  set simu4;
  nstar=((1.645*sqrt(2*phatpool*(1-phatpool))+0.84*sqrt(pchat*(1-pchat)+pthat*(1-
pthat)))/diff)**2;
  nnew2=ceil(max(nstar,840));
run;

data addition2;
  set simu5;
  ad2=nnew2-420;
run;

/*Average of new sample size*/
ods output summary=means2;
proc means data = simu5;
  var nnew2;
run;

PROC SQL;
create table ctrl_mean as
SELECT &truepc as tpc,'ctrl' as method, nnew2_Mean as mean
FROM means2;
QUIT;

/*generate new data for method 2*/
data simu22;
set addition2;
  call streaminit(52101);
  x12=rand('binomial', &truepc,ad2);
  x22=rand('binomial',&truepc,ad2);
  x1=x11+x12;
  x2=x21+x22;
run;

/*Calculate Z-statistic*/
data simu221;
  set simu22;
  p1hat=x1/nnew2;
  p2hat=x2/nnew2;
  phat=(x1+x2)/(2*nnew2);
  z=(p1hat-p2hat)/sqrt(phat*(1-phat)*2/nnew2);
run;

data simu2211;
  set simu221;
  if z GT 1.645 then y=1;
  if z LT 1.645 then y=0;
  if nnew2>840 then flag_nnew2=1;
  else if nnew2<=840 then flag_nnew2=0;
  keep sim y flag_nnew2;
  run;

Proc summary data = simu2211;
 var y flag_nnew2;
 output out=summation_ctrl sum=;
run;

data summation_ctrl;
       set summation_ctrl;
       rename _freq_=freq flag_nnew2=count_nnew;
       keep _freq_ y flag_nnew2;
run;

PROC SQL;
create table ctrl_type1 as
SELECT &truepc as tpc, 'ctrl' as method, *, y/freq as type1_err_rate
FROM summation_ctrl;
QUIT;

data result;
       set pool_type1 ctrl_type1;
run;

data means;
       set pool_mean ctrl_mean;
run;

data dannic.result_&try;
set result;
run;

data dannic.means_&try;
set means;
run;
%mend simulation;
options nomprint;
%simulation(truepc=0.1, try=m1);
%simulation(truepc=0.2, try=m2);
%simulation(truepc=0.3, try=m3);
%simulation(truepc=0.4, try=m4);
%simulation(truepc=0.5, try=m5);
%simulation(truepc=0.6, try=m6);
%simulation(truepc=0.7, try=m7);
%simulation(truepc=0.8, try=m8);

data dannic.type1err_temp_&group;
set dannic.result_m1 dannic.result_m2 dannic.result_m3 dannic.result_m4
       dannic.result_m5 dannic.result_m6 dannic.result_m7 dannic.result_m8;
run;
data dannic.means_temp_&group;
set dannic.means_m1 dannic.means_m2 dannic.means_m3 dannic.means_m4
       dannic.means_m5 dannic.means_m6 dannic.means_m7 dannic.means_m8;
run;

proc sql;
create table dannic.type1err_&group as
select &groupname as grpnm, *
from dannic.type1err_temp_&group;
quit;
proc sql;
```

```
create table dannic.means_&group as
select &groupname as grpnm, *
from dannic.means_temp_&group;
quit;

ods rtf file=&rtffile;
proc print data = dannic.type1err_&group;run;
proc print data=dannic.means_&group;run;
ods rtf close;

/*Scenario 5 -- pc=0.6,pt/pc=1.15*/

%let group=group5;
%let groupname='group5';
%let filepath='/home/dannicong0/www/';
%let rtffile='/home/dannicong0/www/group5.rtf';

libname dannic &filepath;

%macro simulation(truepc=,try=);
data D1;
  pc=0.6;
  pt=0.69;
  p=(pc+pt)/2;
  n=((1.645*sqrt(2*p*(1-p))+0.84*sqrt(pc*(1-pc)+pt*(1-pt)))/(pt-pc))**2; /*the initial
sample size for each arm: n=349*/
run;

data simu1;
  call streaminit(52571);
  do sim=1 to 5000;
  n1=175;
  n2=175;
    x11=rand('binomial',&truepc,n1);
    x21=rand('binomial',&truepc,n2);
    output;
  end;
run;

/*method 1-fix ratio: 1.15, use phatctrlpool*/
data simu2;
  set simu1;
  phatpool=(x11+x21)/(n1+n2);
  pchat=2*phatpool/2.15;
  pthat=1.15*pchat;
  diff=pthat-pchat;
run;

data simu3;
  set simu2;
  nstar=((1.645*sqrt(2*phatpool*(1-phatpool))+0.84*sqrt(pchat*(1-pchat)+pthat*(1-
pthat)))/diff)**2;
  nnew1=ceil(max(nstar,349));
run;

data addition1;
  set simu3;
  ad1=nnew1-175;
run;

/*Average of new sample size*/
ods output summary=means1;
proc means data = simu3;
```

```
  var nnew1;
run;

PROC SQL;
create table pool_mean as
SELECT &truepc as tpc,'pool' as method, nnew1_Mean as mean
FROM means1;
QUIT;

/*generate new data for method 1*/
data simu21;
set addition1;
  call streaminit(52477);
  x12=rand('binomial', &truepc,ad1);
  x22=rand('binomial', &truepc,ad1);
  x1=x11+x12;
  x2=x21+x22;
run;

/*Calculate Z-statistic*/
data simu211;
  set simu21;
  p1hat=x1/nnew1;
  p2hat=x2/nnew1;
  phat=(x1+x2)/(2*nnew1);
  z=(p1hat-p2hat)/sqrt(phat*(1-phat)*2/nnew1);
run;

data simu2111;
  set simu211;
  if z GT 1.645 then y=1;
  if z LT 1.645 then y=0;
  if nnew1>349 then flag_nnew1=1;
  else if nnew1<=349 then flag_nnew1=0;
  keep sim y flag_nnew1;
run;

Proc summary data = simu2111;
 var y flag_nnew1;
 output out=summation_pool sum=;
run;

data summation_pool;
       set summation_pool;
       rename _freq_=freq flag_nnew1=count_nnew;
       keep _freq_ y flag_nnew1;
run;

PROC SQL;
create table pool_type1 as
SELECT &truepc as tpc,'pool' as method, *, y/freq as type1_err_rate
FROM summation_pool;
QUIT;

/*method 2- fixed ratio:1.15, use phatctrl*/
data simu4;
  set simu1;
  pchat=x21/n2;
  pthat=1.15*pchat;
  phatpool=(pchat+pthat)/2;
  diff=pthat-pchat;
run;
```

```
data simu5;
  set simu4;
  nstar=((1.645*sqrt(2*phatpool*(1-phatpool))+0.84*sqrt(pchat*(1-pchat)+pthat*(1-
pthat)))/diff)**2;
  nnew2=ceil(max(nstar,349));
run;

data addition2;
  set simu5;
  ad2=nnew2-175;
run;

/*Average of new sample size*/
ods output summary=means2;
proc means data = simu5;
  var nnew2;
run;

PROC SQL;
create table ctrl_mean as
SELECT &truepc as tpc,'ctrl' as method, nnew2_Mean as mean
FROM means2;
QUIT;

/*generate new data for method 2*/
data simu22;
set addition2;
  call streaminit(52101);
  x12=rand('binomial', &truepc,ad2);
  x22=rand('binomial',&truepc,ad2);
  x1=x11+x12;
  x2=x21+x22;
run;

/*Calculate Z-statistic*/
data simu221;
  set simu22;
  p1hat=x1/nnew2;
  p2hat=x2/nnew2;
  phat=(x1+x2)/(2*nnew2);
  z=(p1hat-p2hat)/sqrt(phat*(1-phat)*2/nnew2);
run;

data simu2211;
  set simu221;
  if z GT 1.645 then y=1;
  if z LT 1.645 then y=0;
  if nnew2>349 then flag_nnew2=1;
  else if nnew2<=349 then flag_nnew2=0;
  keep sim y flag_nnew2;
  run;

Proc summary data = simu2211;
 var y flag_nnew2;
 output out=summation_ctrl sum=;
run;

data summation_ctrl;
       set summation_ctrl;
       rename _freq_=freq flag_nnew2=count_nnew;
       keep _freq_ y flag_nnew2;
run;
```

```
PROC SQL;
create table ctrl_type1 as
SELECT &truepc as tpc, 'ctrl' as method, *, y/freq as type1_err_rate
FROM summation_ctrl;
QUIT;

data result;
       set pool_type1 ctrl_type1;
run;

data means;
       set pool_mean ctrl_mean;
run;

data dannic.result_&try;
set result;
run;

data dannic.means_&try;
set means;
run;

%mend simulation;
options nomprint;
%simulation(truepc=0.1, try=m1);
%simulation(truepc=0.2, try=m2);
%simulation(truepc=0.3, try=m3);
%simulation(truepc=0.4, try=m4);
%simulation(truepc=0.5, try=m5);
%simulation(truepc=0.6, try=m6);
%simulation(truepc=0.7, try=m7);
%simulation(truepc=0.8, try=m8);

data dannic.type1err_temp_&group;
set dannic.result_m1 dannic.result_m2 dannic.result_m3 dannic.result_m4
       dannic.result_m5 dannic.result_m6 dannic.result_m7 dannic.result_m8;
run;
data dannic.means_temp_&group;
set dannic.means_m1 dannic.means_m2 dannic.means_m3 dannic.means_m4
       dannic.means_m5 dannic.means_m6 dannic.means_m7 dannic.means_m8;
run;

proc sql;
create table dannic.type1err_&group as
select &groupname as grpnm, *
from dannic.type1err_temp_&group;
quit;
proc sql;
create table dannic.means_&group as
select &groupname as grpnm, *
from dannic.means_temp_&group;
quit;

ods rtf file=&rtffile;
proc print data = dannic.type1err_&group;run;
proc print data=dannic.means_&group;run;
ods rtf close;


/*Scenario 6 -- pc=0.7,pt/pc=1.15*/

%let group=group6;
%let groupname='group6';
```

```
%let filepath='/home/dannicong0/www/';
%let rtffile='/home/dannicong0/www/group6.rtf';

libname dannic &filepath;

%macro simulation(truepc=,try=);
data D1;
  pc=0.7;
  pt=0.805;
  p=(pc+pt)/2;
  n=((1.645*sqrt(2*p*(1-p))+0.84*sqrt(pc*(1-pc)+pt*(1-pt)))/(pt-pc))**2; /*the initial
sample size for each arm: n=208*/
run;

data simu1;
  call streaminit(52571);
  do sim=1 to 5000;
  n1=104;
  n2=104;
    x11=rand('binomial',&truepc,n1);
    x21=rand('binomial',&truepc,n2);
    output;
  end;
run;

/*method 1-fix ratio: 1.15, use phatctrlpool*/
data simu2;
  set simu1;
  phatpool=(x11+x21)/(n1+n2);
  pchat=2*phatpool/2.15;
  pthat=1.15*pchat;
  diff=pthat-pchat;
run;

data simu3;
  set simu2;
  nstar=((1.645*sqrt(2*phatpool*(1-phatpool))+0.84*sqrt(pchat*(1-pchat)+pthat*(1-
pthat)))/diff)**2;
  nnew1=ceil(max(nstar,208));
run;

data addition1;
  set simu3;
  ad1=nnew1-104;
run;

/*Average of new sample size*/
ods output summary=means1;
proc means data = simu3;
  var nnew1;
run;

PROC SQL;
create table pool_mean as
SELECT &truepc as tpc,'pool' as method, nnew1_Mean as mean
FROM means1;
QUIT;

/*generate new data for method 1*/
data simu21;
set addition1;
  call streaminit(52477);
  x12=rand('binomial', &truepc,ad1);
  x22=rand('binomial', &truepc,ad1);
  x1=x11+x12;
  x2=x21+x22;
run;

/*Calculate Z-statistic*/
data simu211;
  set simu21;
  p1hat=x1/nnew1;
  p2hat=x2/nnew1;
  phat=(x1+x2)/(2*nnew1);
  z=(p1hat-p2hat)/sqrt(phat*(1-phat)*2/nnew1);
run;

data simu2111;
  set simu211;
  if z GT 1.645 then y=1;
  if z LT 1.645 then y=0;
  if nnew1>208 then flag_nnew1=1;
  else if nnew1<=208 then flag_nnew1=0;
  keep sim y flag_nnew1;
run;

Proc summary data = simu2111;
 var y flag_nnew1;
 output out=summation_pool sum=;
run;

data summation_pool;
       set summation_pool;
       rename _freq_=freq flag_nnew1=count_nnew;
       keep _freq_ y flag_nnew1;
run;

PROC SQL;
create table pool_type1 as
SELECT &truepc as tpc,'pool' as method, *, y/freq as type1_err_rate
FROM summation_pool;
QUIT;

/*method 2- fixed ratio:1.15, use phatctrl*/
data simu4;
  set simu1;
  pchat=x21/n2;
  pthat=1.15*pchat;
  phatpool=(pchat+pthat)/2;
  diff=pthat-pchat;
run;

data simu5;
  set simu4;
  nstar=((1.645*sqrt(2*phatpool*(1-phatpool))+0.84*sqrt(pchat*(1-pchat)+pthat*(1-
pthat)))/diff)**2;
  nnew2=ceil(max(nstar,208));
run;

data addition2;
  set simu5;
  ad2=nnew2-104;
run;

/*Average of new sample size*/
ods output summary=means2;
```

```
proc means data = simu5;
  var nnew2;
run;

PROC SQL;
create table ctrl_mean as
SELECT &truepc as tpc,'ctrl' as method, nnew2_Mean as mean
FROM means2;
QUIT;

/*generate new data for method 2*/
data simu22;
set addition2;
  call streaminit(52101);
  x12=rand('binomial', &truepc,ad2);
  x22=rand('binomial',&truepc,ad2);
  x1=x11+x12;
  x2=x21+x22;
run;

/*Calculate Z-statistic*/
data simu221;
  set simu22;
  p1hat=x1/nnew2;
  p2hat=x2/nnew2;
  phat=(x1+x2)/(2*nnew2);
  z=(p1hat-p2hat)/sqrt(phat*(1-phat)*2/nnew2);
run;

data simu2211;
  set simu221;
  if z GT 1.645 then y=1;
  if z LT 1.645 then y=0;
  if nnew2>208 then flag_nnew2=1;
  else if nnew2<=208 then flag_nnew2=0;
  keep sim y flag_nnew2;
  run;

Proc summary data = simu2211;
 var y flag_nnew2;
 output out=summation_ctrl sum=;
run;

data summation_ctrl;
       set summation_ctrl;
       rename _freq_=freq flag_nnew2=count_nnew;
       keep _freq_ y flag_nnew2;
run;

PROC SQL;
create table ctrl_type1 as
SELECT &truepc as tpc, 'ctrl' as method, *, y/freq as type1_err_rate
FROM summation_ctrl;
QUIT;

data result;
       set pool_type1 ctrl_type1;
run;

data means;
       set pool_mean ctrl_mean;
run;
```

```
data dannic.result_&try;
set result;
run;

data dannic.means_&try;
set means;
run;

%mend simulation;
options nomprint;
%simulation(truepc=0.1, try=m1);
%simulation(truepc=0.2, try=m2);
%simulation(truepc=0.3, try=m3);
%simulation(truepc=0.4, try=m4);
%simulation(truepc=0.5, try=m5);
%simulation(truepc=0.6, try=m6);
%simulation(truepc=0.7, try=m7);
%simulation(truepc=0.8, try=m8);

data dannic.type1err_temp_&group;
set dannic.result_m1 dannic.result_m2 dannic.result_m3 dannic.result_m4
      dannic.result_m5 dannic.result_m6 dannic.result_m7 dannic.result_m8;
run;
data dannic.means_temp_&group;
set dannic.means_m1 dannic.means_m2 dannic.means_m3 dannic.means_m4
      dannic.means_m5 dannic.means_m6 dannic.means_m7 dannic.means_m8;
run;

proc sql;
create table dannic.type1err_&group as
select &groupname as grpnm, *
from dannic.type1err_temp_&group;
quit;
proc sql;
create table dannic.means_&group as
select &groupname as grpnm, *
from dannic.means_temp_&group;
quit;

ods rtf file=&rtffile;
proc print data = dannic.type1err_&group;run;
proc print data=dannic.means_&group;run;
ods rtf close;
```