5-2019

Methods for Meta–Analyses of Rare Events, Sparse Data, and Heterogeneity

Brinley Zabriskie
Utah State University

METHODS FOR META–ANALYSES OF RARE EVENTS,

SPARSE DATA, AND HETEROGENEITY

by

Brinley Zabriskie

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

of

DOCTOR OF PHILOSOPHY

in

Mathematical Sciences
(Statistics Specialization)

Approved:

Chris Corcoran, Sc.D. Major Professor	Adele Cutler, Ph.D. Committee Member
John Stevens, Ph.D. Committee Member	David Brown, Ph.D. Committee Member
Tyler Brough, Ph.D. Committee Member	Richard S. Inouye, Ph.D. Vice Provost for Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2019

ABSTRACT

Methods for Meta–Analyses of Rare Events,

Sparse Data, and Heterogeneity

by

Brinley Zabriskie, Doctor of Philosophy

Utah State University, 2019

Major Professor: Dr. Chris Corcoran
Department: Mathematics and Statistics

The increasingly widespread use of meta–analysis has led to growing interest in meta–analytic methods for rare events and sparse data. Conventional approaches tend to perform very poorly in such settings. Recent work in this area has provided options for sparse data, but these are still often hampered when heterogeneity across the available studies differs based on treatment group. We propose several new exact methods that accommodate this common contingency, providing more reliable statistical tests when such patterns on heterogeneity are observed. First, we develop a permutation–based approach that can also be used as a basis for computing exact confidence intervals when estimating the effect size. Second, we extend the permutation–based approach to the network meta–analysis setting. Third, we develop a new exact confidence distribution approach for effect size estimation. We show these new methods perform markedly better than traditional methods when events are rare, and heterogeneity is present.

(264 pages)

PUBLIC ABSTRACT

Methods for Meta–Analyses of Rare Events,

Sparse Data, and Heterogeneity

Brinley Zabriskie

The vast and complex wealth of information available to researchers often leads to a systematic review, which involves a detailed and comprehensive plan and search strategy with the goal of identifying, appraising, and synthesizing all relevant studies on a particular topic. A meta–analysis, conducted ideally as part of a comprehensive systematic review, statistically synthesizes evidence from multiple independent studies to produce one overall conclusion. The increasingly widespread use of meta–analysis has led to growing interest in meta–analytic methods for rare events and sparse data. Conventional approaches tend to perform very poorly in such settings. Recent work in this area has provided options for sparse data, but these are still often hampered when heterogeneity across the available studies differs based on treatment group. Heterogeneity arises when participants in a study are more correlated than participants across studies, often stemming from differences in the administration of the treatment, study design, or measurement of the outcome. We propose several new exact methods that accommodate this common contingency, providing more reliable statistical tests when such patterns on heterogeneity are observed. First, we develop a permutation–based approach that can also be used as a basis for computing exact confidence intervals when estimating the effect size. Second, we extend the permutation–based approach to the network meta–analysis setting. Third, we develop a new exact confidence distribution approach for effect size estimation. We show these new methods perform markedly better than traditional methods when events are rare, and heterogeneity is present.

ACKNOWLEDGMENTS

I would like to first thank Dr. Chris Corcoran for providing me the opportunity to pursue this degree and for his time, advice, and encouragement. I am also grateful for the teaching opportunities he gave me throughout my time at Utah State University. Dr. Corcoran has been very influential in my career development. I would also like to thank my committee members, Drs. Adele Cutler, John Stevens, Dave Brown, and Tyler Brough for their time, input, and assistance. Additionally, the faculty, staff, and students in the Department of Mathematics and Statistics have made my time here enjoyable and memorable.

I would also like to thank the co–founders of Cytel Inc., Drs. Cyrus Mehta and Nitin Patel, whose pioneering advancements made this work possible. I greatly appreciate Dr. Pralay Senchaudhuri, Senior Vice President for Research & Development at Cytel Inc., who has taken the time to provide invaluable advice throughout this process. I would also like to thank software specialist Sumit Singh who provided technical support. This work would also not have been possible without the Center for High Performance Computing at the University of Utah. I am grateful for their resources and technical assistance.

Lastly, I am grateful for my family for their assistance, support, and encouragement.

Brinley Zabriskie

CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION TO META–ANALYSIS METHODS FOR RARE EVENTS AND HETEROGENEITY

1.1 Background

A meta–analysis, conducted ideally as part of a comprehensive systematic review, synthesizes evidence from multiple independent studies to produce one overall conclusion. Meta–analyses provide a consolidated review of the vast and complex body of literature, that may sometimes contain conflicting conclusions, by combining and comparing effect sizes using metrics like the odds ratio, relative risk, risk difference, correlation coefficient, and mean difference. Combining effect sizes yields an overall effect size estimate, which can be used for significance testing and creating confidence intervals. Comparing effect sizes is used to analyze the accuracy and reliability of the combined effect size estimate. This includes analyzing the effects of heterogeneity, or differences in variability due to treatment.

Meta–analyses are increasingly employed across many disciplines, including public health, medicine, epidemiology, psychology, criminology, sociology, political science, and genetics. One of the earliest uses of a meta–analysis can be traced back to the British statistician Karl Pearson. In 1904, he combined observations from multiple clinical studies assessing typhoid vaccine effectiveness (Sutton and Higgins, 2008). Over the ensuing decades, this form of research synthesis was formalized and grew in popularity and application. In 1976, Gene Glass coined the term "meta–analysis" to describe this pooling of study results, but still relatively few of these analyses were conducted (O'Rourke, 2007). Meanwhile, contemporaries of Glass began to more fully recognize a need for clear guidelines on conducting literature reviews and sharing research findings. A significant

response to this need came in the form of the international Cochrane Collaboration, which was established in the early 1990s. The mission of this organization is to "promote evidence–informed health decision–making by producing high–quality, relevant, accessible systematic reviews and other synthesized research evidence" (Cochrane, 2019). The structure of this organization (and many others like it), the ability to store information electronically, and the wide accessibility of the internet helped make meta–analyses much more feasible and popular to conduct. As such, the use of meta–analyses has increased dramatically over the years, with thousands of meta–analyses and systematic reviews now published each year (Sutton and Higgins, 2008).

This growth has been especially prevalent in public health and medicine, particularly in the context of treatment comparisons (Sutton and Higgins, 2008). Where expert opinion once played a significant role in healthcare decision–making, a recent push toward "evidence–based medicine" now has medical data driving health–related decisions (Ioannidis, 2006). Systematic reviews and meta–analyses are at the heart of this ideal, and they are considered by some to be the highest form of evidence (Akobeng, 2005).

In spite of the widespread use of systematic reviews and meta–analysis, there are, nonetheless, many obstacles researchers face when using these approaches. To begin, conducting a literature search sufficient to obtain all studies conducted on a specific treatment requires an extreme amount of effort and expertise. Study results may come from different journals, countries, and languages. While it is relatively straightforward, albeit time intensive, to collect published studies on a specific treatment, it is much more difficult to gather information from unpublished studies. Including only published studies leads to what is known as the "file drawer" problem, or publication bias, which occurs as journals tend to publish only studies with significant results, leaving many studies that show non–significant results in one person's "file drawer." If all the significant, published studies happen to be results of a type I error, and all the non–significant, non–published results actually reflect the truth, the results of a meta–analysis will be completely

inaccurate. Conducting an extensive systematic review is daunting, but critical, for meta–analyses.

However, even when all relevant studies have been ascertained, the quality of each study must be determined to avoid the "garbage in, garbage out" problem. If the individual studies used in a meta–analysis are biased or have flawed study designs, the combined estimate produced by a meta–analysis will likewise be biased and flawed. Each individual study should be carefully reviewed by experts in the field of interest to ensure the data for a meta–analysis are high quality and of sound methodology. Another so–called "apples and oranges" problem arises when studies considered for a meta–analysis are not actually comparable. Prior to conducting a systematic review, researchers should develop clear criteria as to what characteristics a study needs in order to be included in the analysis so that only similar studies are combined in a meta–analysis. For example, if two studies are found that assess the effectiveness of aspirin in relieving a headache, but one study included only female patients while the other included only male patients, it may be unwise to pool both studies together depending on the research question.

After all the work required to ensure the comprehensive accumulation of relevant data, and the additional critical evaluation of the available studies to ensure that they are sufficiently comparable, the statistical analysis requires additional care. It is computationally easy to conduct a meta–analysis with widely–available software, but additional insight is required to understand how to choose and apply appropriate models and how their characteristics and potential drawbacks in certain settings need to be considered when interpreting and presenting the results.

With these considerations in mind, both with respect to systematic reviews and the statistical methodology of meta–analysis, the potential benefits of meta–analyses have led to a significant increase in their application. Advances in the scope and performance of the internet have made the intensive literature search process much more efficient.

Virtually all journals and other research–focused publications are archived online, and many other resources are available to ease the burden of conducting an extensive literature search. A thoughtful and well–conducted meta–analysis can help leverage the increasingly open availability of data from across many sources to address research questions that cannot be fully understood within the limits of a single experiment. Meta–analysis can be especially useful when results from small, independent experiments, such as randomized controlled clinical trials, are individually under–powered to understand treatment effects on their own. In these cases, a meta–analysis can produce an overall treatment effect estimate with more precision and statistical power than estimates produced individually from independent studies. The ideal meta–analysis provides a disciplined, statistically defensible process to summarize research findings. Professionally accepted guidelines for conducting a meta–analysis are helping to make research findings transparent and reproducible and provide the opportunity for other researchers to verify or correct results. Meta–analyses have become so ubiquitous across virtually all disciplines that rely on data. In spite of the capabilities and widespread application of meta–analyses, gaps in the statistical methodology nevertheless exist. This dissertation attempts to address some of these gaps in the context of relatively rare outcomes. From this point, we will assume that an extensive and complete literature review has been conducted, and the studies to be pooled have been appropriately vetted and are sufficiently well–conducted and comparable.

1.2 Common Statistical Methods for Meta–Analyses

For binary outcome medical data, the odds ratio (OR), relative risk (RR), and risk difference (RD) are the three most common effect size estimates to be pooled in a meta–analysis. How to combine these effect size estimates stems from a relatively simple concept. Suppose k studies are to be pooled in a meta–analysis. A weighted average of the individual study's effect size estimates $\hat{\theta}_i$, $i = 1, ..., k$ is computed, yielding one combined

effect size estimate

$$\hat{\theta}_c = \frac{\sum_{i=1}^{k} \left(\hat{w}_i \hat{\theta}_i \right)}{\sum_{i=1}^{k} \hat{w}_i},$$

with a corresponding standard error

$$\text{SE}\left(\hat{\theta}_c \right) = \sqrt{\frac{1}{\sum_{i=1}^{k} \hat{w}_i}}.$$

A weighted average is used since some studies may have a relatively large, more representative sample and should intuitively be given more weight than studies with a relatively small sample size. Studies with larger sample sizes are considered to be more precise and carry more weight in the analysis.

A straightforward way of accounting for study precision is to weight each study by the inverse of its variance σ_i^2, as the variance is a direct reflection of a study's precision. Since the true variance for each study is unknown, σ_i^2 is estimated by $\hat{\sigma}_i^2$. Fittingly, this framework, where studies with larger variability are given less weight, and studies with less variability are given more weight, is known as the inverse variance method (INV). Under the INV framework, the weights for each study are defined as

$$\hat{w}_i^{INV} = \frac{1}{\hat{\sigma}_i^2}.$$

The INV method is a fixed–effect method. Fixed–effect methods assume all studies' effect sizes are homogeneous and estimate a common treatment effect θ. Any variation in the treatment effect is assumed to be due to sampling error alone. These assumptions of fixed-effect methods, however, are rarely met in practice. Often, a random–effects method is more appropriate. Random–effects methods are less strict than fixed–effect methods in that they allow the true treatment effect θ to vary across studies. Random-effects methods assume the effect sizes across studies are heterogeneous and sampled from a distribution of

population effects. These assumptions allow researchers to make conclusions that can be generalized beyond the specific studies included in the meta–analysis.

Since the studies' effect sizes are assumed to come from a distribution, random–effects methods consider two sources of variability: variability due to sampling error σ_i^2 and variability due to between–study differences τ^2, known as the heterogeneity parameter. The heterogeneity parameter reflects the variance of the individual effect size estimates $\hat{\theta}_i$ around the true treatment effect θ. As such, heterogeneity is present when treatment event rates vary significantly across studies, indicating the studies are not estimating a common treatment effect, but rather a distribution of treatment effects. Heterogeneity often arises from differences in participants, administration of the treatment, study design, and measurement of the outcome. It should also be noted that τ^2 is not estimated for each study; rather, one estimate of τ^2 is used to capture the variability in the treatment effect across all studies.

While sampling error and some heterogeneity between studies is expected, if a relatively large amount of between–study variability exists, the results of a fixed–effect meta–analysis can be biased and unreliable. We refer to this difference in variability due to treatment as heterogeneity. Hunter and Schmidt (2000) explain that using a fixed–effect model in the presence of heterogeneity among studies leads to type I error rates that highly exceed the nominal rate and to overly narrow confidence intervals that mistakenly over–state the precision of the results. Thus, when there is variation in the treatment effect that is not simply due to chance, a random–effects model should be used over the fixed–effect model.

To determine if there is variation in the treatment effect not due to chance, the non-parametric Cochran's Q test is commonly used (Cochran, 1954). This test determines if the proportion of events is the same in both treatment groups. The null hypothesis for this test is that there is no meaningful heterogeneity among studies, indicated by both

treatment groups having similar event rates. Additionally, Q follows a chi–square distribution with $k-1$ degrees of freedom, and rejecting the null hypothesis is evidence that a random–effects model should be used over a fixed–effect model. It is well known that the Q test is under–powered when k is relatively small and over–powered when k is relatively large (Huedo-Medina et al., 2006), so the Q test should not be solely relied upon to assess the presence of heterogeneity.

While the Q test is known to have power–related issues, the Q statistic is still routinely used. The Q statistic is calculated by summing the squared deviations of each study's effect size estimate from the combined fixed–effect estimate and weighting those deviations by the inverse of each study's variance

$$Q = \sum_{i=1}^{k} \hat{w}_i^{INV} \left(\hat{\theta}_i - \hat{\theta}_c\right)^2.$$

The I^2 statistic is one such statistic that relies on the Q statistic (Higgins and Thompson, 2002). While the Q statistic helps determine if heterogeneity is present, the I^2 statistic aims to quantify the amount of heterogeneity present. I^2 is defined as

$$I^2 = \max\left\{0\%, \frac{Q-(k-1)}{Q} \times 100\%\right\}.$$

One main advantage of using the I^2 statistic lies in its interpretation. It can be directly interpreted as the percentage of all variability that is due to true heterogeneity and not due to sampling error. Thus, an I^2 of 0% means there is no between–study variability, an I^2 of 100% means all variability in the studies is not due to chance, but to heterogeneity, and an I^2 of 50% means that half the variability is due to heterogeneity (Higgins and Thompson, 2002).

The DerSimonian and Laird random–effects method is another example of where the Q statistic is utilized. The DerSimonian and Laird method (DSL) is similar in concept to the INV method, and it is the most commonly used method in the medical field for

binomial outcomes (DerSimonian and Laird, 1986; Shuster and Walker, 2016). The DSL method estimates τ^2 as

$$\hat{\tau}^2 = \max\left\{0, \frac{Q-(k-1)}{c}\right\},$$

where c is defined as

$$c = \sum_{i=1}^{k} \hat{w}_i^{INV} - \frac{\sum_{i=1}^{k}\left(\hat{w}_i^{INV}\right)^2}{\sum_{i=1}^{k}\hat{w}_i^{INV}}.$$

The weights for the DSL method are similar to the weights of the INV method, but the DSL weights allow for between–study differences. Thus, the weights are defined as

$$\hat{w}_i^{DSL} = \frac{1}{\hat{\sigma}_i^2 + \hat{\tau}^2}.$$

It is easy to see that when $\hat{\tau}^2 = 0$, the fixed–effect and random–effects models will produce the same results. As $\hat{\tau}^2$ increases, the weights for each study will be smaller than the weights under the fixed–effect framework. This causes the random–effects model to be more conservative than the fixed–effect model, which can be problematic if k is relatively small or high precision is needed. Additionally, when $\hat{\tau}^2$ increases, the study weights will be more similar to each other than in the fixed–effect model. Thus, less weight will be given to larger studies, and more weight will be given to smaller studies.

1.3 Common Statistical Methods for Meta–Analyses of Rare Events

Both the INV and the DSL methods rely on large–sample approximations. These assumptions may not hold for small or sparse samples. Such circumstances arise often in health and medical research; for example, in epidemiology studies with rare outcomes or in clinical trials evaluating uncommon adverse events. Vandermeer et al. (2009) conducted a

random sample of 500 Cochrane systematic reviews and found that over 50% of these meta–analyses contained studies with rare events (an observed frequency of less than 5%).

For small or sparse samples, conventional meta-analysis methods, such as the INV fixed-effect method and DSL random-effects method, often fail to produce reliable estimates (or any estimates at all). If there are zero observed events in one or both treatment arms, commonly used effect sizes, like the odds ratio and relative risk, compute to either zero or infinity, and their standard errors are undefined, making it impossible to pool these studies. Existing methods generally do not reasonably account for studies in which there are zero observed events in both arms. By default, software packages typically exclude such studies from a meta–analysis, with some arguing they are unable to provide information on the direction and magnitude of the effect size (Whitehead and Whitehead, 1991; Sweeting et al., 2004; Bradburn et al., 2007; Higgins and Green, 2011, chap. 16.9.3). Others recommend including information from these studies to avoid inflating the magnitude of the overall effect size estimate and to ensure that the estimated treatment effect is generalizable (Sankey et al., 1996; Bhaumik et al., 2012; Kuss, 2015). Studies in which there are zero observed events in one treatment arm (but not both) are also often simply implicitly excluded from a meta–analysis by software programs that assign them zero weight. This practice is demonstrably biased, as these studies clearly contain information about the treatment effect (Böhning et al., 2015).

A common alternative to study exclusion is a continuity correction where either all cells or only those cells with zero counts are incremented by 0.5. Some have argued that applying such a correction may lead to biased results and poor coverage (Sweeting et al., 2004; Bradburn et al., 2007; Tian et al., 2009; Cai et al., 2010; and Kuss, 2015). Other corrections have been proposed, with others suggesting a sensitivity analysis to determine which correction method is most appropriate (Sweeting et al., 2004). Unfortunately, in settings involving rare events, the relative performance of these options varies widely on a case–specific basis (Sweeting et al., 2004; Liu et al., 2014).

The *Cochrane Handbook for Systematic Reviews of Interventions* notes that all these commonly used approaches perform poorly when studying rare events, yielding biased results, overly wide confidence intervals, and low statistical power, concluding that inverse variance methods should be avoided with rare events, including the DerSimonian and Laird random-effects method (Higgins and Green, 2011, chap. 16.9.5). In DerSimonian and Laird (1986)'s defense, they actually comment that the assumptions made for their model may not hold for small sample sizes. If these methods are known to produce unreliable estimates for meta–analyses with rare events (even warned by the developers themselves), how is it that the DerSimonian and Laird method is the most commonly used method for meta–analyses in the medical field? Shuster and Walker (2016) attempt to answer this question by noting that common meta–analysis software packages issue no warnings when studies have rare events. They also point out that the DerSimonian and Laird method is the most common random–effects method available in software. Much work is needed to help researchers be aware of the danger of using inappropriate models for meta–analyses with rare events and to provide alternative methods for researchers faced with such data.

One alternative to the common inverse variance methods that is available in most software packages is the fixed–effect Mantel–Haenszel method (MH) for discrete data (Mantel and Haenszel, 1959). While the DerSimonian and Laird random–effects method is the most commonly used method for meta–analyses of binary outcomes, the MH method is the most commonly used fixed–effect method in this scenario (Bhaumik et al., 2012). Consider Table 1.1 as a way to summarize the data of the i^{th} study in a meta–analysis.

TABLE 1.1: Typical 2 x 2 table structure for binary outcome data used in a meta–analysis.

Study (i)	Event	Non–Event	Total
Treatment	a_i	c_i	$a_i + c_i$
Control	b_i	d_i	$b_i + d_i$
Total	$a_i + b_i$	$c_i + d_i$	n_i

The MH weights $\hat{w}_i^{MH}$ for each study are defined differently for each effect size

$$\hat{w}_i^{MH} = \frac{b_i c_i}{n_i} \text{ for the OR,}$$
$$\hat{w}_i^{MH} = \frac{b_i(a_i + c_i)}{n_i} \text{ for the RR, and}$$
$$\hat{w}_i^{MH} = \frac{(a_i + c_i)(b_i + d_i)}{n_i} \text{ for the RD.}$$

The standard error of the combined effect size estimate also depends on the effect size

$$\text{SE}_{ln\left(\widehat{OR}_c^{MH}\right)} = \sqrt{\frac{\sum_{i=1}^{k}\left(\frac{a_i d_i}{n_i}\frac{a_i+d_i}{n_i}\right)}{2\left(\sum_{i=1}^{k}\frac{a_i d_i}{n_i}\right)^2} + \frac{\sum_{i=1}^{k}\left(\frac{b_i c_i}{n_i}\frac{a_i+d_i}{n_i} + \frac{b_i+c_i}{n_i}\frac{a_i d_i}{n_i}\right)}{2\sum_{i=1}^{k}\left(\frac{a_i d_i}{n_i}\right)\sum_{i=1}^{k}\left(\frac{b_i c_i}{n_i}\right)} + \frac{\sum_{i=1}^{k}\left(\frac{b_i c_i}{n_i}\frac{b_i+c_i}{n_i}\right)}{2\left(\sum_{i=1}^{k}\frac{b_i c_i}{n_i}\right)^2}},$$

$$\text{SE}_{ln\left(\widehat{RR}_c^{MH}\right)} = \sqrt{\frac{\sum_{i=1}^{k}\left(\frac{(a_i+c_i)(b_i+d_i)(a_i+b_i)-(a_i b_i n_i)}{n_i^2}\right)}{\sum_{i=1}^{k}\left(\frac{a_i(b_i+d_i)}{n_i}\right)\sum_{i=1}^{k}\left(\frac{b_i(a_i+c_i)}{n_i}\right)}}, \text{ and}$$

$$\text{SE}_{\widehat{RD}_c^{MH}} = \frac{\sqrt{\sum_{i=1}^{k}\left(\frac{a_i c_i(b_i+d_i)^2 + b_i d_i(a_i+c_i)^3}{(a_i+c_i)(b_i+d_i)n_i^2}\right)}}{\sum_{i=1}^{k}\left(\frac{(a_i+c_i)(b_i+d_i)}{n_i}\right)}.$$

Bradburn et al. (2007) show that this method performs better than the inverse variance method when events are rare. Additionally, they demonstrate that the MH method actually performs better without the use of a continuity correction when events are rare, but Sankey et al. (1996) show this appears to hold true only when there is almost no study heterogeneity. If there are zero cells across all studies in one or more treatment arm, a continuity correction must be used, and performance is weakened. In this case, the results may be biased, the method may have low statistical power, and it may produce a conservative estimate (Bradburn et al., 2007). The MH method also relies on a normal approximation, which is generally not appropriate for discrete data and small samples.

Another alternative method for discrete data with rare events is the Peto method (Yusuf et al., 1985). This method can be used to combine only odds ratios, and it does not require a continuity correction unless both treatment arms have zero events. The OR and standard error of the OR for the individual studies are computed differently than in previous methods. They are defined as

$$\widehat{OR}_i = \exp\left\{\frac{O_i - E_i}{V_i}\right\} \text{ and}$$
$$\text{SE}_{ln(\widehat{OR}_i)} = \sqrt{\frac{1}{V_i}},$$

where

$$O_i = a_i,$$
$$E_i = \frac{(a_i + b_i)(a_i + c_i)}{n_i}, \text{ and}$$
$$V_i = \frac{(a_i + b_i)(c_i + d_i)(a_i + c_i)(b_i + d_i)}{n_i^2(n_i - 1)}.$$

The weights for each study are

$$\hat{w}_i^{Peto} = V_i.$$

The combined effect size estimate and its associated standard error is

$$\widehat{OR}_c^{Peto} = \exp\left\{\frac{\sum_{i=1}^k O_i - \sum_{i=1}^k E_i}{\sum_{i=1}^k \hat{w}_i^{Peto}}\right\} \text{ and}$$
$$\text{SE}_{ln(\widehat{OR}_{Peto})} = \sqrt{\frac{1}{\sum_{i=1}^k \hat{w}_i^{Peto}}}.$$

Bradburn et al. (2007) show the Peto method works well (often better than the MH method) when these criteria are met: the treatment effects are small, the events are fairly rare, and the studies have similar sample sizes in the treatment and control groups. If these conditions are not met, the Peto method tends to underestimate the treatment effect

(Greenland and Salvan, 1990). Greenland and Salvan (1990), however, state that this bias "is negligible in meta–analysis of randomized trials involving small effects and a reasonable number of outcome events." Furthermore, if these conditions are met, this method performs well for rare event data, and Bradburn et al. (2007) found it to be one of the least biased and most powerful methods they studied (including the inverse variance methods and the MH method). However, like the MH method, this method relies on a normal approximation, which can cause undesirable results when events are rare, and studies are small. Kuss (2015) states that this method may not be the best for very sparse data, even though it is recommended in the Cochrane Handbook (Higgins and Green, 2011, chap. 16.9.5).

1.4 Novel Statistical Methods for Meta–Analyses of Rare Events

While the MH and Peto methods may outperform the inverse variance methods, they still leave much to be desired in methods for sparse data and rare events. The large–sample assumptions of these common methods are ill–suited for discrete data with rare events. Additionally, a common issue hampering both alternatives is non–negligible heterogeneity. Several methods have been proposed to fill this gap, and recent interest in meta–analysis for rare events has been growing (Cai et al., 2010; Bhaumik et al., 2012; Moreno et al., 2012; Liu et al., 2014). We will review a few of these here.

A recent fixed–effect method by Liu et al. (2014), which builds on the work of Tian et al. (2009) and Singh et al. (2005), involves combining p–value functions (as opposed to combining effect size estimates) which are obtained from the mid–p adaption of Fisher's exact test to obtain one overall p–value function. p–value functions, often referred to as significance functions, are a type of confidence distribution. Confidence distributions are sample–dependent distributions that represent confidence intervals for all levels of the

effect size. A p–value function produces the one–sided p–value for the null hypothesis and the one–sided p–values for every alternative to the null hypothesis for the effect size. Providing information about both the size and precision of the effect size estimate, p–value functions contain richer information than p–values alone. This method does not rely on large–sample approximations, nor does it require any arbitrary continuity correction. This method also allows a wide range of options for combining studies, including the choice of weights, effect sizes, transformation functions (which is used in combining studies), exact tests, and adjustments to individual study's p–value functions. One drawback to this method is that it does not attempt to address heterogeneity.

Another method for rare event data is a random–effects model based on generalized linear mixed models proposed by Stijnen et al. (2015). They recognize the large–sample assumptions of a traditional random–effects model do not hold when data is sparse, or events are rare, so they replace these normality assumptions for the within–study likelihood with exact likelihoods. They suggest that in traditional models, the effect size estimate and the corresponding weights (due to the incorporation of the standard error) are correlated, and failing to take this correlation into account can bias results. This assessment is similar to arguments made by Shuster et al. (2012) and Bhaumik et al. (2012). Their proposed methods avoid this bias, and there is also no need for a continuity correction. However, they point out that their approach is not easily utilized when the risk difference or the relative risk are the effect sizes to be pooled, as opposed to the odds ratio. Additionally, this approach is not an "exact" approach, as it still relies on asymptotic assumptions outside the within–study exact likelihood.

Another regression–based approach, developed by Moreno et al. (2012), gives larger studies more weight than both the traditional fixed–effect and random–effects methods, with the idea being that smaller studies are more prone to small–study effects and should be heavily down–weighted. Additionally, this method adds study precision as a covariate into the regression model. The regression line is then extrapolated to predict the effect

size of a hypothetical study with an infinite sample size. A study of infinite size reflects an effect size that has an associated standard error of zero. The rationale behind this approach is that large studies, especially one of infinite size, are less likely to reflect biases caused by small–study effects.

Bhaumik et al. (2012) developed a random–effects simple average approach, along with a new heterogeneity estimator, that works well with rare event data even where there is heterogeneity among studies. They argue (as does Shuster et al. (2012) and Stijnen et al. (2015)) that the study weights, in addition to the effect sizes themselves, are random and that the weights are correlated with the effect sizes, which introduces bias. They suggest this unweighted estimate can be viewed as a bias–corrected weighted estimate with the bias proportional to the rarity of the event. In addition to the heterogeneity estimator by Bhaumik et al. (2012), other estimators have recently been proposed (Paule and Mandel, 1982; Sidik and Jonkman, 2005; Sidik and Jonkman, 2007). Unfortunately, this simple–average method still relies on large–sample approximations and a continuity correction. Additionally, the authors present a very interesting figure from their paper showing a plot of type I error rates for several methods when there is a large amount of heterogeneity present in the data (Bhaumik et al. (2012) Figure 2 (a)). Strikingly, the type I error rates of the Mantel–Haenszel and inverse variance methods approach 80% under their simulation conditions! This is further evidence that these commonly used methods are inaccurate for meta-analyses with rare events and heterogeneity.

There are also Bayesian methods that could be used for meta–analyses with sparse data (T. C. Smith, 1995; Cai et al., 2010; Kim et al., 2016). The distinction in Bayesian meta–analyses is that Bayesian methods assign prior distributions to the parameters of interest (e.g., the treatment effect or the heterogeneity parameter) and update these distributions based on the data. Bayesian methods are becoming increasingly popular – especially in the context of network meta–analysis. However, we will not discuss these methods here in detail, as the focus is on frequentist methods. Additionally, Kuss (2015)

shows via simulation studies that the choice of a prior distribution, even if considered noninformative, has a large impact on the results when data are sparse, suggesting that these methods may not be desirable for rare events and sparse data.

1.5 Summary of the Remaining Chapters

With this recent emphasis on meta–analyses of rare events, it is crucial to develop reliable methods that perform well with sparse data and in the presence of heterogeneity. Many of the methods summarized here, though designed for rare event data, are severely hampered when heterogeneity is present.

In Chapter 2, we propose an exact permutation–based method for meta–analysis that is adjusted for heterogeneity but does not involve estimating the extent of heterogeneity. Our proposed exact method does not rely on large–sample approximations or continuity corrections for rare event data. We assess our method via a simulation study and by comparing our exact method to several existing methods that are readily available in software.

Additionally, in Chapter 3, we apply our method to network meta–analysis, where several treatments are independently compared to a common control, but never compared to each other. Network meta-analysis inherits the same problems as traditional meta-analysis, so network meta-analyses with adverse events are likely to be crippled by heterogeneity. We consider the utility of our method with a simulation study.

In Chapter 4, we focus on the method of combining confidence distributions by Liu et al. (2014). The use of confidence distributions is growing rapidly, and with several different approaches in the literature, we conduct a simulation study to help determine which method performs best for meta–analyses with heterogeneity and rare events.

Additionally, we extend current methods to create a confidence distribution approach that works well when events are rare, and heterogeneity is present.

CHAPTER 2

EXACT META–ANALYSIS FOR RARE EVENTS AND HETEROGENEITY

2.1 Background

A method that has been used extensively for correlated categorical data is an exact permutation–based approach developed by Corcoran et al. (2001). This method involves conditioning on sufficient statistics in an exponential family model to eliminate the nuisance parameter corresponding to within–study variability. This yields exact inference for the treatment effect in the presence of within–study correlation. We can extend this method to allow for variation of treatment level within study so that this method can be applied in the setting of a meta-analysis. Recognizing that heterogeneity reflects the difference in event rates between treatment groups, we make another extension by conditioning relative to additional nuisance parameters to account for within–study correlation by treatment: one parameter which captures the control group correlation effects, the other parameter which captures the treatment group correlation effects.

We first summarize the permutation–based approach for correlated binary data, as proposed by Corcoran et al. (2001), and then suggest the necessary modifications for a meta–analysis. These modifications allow the treatment group to vary within studies and the correlation structure to vary among treatment groups in order to adjust for non-negligible heterogeneity in the data. For all three of these methods, we outline the network algorithm used for computational purposes. We also use an exact estimation procedure, as described by Mehta and Patel (1995), to obtain a point estimate and its $(1-\alpha)$ level confidence interval, as opposed to just a p–value. We evaluate our method via a simulation study and by comparing our exact method to other methods available in software.

2.2 Methodology

As an initial framework, we will first present the exact conditional logistic regression model, assuming independence. This assumption, however, is likely not valid for meta–analysis data as participants within a study are more likely to be correlated with each other than with participants from another study. As such, we will then introduce an exact conditional logistic regression model for correlated data. This model is the basis for our extensions and innovations. Finally, we present an exact estimation method used for calculating point estimates and confidence intervals.

2.2.1 Exact Conditional Logistic Regression

When considering exact conditional logistic regression for independent data, suppose a systematic review produces k independent studies with binary responses for a meta-analysis. Let Y_{ij} represent the binary response of the j^{th} $(j = 1, 2, \ldots, n_i)$ participant in the i^{th} $(i = 1, 2, \ldots, k)$ study such that $Y_{ij} = 1$ if participant j in study i had an "event," and $Y_{ij} = 0$ if participant j in study i did not have an "event." Let $\Pr(Y_{ij} = 1) = \pi_i$ and $\Pr(Y_{ij} = 0) = 1 - \pi_i$, and let x_i indicate the treatment group for all participants in the i^{th} study. Then,

$$Y_{ij} \sim \text{Bernoulli}(\pi_i),$$

and

$$\Pr(Y_{ij} = y_{ij}) = \pi_i^{y_{ij}}(1 - \pi_i)^{1-y_{ij}}.$$

The likelihood, or joint probability of the observed responses, is

$$\mathrm{L}(\pi_i) = \prod_{j=1}^{n_i} \pi_i^{y_{ij}} (1-\pi_i)^{1-y_{ij}}.$$

The log likelihood is then

$$\begin{aligned} l(\pi_i) &= \log(\mathrm{L}(\pi_i)) \\ &= \log\left(\prod_{j=1}^{n_i} \pi_i^{y_{ij}} (1-\pi_i)^{1-y_{ij}}\right) \\ &= \sum_{j=1}^{n_i} \log\left(\pi_i^{y_{ij}} (1-\pi_i)^{1-y_{ij}}\right) \\ &= \sum_{j=1}^{n_i} y_{ij} \log(\pi_i) + \sum_{j=1}^{n_i} (1-y_{ij}) \log(1-\pi_i) \\ &= \sum_{j=1}^{n_i} y_{ij} \log\left(\frac{\pi_i}{1-\pi_i}\right) + \sum_{j=1}^{n_i} \log(1-\pi_i). \end{aligned}$$

Since we have a binary response, we can use logistic regression to model the dependency of π_i on x_i through the relationship

$$\log\left(\frac{\pi_i}{1-\pi_i}\right) = \alpha + \beta x_i,$$

where α and β are unknown parameters. Solving for π_i yields

$$\pi_i = \frac{\exp\{\alpha + \beta x_i\}}{1 + \exp\{\alpha + \beta x_i\}}.$$

We can now reparameterize the log likelihood in terms of α, a nuisance parameter, and β, the treatment effect quantified as a log odds ratio,

$$\begin{aligned} l(\alpha,\beta) &= \sum_{j=1}^{n_i} y_{ij}\left(\alpha+\beta x_i\right) + \sum_{j=1}^{n_i} \log\left(1-\frac{\exp\{\alpha+\beta x_i\}}{1+\exp\{\alpha+\beta x_i\}}\right) \\ &= \sum_{j=1}^{n_i} y_{ij}\left(\alpha+\beta x_i\right) + \sum_{j=1}^{n_i} \log\left(\frac{1}{1+\exp\{\alpha+\beta x_i\}}\right) \\ &= \sum_{j=1}^{n_i} y_{ij}\left(\alpha+\beta x_i\right) - \sum_{j=1}^{n_i} \log\left(1+\exp\{\alpha+\beta x_i\}\right). \end{aligned}$$

Exponentiating to get the reparameterized likelihood gives

$$\begin{aligned} \mathrm{L}(\alpha,\beta) &= \exp\left\{\sum_{j=1}^{n_i} y_{ij}\left(\alpha+\beta x_i\right) - \sum_{j=1}^{n_i} \log\left(1+\exp\{\alpha+\beta x_i\}\right)\right\} \\ &= \exp\left\{\sum_{j=1}^{n_i} y_{ij}\left(\alpha+\beta x_i\right)\right\} \exp\left\{-\sum_{j=1}^{n_i} \log\left(1+\exp\{\alpha+\beta x_i\}\right)\right\} \\ &= \frac{\exp\left\{\sum_{j=1}^{n_i} y_{ij}\left(\alpha+\beta x_i\right)\right\}}{\exp\left\{\sum_{j=1}^{n_i} \log\left(1+\exp\{\alpha+\beta x_i\}\right)\right\}} \\ &= \frac{\exp\left\{\sum_{j=1}^{n_i} y_{ij}\left(\alpha+\beta x_i\right)\right\}}{\exp\left\{\log\prod_{j=1}^{n_i}\left(1+\exp\{\alpha+\beta x_i\}\right)\right\}} \\ &= \frac{\exp\left\{\sum_{j=1}^{n_i} y_{ij}\left(\alpha+\beta x_i\right)\right\}}{\prod_{j=1}^{n_i}\left(1+\exp\{\alpha+\beta x_i\}\right)}. \end{aligned} \tag{2.1}$$

Traditionally, we maximize the unconditional likelihood, Equation 2.1, with respect to α and β to make inference about these parameters. An alternative approach is to make use of the conditional likelihood function. If we are interested in making inferences about β, then, instead of estimating α, which we consider a nuisance parameter, we can condition on the observed value of its sufficient statistic to eliminate this parameter from the model.

Thus, the conditional likelihood reduces to

$$\mathrm{L}(\alpha, \beta | z_i) = \frac{\exp\left\{\sum_{j=1}^{n_i} \beta y_{ij} x_i\right\}}{\prod_{j=1}^{n_i}\left(1 + \exp\{\beta x_i\}\right)}. \tag{2.2}$$

From here, there are two ways to make inference about β: asymptotic and exact. The asymptotic method maximizes the conditional likelihood function, Equation 2.2, with respect to β. The exact method is based on the permutational distribution of β's sufficient statistic.

2.2.2 Exact Conditional Logistic Regression for Correlated Data

With this background, we will now describe exact conditional logistic regression for correlated data. This is an appropriate framework for meta–analysis data since participants in the same study are likely to be more alike, or correlated, than participants from a different study. When data are correlated, or clustered, the correlation among observations within cluster causes overdispersion, or extra–binomial variation. This extra variability must be accounted for in order to obtain an accurate p–value. Failing to account for this extra variation usually results in an artificially small p–value.

To start, let $Z_i = \sum_{j=1}^{n_i} Y_{ij}$ represent the total number of events among the n_i participants in the i^{th} study, and let x_i indicate the treatment group for all participants in the i^{th} study. The probability mass function of $\boldsymbol{Y_i} = (Y_{i1}, Y_{i2}, \ldots, Y_{in_i})'$ for correlated data as derived by Molenberghs and Ryan (1999) is

$$\Pr(\boldsymbol{Y_i} = \boldsymbol{y_i}) = \exp\{\theta_i z_i - \delta_i z_i (n_i - z_i) + A_i(\theta_i, \delta_i)\}, \tag{2.3}$$

where δ_i is the dispersion parameter, and $A_i(\theta_i, \delta_i)$ is the normalizing constant, summing over all possible realizations of $\boldsymbol{Y_i}$. δ_i is used to reflect the extent of the correlation among

participants in the i^{th} study. When there is no correlation among participants in each study ($\delta_i = 0$ for all i), Equation 2.3 reduces to a product of independent binary probabilities. Using the logit link for $\theta_i = \alpha + \beta x_i$, where, again, x_i is constant for all participants in the i^{th} study, and assuming equal dispersion among clusters ($\delta_i = \delta$ for all i), the probability mass function of $\boldsymbol{Y_i}$ is expressed as

$$P(\boldsymbol{Y_i} = \boldsymbol{y_i}) = \exp\{(\alpha + \beta x_i)z_i - \delta z_i(n_i - z_i) + A_i(\alpha, \beta, \delta)\}. \tag{2.4}$$

Assuming exchangeability of the binary responses within a given study, the probability mass function for each study depends on the individual responses $\boldsymbol{Y_i}$ only through their sum Z_i. Any permutation of the binary responses within $\boldsymbol{Y_i}$ yields the same probability. The conditional distribution of Z_i can, therefore, be expressed as

$$P(Z_i = z_i|x_i) = \binom{n_i}{z_i}\exp\{(\alpha + \beta x_i)z_i - \delta z_i(n_i - z_i) + A_i(\alpha, \beta, \delta)\},$$

where $z_i = 0, \ldots, n_i$, and $\binom{n_i}{z_i}$ is the number of distinct vectors $\boldsymbol{y_i}$ that give the same value of $z_i = \sum_{j=1}^{n_i} y_{ij}$. Assuming study independence, the joint probability mass function for $\boldsymbol{Z} = (Z_1, Z_2, \ldots, Z_k)'$, a vector representing the total number of events among the n_i participants in each of the i studies, given $\boldsymbol{x} = (x_1, x_2, \ldots, x_k)'$, a vector indicating the

treatment group for all participants for each of the i studies, is

$$\begin{aligned}
\Pr(\boldsymbol{Z}=\boldsymbol{z}|\boldsymbol{x}) &= \prod_{i=1}^{k}\binom{n_i}{z_i}\exp\{(\alpha+\beta x_i)z_i-\delta z_i(n_i-z_i)+A_i(\alpha,\beta,\delta)\} \qquad (2.5)\\
&= \left[\prod_{i=1}^{k}\binom{n_i}{z_i}\right]\prod_{i=1}^{k}\exp\{(\alpha+\beta x_i)z_i-\delta z_i(n_i-z_i)+A_i(\alpha,\beta,\delta)\}\\
&= \left[\prod_{i=1}^{k}\binom{n_i}{z_i}\right]\exp\left\{\sum_{i=1}^{k}\left[(\alpha+\beta x_i)z_i-\delta z_i(n_i-z_i)+A_i(\alpha,\beta,\delta)\right]\right\}\\
&= \left[\prod_{i=1}^{k}\binom{n_i}{z_i}\right]\exp\left\{\sum_{i=1}^{k}(\alpha+\beta x_i)z_i-\sum_{i=1}^{k}\delta z_i(n_i-z_i)+\sum_{i=1}^{k}A_i(\alpha,\beta,\delta)\right\}\\
&= \left[\prod_{i=1}^{k}\binom{n_i}{z_i}\right]\exp\left\{\sum_{i=1}^{k}(\alpha z_i+\beta x_i z_i)-\sum_{i=1}^{k}\delta z_i(n_i-z_i)+\sum_{i=1}^{k}A_i(\alpha,\beta,\delta)\right\}\\
&= \left[\prod_{i=1}^{k}\binom{n_i}{z_i}\right]\exp\left\{\alpha\sum_{i=1}^{k}z_i+\beta\sum_{i=1}^{k}x_i z_i-\delta\sum_{i=1}^{k}z_i(n_i-z_i)+\sum_{i=1}^{k}A_i(\alpha,\beta,\delta)\right\}\\
&= \left[\prod_{i=1}^{k}\binom{n_i}{z_i}\right]\exp\left\{\alpha s^{\alpha}(\boldsymbol{z})+\beta t(\boldsymbol{z})-\delta s^{\delta}(\boldsymbol{z})+\sum_{i=1}^{k}A_i(\alpha,\beta,\delta)\right\}, \qquad (2.6)
\end{aligned}$$

where $\boldsymbol{z}=(z_1,z_2,\ldots z_k)'$. As this density is also of the exponential family, the sufficient statistics are

$$\boldsymbol{s}=\begin{cases} s^{\alpha}(\boldsymbol{z})=\sum_{i=1}^{k}z_i & \text{for } \alpha\\ t(\boldsymbol{z})=\sum_{i=1}^{k}x_i z_i & \text{for } \beta\\ s^{\delta}(\boldsymbol{z})=\sum_{i=1}^{k}z_i(n_i-z_i) & \text{for } \delta. \end{cases}$$

Since we are interested in testing $H_0:\beta=0$ against $H_a:\beta>0$, we eliminate the nuisance parameters α and δ to obtain the exact permutational distribution of $\boldsymbol{Z}$ under H_0 by conditioning on $s^{\alpha}(\boldsymbol{z})$ and $s^{\delta}(\boldsymbol{z})$. Define the conditional reference set $\Gamma(s^{\alpha}(\boldsymbol{z}),s^{\delta}(\boldsymbol{z}))$ such that

$$\Gamma(s^{\alpha}(\boldsymbol{z}),s^{\delta}(\boldsymbol{z}))=\left\{\boldsymbol{z}^{*}:s^{\alpha}(\boldsymbol{z}^{*})=s^{\alpha}(\boldsymbol{z}),s^{\delta}(\boldsymbol{z}^{*})=s^{\delta}(\boldsymbol{z})\right\},$$

where $\boldsymbol{z}^*$ is any generic table, like Table 2.1, of the form $\boldsymbol{z}^* = (z_1^*, z_2^*, \ldots, z_k^*)'$ that results in the observed values of $s^\alpha(\boldsymbol{z})$ and $s^\delta(\boldsymbol{z})$, with the study sizes n_m held constant. Then, the conditional probability of $\boldsymbol{Z}$, given the observed values of $s^\alpha(\boldsymbol{z})$ and $s^\delta(\boldsymbol{z})$, is

$$\begin{aligned}\Pr(\boldsymbol{Z} = \boldsymbol{z}|\boldsymbol{x}; s^\alpha(\boldsymbol{z}), s^\delta(\boldsymbol{z})) &= \frac{\Pr(\boldsymbol{Z} = \boldsymbol{z}|\boldsymbol{x})}{\sum_{\boldsymbol{z}^* \in \Gamma(s^\alpha(\boldsymbol{z}), s^\delta(\boldsymbol{z}))} \Pr(\boldsymbol{Z} = \boldsymbol{z}^*|\boldsymbol{x})} \\ &= \frac{\left[\prod_{i=1}^{k} \binom{n_i}{z_i}\right] \exp\{\beta t(\boldsymbol{z})\}}{\sum_{\boldsymbol{z}^* \in \Gamma(s^\alpha(\boldsymbol{z}), s^\delta(\boldsymbol{z}))} \left[\prod_{i=1}^{k} \binom{n_i}{z_i^*}\right] \exp\{\beta t(\boldsymbol{z}^*)\}}. \end{aligned} \tag{2.7}$$

Under H_0, this reduces to

$$\Pr(\boldsymbol{Z} = \boldsymbol{z}|\boldsymbol{x}; s^\alpha(\boldsymbol{z}), s^\delta(\boldsymbol{z}), H_0) = \frac{\prod_{i=1}^{k} \binom{n_i}{z_i}}{\sum_{\boldsymbol{z}^* \in \Gamma(s^\alpha(\boldsymbol{z}), s^\delta(\boldsymbol{z}))} \prod_{i=1}^{k} \binom{n_i}{z_i^*}}, \tag{2.8}$$

which is free of all unknown parameters.

TABLE 2.1: Generic contingency table for a binary outcome with treatment constant per study.

Study (i)	1	2	...	k	Sum
Treatment (x_i)	x_1	x_2	...	x_k	
z_i	z_1	z_2	...	z_k	s^α
$n_i - z_i$	$n_1 - z_1$	$n_2 - z_2$	...	$n_k - z_k$	
n_i	n_1	n_2	...	n_k	N
$x_i z_i$	$x_1 z_1$	$x_2 z_2$	...	$x_k z_k$	t
$z_i(n_i - z_i)$	$z_1(n_1 - z_1)$	$z_2(n_2 - z_2)$	...	$z_k(n_k - z_k)$	s^δ

To conduct an exact test, we can order the tables in $\Gamma(s^\alpha(\boldsymbol{z}), s^\delta(\boldsymbol{z}))$ according to

their respective values of the trend statistic $t(\boldsymbol{z}) = \sum_{i=1}^{k} x_i z_i$, the sufficient statistic for β, conditional on the observed values of $s^{\alpha}(\boldsymbol{z})$ and $s^{\delta}(\boldsymbol{z})$. From Equation 2.8, the probability under H_0 of observing a table $\boldsymbol{z}_{obs} = (z_1, z_2, \ldots, z_k)'$ with associated sufficient statistics $s^{\alpha}(\boldsymbol{z})$ and $s^{\delta}(\boldsymbol{z})$ and a realization of $t(\boldsymbol{z})$ denoted by $t(\boldsymbol{z}_{obs})$ is

$$\Pr(t(\boldsymbol{z}) \geq t(\boldsymbol{z}_{obs}) | \boldsymbol{x}; s^{\alpha}(\boldsymbol{z}), s^{\delta}(\boldsymbol{z}), H_0) = \sum_{\substack{\boldsymbol{z}^* \in \Gamma(s^{\alpha}(\boldsymbol{z}), s^{\delta}(\boldsymbol{z})): \\ t(\boldsymbol{z}^*) \geq t(\boldsymbol{z}_{obs})}} \left[\frac{\prod_{i=1}^{k} \binom{n_i}{z_i}}{\sum_{\boldsymbol{z}^* \in \Gamma(s^{\alpha}(\boldsymbol{z}), s^{\delta}(\boldsymbol{z}))} \prod_{i=1}^{k} \binom{n_i}{z_i^*}} \right]. \quad (2.9)$$

A one–sided α–level test then rejects H_0 when $t(\boldsymbol{z}_{obs}) > t(\boldsymbol{z}_{\alpha})$, where $t(\boldsymbol{z}_{\alpha})$ is defined as the smallest value such that $\Pr(t(\boldsymbol{z}) > t(\boldsymbol{z}_{\alpha}) | H_0, s^{\alpha}(\boldsymbol{z}), s^{\delta}(\boldsymbol{z})) \leq \alpha$. A two sided p–value can be obtained by doubling Equation 2.9.

The difficulty of this process lies in enumerating all tables in the reference set $\Gamma(s^{\alpha}(\boldsymbol{z}), s^{\delta}(\boldsymbol{z}))$, which is needed to compute the denominator of Equation 2.9. Despite conditioning on the two parameters, $s^{\alpha}(\boldsymbol{z})$ and $s^{\delta}(\boldsymbol{z})$, enumerating $\Gamma(s^{\alpha}(\boldsymbol{z}), s^{\delta}(\boldsymbol{z}))$ explicitly is computationally infeasible for practical purposes, especially when the number of observed events is large. Corcoran et al. (2001) propose a network approach to this problem that builds on the network approach of Mehta et al. (1992) and which allows $\Gamma(s^{\alpha}(\boldsymbol{z}), s^{\delta}(\boldsymbol{z}))$ to be implicitly enumerated. This method is an efficient and practical way to obtain $\Gamma(s^{\alpha}(\boldsymbol{z}), s^{\delta}(\boldsymbol{z}))$ and will be discussed in Section 2.2.4.

2.2.3 Exact Estimation Method

While it is valuable to determine if an effect is present, in a meta–analysis it may be even more important to know the magnitude of that effect. Here, we present an exact estimation procedure, as outlined by Mehta and Patel (1995) based on Section 2.2, and that readily applies to our new method outlined in Section 2.3.2. As described in the

previous sections, exact inference is made possible by conditioning on nuisance parameters and working with the conditional likelihood. This exact conditioning approach can be used for estimation when unconditional asymptotic methods yield unreliable estimates, as is the case when sample sizes are small.

Recall the conditional probability, Equation 2.7, which is repeated here, modified slightly

$$f_\beta(t(\boldsymbol{z}_{obs})|s^\alpha(\boldsymbol{z}), s^\delta(\boldsymbol{z})) = \frac{C(t(\boldsymbol{z}_{obs})|s^\alpha(\boldsymbol{z}), s^\delta(\boldsymbol{z}))\exp\{\beta t(\boldsymbol{z}_{obs})\}}{\sum_{u=\min(t(\boldsymbol{z}))}^{\max(t(\boldsymbol{z}))} C(u|s^\alpha(\boldsymbol{z}), s^\delta(\boldsymbol{z}))\exp\{\beta u\}}, \tag{2.10}$$

where $t(\boldsymbol{z}_{obs})$ is the observed value of the sufficient statistic for β and $C(\cdot)$ is the unnormalized probability. The conditional maximum likelihood estimate (CMLE) of β is the value of β which maximizes Equation 2.10.

However, if $t(\boldsymbol{z}_{obs})$ is at either of the extremes of its distribution (i.e. $t(\boldsymbol{z}_{obs}) = \max(t(\boldsymbol{z}))$ or $t(\boldsymbol{z}_{obs}) = \min(t(\boldsymbol{z}))$), then it is not possible to maximize the conditional probability with respect to β. In these cases, the best we can do is to set $\hat{\beta}$ to be $-\infty$ when $t(\boldsymbol{z}_{obs}) = \min(t(\boldsymbol{z}))$, or ∞, when $t(\boldsymbol{z}_{obs}) = \max(t(\boldsymbol{z}))$. An alternative approach that is not maximum likelihood–based (which is known to be unreliable when data are sparse or the sample size is small) is a median unbiased estimate (MUE). Unlike the CMLE, the MUE can be computed even when $t(\boldsymbol{z}_{obs})$ is at the minimum or maximum of its distribution. The MUE of β satisfies the condition

$$f_\beta(t(\boldsymbol{z}_{obs})|s^\alpha(\boldsymbol{z}), s^\delta(\boldsymbol{z})) = 0.5,$$

To obtain an exact confidence interval for β, we define the left and right tails of the distribution of the test statistic given $s^{\alpha}(\boldsymbol{z})$ and $s^{\delta}(\boldsymbol{z})$ to be

$$\begin{aligned} L_{\beta}(t(\boldsymbol{z}_{obs})) &= \sum_{u^*=\min(t(\boldsymbol{z}))}^{t(\boldsymbol{z}_{obs})} f_{\beta}(u^*|s^{\alpha}(\boldsymbol{z}), s^{\delta}(\boldsymbol{z})) \\ &= \sum_{u^*=\min(t(\boldsymbol{z}))}^{t(\boldsymbol{z}_{obs})} \frac{C(u^*|s^{\alpha}(\boldsymbol{z}), s^{\delta}(\boldsymbol{z}))\exp\{\beta u^*\}}{\sum_{u=\min(t(\boldsymbol{z}))}^{\max(t(\boldsymbol{z}))} C(u|s^{\alpha}(\boldsymbol{z}), s^{\delta}(\boldsymbol{z}))\exp\{\beta u\}} \\ &= \frac{\sum_{u^*=\min(t(\boldsymbol{z}))}^{t(\boldsymbol{z}_{obs})} C(u^*|s^{\alpha}(\boldsymbol{z}), s^{\delta}(\boldsymbol{z}))\exp\{\beta u^*\}}{\sum_{u=\min(t(\boldsymbol{z}))}^{\max(t(\boldsymbol{z}))} C(u|s^{\alpha}(\boldsymbol{z}), s^{\delta}(\boldsymbol{z}))\exp\{\beta u\}}, \end{aligned}$$

and

$$\begin{aligned} R_{\beta}(t(\boldsymbol{z}_{obs})) &= \sum_{u^*=t(\boldsymbol{z}_{obs})}^{\max(t(\boldsymbol{z}))} f_{\beta}(u^*|s^{\alpha}(\boldsymbol{z}), s^{\delta}(\boldsymbol{z})) \\ &= \sum_{u^*=t(\boldsymbol{z}_{obs})}^{\max(t(\boldsymbol{z}))} \frac{C(u^*|s^{\alpha}(\boldsymbol{z}), s^{\delta}(\boldsymbol{z}))\exp\{\beta u^*\}}{\sum_{u=\min(t(\boldsymbol{z}))}^{\max(t(\boldsymbol{z}))} C(u|s^{\alpha}(\boldsymbol{z}), s^{\delta}(\boldsymbol{z}))\exp\{\beta u\}} \\ &= \frac{\sum_{u^*=t(\boldsymbol{z}_{obs})}^{\max(t(\boldsymbol{z}))} C(u^*|s^{\alpha}(\boldsymbol{z}), s^{\delta}(\boldsymbol{z}))\exp\{\beta u^*\}}{\sum_{u=\min(t(\boldsymbol{z}))}^{\max(t(\boldsymbol{z}))} C(u|s^{\alpha}(\boldsymbol{z}), s^{\delta}(\boldsymbol{z}))\exp\{\beta u\}}, \end{aligned}$$

respectively. Let β_- and β_+ be the lower and upper bounds, respectively, of a two–sided $(1-\alpha)\%$ confidence interval for β. Then β_- satisfies

$$\begin{aligned} R_{\beta_-}(t(\boldsymbol{z}_{obs})) &= \frac{\alpha}{2} \quad \text{if } \min(t(\boldsymbol{z})) < t(\boldsymbol{z}_{obs}) \leq \max(t(\boldsymbol{z})) \\ \beta_- &= -\infty \quad \text{if } t(\boldsymbol{z}_{obs}) = \min(t(\boldsymbol{z})). \end{aligned}$$

Similarly, β_+ satisfies

$$L_{\beta_+}(t(\boldsymbol{z}_{obs})) = \frac{\alpha}{2} \quad \text{if } \min(t(\boldsymbol{z})) \leq t(\boldsymbol{z}_{obs}) < \max(t(\boldsymbol{z}))$$
$$\beta_+ = \infty \quad \text{if } t(\boldsymbol{z}_{obs}) = \max(t(\boldsymbol{z})).$$

For more information on obtaining these bounds, see Appendix A (especially Figures A.1 and A.2). When data sets are large, the normalized probabilities can be used in place of the unnormalized probabilities, and the test statistics can be re–scaled by replacing $\exp \beta t(\boldsymbol{z}_{obs})$ with $\exp \beta(t(\boldsymbol{z}_{obs}) - \min(t(\boldsymbol{z})))$ in Equation 2.10 (see Appendix B).

Under the framework of Section 2.3.2, the conditional probability is in the same form as Equation 2.10. The only differences are conditioning on the n_{ij} instead of the n_i and conditioning on two correlation parameters instead of one. The methodology of computing a point estimate and its confidence interval are analogous to the methods outlined in this section.

2.2.4 Network Algorithm

As was mentioned in Section 2.2.3, enumerating all tables in the reference set $\Gamma(s^\alpha(\boldsymbol{z}), s^\delta(\boldsymbol{z}))$ is difficult. Explicitly enumerating these tables is computationally infeasible for practical purposes, but by implicitly enumerating the tables, the process becomes feasible. Corcoran et al. (2001) propose a network approach, building on the network approach of Mehta et al. (1992), which allows $\Gamma(s^\alpha(\boldsymbol{z}), s^\delta(\boldsymbol{z}))$ to be implicitly enumerated. We begin by describing the graphical network algorithm approach used by Corcoran et al. (2001) under the original framework of Section 2.2.

We start by building the $\Gamma(s^\alpha)$ network by moving forward through the network and conditioning on only s^α. We then move backward and prune the $\Gamma(s^\alpha)$ network by

imposing the additional s^{δ} constraint to create the $\Gamma(s^{\alpha}, s^{\delta})$ network. Each path in the $\Gamma(s^{\alpha}, s^{\delta})$ network represents exactly one possible table, or meta–analysis data set, subject to the sufficient statistics constraints. After the $\Gamma(s^{\alpha}, s^{\delta})$ network is built, we again make a forward pass through the network gathering information on the distribution of the test statistic and the corresponding probabilities.

First, we need to determine the number of nodes that will be in the network. Each network is divided into $k+1$ stages, indexed over $0, \ldots, k$, where k represents the number of studies in the meta–analysis. At each stage m, there is a set of nodes. For $\Gamma(s^{\alpha})$, each node is indexed by two elements, denoted by $(m, s^{\alpha,m})$. The first element m represents the m^{th} cluster, or study, of correlated binary observations. The second component $s^{\alpha,m} = \sum_{i=1}^{m} y_i$ is one possible value of the partial sum of responses from the first m studies. For each network, there is a single initial node and a single terminal node. The initial node is $(0, 0)$, and the terminal node is (k, s^{α}). The set of successor nodes $\mathcal{R}(m-1, s^{\alpha,m-1})$ to a given node $(m-1, s^{\alpha,m-1})$ can be enumerated explicitly as

$$\mathcal{R}(m-1, s^{\alpha,m-1}) = \left\{(m, u) : \max\left(s^{\alpha,m-1}, s^{\alpha} - \sum_{l=m+1}^{k} n_l\right) \leq u \leq \min\left(s^{\alpha}, s^{\alpha,m-1} + n_m\right)\right\},$$

for $m = 1, \ldots, k$ and where u represents the values for partial sums for the successor nodes.

We can now make the forward pass to build the $\Gamma(s^{\alpha})$ network. Consider a node $(m, s^{\alpha,m}) \in \Gamma(s^{\alpha})$, for $m = 0, \ldots, k-1$. For each $(m+1, u) \in \mathcal{R}(m, s^{\alpha,m})$, define the length r_{2m} of the connecting arc as

$$r_{2m} = (s^{\alpha,m+1} - s^{\alpha,m})(n_{m+1} - s^{\alpha,m+1} + s^{\alpha,m}),$$

and calculate the arc length for each arc.

Once all arc lengths are computed, we proceed by calculating $SP_2(m, s^{\alpha,m})$ and $LP_2(m, s^{\alpha,m})$, which respectively represent the shortest and longest path lengths for s^{δ}

over all partial paths that originate at node $(0,0)$ and terminate at node $(m, s^{\alpha,m})$. These bounds can be calculated recursively. (Note: $SP_2(0,0)$ and $LP_2(0,0)$ are always 0.) Let the set $\mathcal{P}(m, s^{\alpha,m})$ contain the predecessor nodes to $(m, s^{\alpha,m})$, so that, for $m = 1, \ldots, k$,

$$\mathcal{P}(m, s^{\alpha,m}) = \{(m-1, u) : (m, s^{\alpha,m}) \in \mathcal{R}(m-1, u)\}.$$

Then, we can calculate $SP_2(m, s^{\alpha,m})$ and $LP_2(m, s^{\alpha,m})$ as follows

$$SP_2(m, s^{\alpha,m}) = \min_{\{(m-1,u)\in\mathcal{P}(m,s^{\alpha,m})\}} \{SP_2(m-1, u) + r_{2m}\}$$
$$LP_2(m, s^{\alpha,m}) = \max_{\{(m-1,u)\in\mathcal{P}(m,s^{\alpha,m})\}} \{LP_2(m-1, u) + r_{2m}\}.$$

We have now constructed the $\Gamma(s^\alpha)$ network and can proceed by creating the $\Gamma(s^\alpha, s^\delta)$ network. First, we create the terminal node (k, s^α, s^δ). We then proceed, stage by stage in reverse, beginning at stage k and ending at Stage 1. At the m^{th} stage,

1. Choose a node $(m, s^{\alpha,m}, s^{\delta,m})$.
2. For each node $(m-1, u) \in \mathcal{P}(m, s^{\alpha,m})$:
   (a) Create the triple $(m-1, u, v)$, where
   $v = s^{\delta,m} - r_{2m} = s^{\delta,m} - (s^{\alpha,m} - u)(n_m - s^{\alpha,m} + u)$.
   (b) If $SP_2(m-1, u) > v$ or $LP_2(m-1, u) < v$, then this node cannot be a member of the network $\Gamma(s^\alpha, s^\delta)$ and is dropped.
   (c) If the triple $(m-1, u, v)$ passes the shortest path and longest path tests of the previous step, then this node is feasible and is stored.

Once we know the node $(m-1, u, v)$ is a feasible predecessor of $(m, s^{\alpha,m}, s^{\delta,m})$, we need to compute the rank length $r_{1m} = x_m(s^{\alpha,m} - u)$ and the probability length c_{0m} of the arc connecting $(m-1, u, v)$ to $(m, s^{\alpha,m}, s^{\delta,m})$ as $c_{0m} = \binom{n_m}{s^{\alpha,m}-u}$.

Note that for any path that begins at the initial node $(0,0,0)$ and ends at the terminal node $(k, s^{\alpha}, s^{\delta})$, the value of the test statistic for that path can be obtained as $\sum_{l=1}^{k} r_{1l}$, and the unnormalized probability under H_0 of having observed the corresponding table is $\prod_{l=1}^{k} c_{0l}$. We can, therefore, define $SP_1(m-1,u,v)$ and $LP_1(m-1,u,v)$ respectively as the shortest and longest path lengths $\sum_{l=1}^{k} r_{1l}$ over all paths that begin at $(m-1,u,v)$ and end at the terminal node $(k, s^{\alpha}, s^{\delta})$. Likewise, we let $TP(m-1,u,v)$ represent the sum of all unnormalized probability paths $\prod_{l=1}^{k} c_{0l}$ over all such partial paths. This implies that $SP_1(k, s^{\alpha}, s^{\delta}) = LP_1(k, s^{\alpha}, s^{\delta}) = 0$, $TP(k, s^{\alpha}, s^{\delta}) = 1$, and $TP(0,0,0)$ is the normalizing constant.

When storing the node $(m-1,u,v)$, note that

- If $(m-1,u,v)$ already exists (i.e., was found previously to be the predecessor of another node at the m^{th} stage), then
  - $SP_1(m-1,u,v) = \min\{SP_1(m-1,u,v), r_{1m} + SP_1(m, s^{\alpha,m}, s^{\delta,m})\}$
  - $LP_1(m-1,u,v) = \max\{LP_1(m-1,u,v), r_{1m} + LP_1(m, s^{\alpha,m}, s^{\delta,m})\}$
  - $TP(m-1,u,v) = TP(m-1,u,v) + c_{om} TP(m, s^{\alpha,m}, s^{\delta,m})$
- If $(m-1,u,v)$ has not yet been stored, then
  - $SP_1(m-1,u,v) = r_{1m} + SP_1(m, s^{\alpha,m}, s^{\delta,m})$
  - $LP_1(m-1,u,v) = r_{1m} + LP_1(m, s^{\alpha,m}, s^{\delta,m})$
  - $TP(m-1,u,v) = c_{om} TP(m, s^{\alpha,m}, s^{\delta,m})$

Now that we have created the $\Gamma(s^{\alpha}, s^{\delta})$ network, we will make one final pass through the network. We proceed forward, stage by stage, beginning at the initial node $(0,0,0)$ and carry forward a set of records $\mathcal{Y}_m$ at each stage for $m = 0, \ldots, k-1$. Each record $d \in \mathcal{Y}_m$ is of the form $d = (m, s^{\alpha,m}, s^{\delta,m}, t_m, h_m)$, where $t_m = \sum_m r_{1m}$ is a possible partial rank length of a path terminating at node $(m, s^{\alpha,m}, s^{\delta,m})$, and h_m represents the unnormalized

sum of the probabilities of all partial paths with rank length r_{1m} that terminate at $(m, s^{\alpha,m}, s^{\delta,m})$.

Using shortest path/longest path logic with respect to the partial test statistic t_m, one can navigate through the network breadth first, eliminating the nodes through which no path contributes to the critical region. We begin with a single record $(0,0,0,0,1)$, associated with the initial node $(0,0,0)$, and proceed forward, stage by stage, for $m = 1, \ldots, k$. At the m^{th} stage

1. Choose a record $d = (m, s^{\alpha,m}, s^{\delta,m}, t_m, h_m) \in \mathcal{Y}_m$.
2. For each $g = (m+1, s^{\alpha,m+1}, s^{\delta,m+1}) \in \mathcal{R}(m, s^{\alpha,m}, s^{\delta,m})$, transfer d to $\mathcal{Y}_{m+1}$ as follows
   (a) If $t_m + r_{1,m+1} + LP_1(m+1, s^{\alpha,m+1}, s^{\delta,m+1}) < t_{obs}$ then do not transfer the record.
   (b) Else if there exists a $d' = (m+1, s^{\alpha,m+1}, s^{\delta,m+1}, t_{m+1}, h_{m+1}) \in \mathcal{Y}_{m+1}$ such that $t_{m+1} = t_m + r_{1,m+1}$, then merge d and d' by letting $d' = (m+1, s^{\alpha,m+1}, s^{\delta,m+1}, t_{m+1}, h'_{m+1})$ where $h'_{m+1} = h_{m+1} + h_m \binom{n_{m+1}}{s^{\alpha,m+1} - s^{\alpha,m}}$.
   (c) Else create a new record $d' \in \mathcal{Y}_{m+1}$ such that $d' = (m+1, s^{\alpha,m+1}, s^{\delta,m+1}, t_m + r_{1,m+1}, h_m \binom{n_{m+1}}{s^{\alpha,m+1} - s^{\alpha,m}})$.
3. Continue until all $d \in \mathcal{Y}_m$ are exhausted.

At stage k there remains a collection of records $\mathcal{Y}_k$ such that for each $d = (k, s^{\alpha}, s^{\delta}, t_k, h_k) \in \mathcal{Y}_k$ we have guaranteed that $t_k \geq t_{obs}$. The exact p–value is therefore given by

$$\Pr(T \geq t_{obs} | H_0, s) = \sum_{d \in \mathcal{Y}_k} \frac{h_k}{TP(0,0,0)}.$$

2.3 Innovation

We will now present our innovations to the methodology as applied to meta–analysis. We start by making two extensions to the exact test. The first extension is needed for the exact test to work with meta–analysis data. The second extension is needed to address non–negligible heterogeneity in a meta–analysis data set. We then show how these modifications are implemented in the network algorithm.

2.3.1 Exact Test: Variation of Treatment Level Within Study

A major drawback of the method described in Section 2.2 is that all participants in the i^{th} study must belong to the same treatment group, which is typically not the case for studies used in a meta-analysis. To allow for the variation of treatment group within study, we allow for extra conditioning with respect to the n_{ij}, as opposed to the study sizes $n_{(i+)}$ themselves. As such, the likelihood for this process remains the same as Equation 2.5. Table 2.2 illustrates a generic table for this framework, where

$$x_{ij} = \begin{cases} 0 & \text{when } j = 1 \\ 1 & \text{when } j = 2, \end{cases}$$

with $j = 1$ indicating the control group, and $j = 2$ indicating the treatment group.

With this modification, it would be feasible to apply this method to homogeneous meta–analysis data. However, it is arguably unwise to assume homogeneity, with the penalty for not accounting for heterogeneity being severe bias. It is safe to assume heterogeneity is present to some extent, and the following modification will attempt to account for this heterogeneity.

TABLE 2.2: Generic contingency table for a binary outcome with two treatments per study.

Study (i)	1		2		...	k		Sum
Treatment (x_{ij})	0	1	0	1	...	0	1	
z_{ij}	z_{11}	z_{12}	z_{21}	z_{22}	...	z_{k1}	z_{k2}	s^{α}
$n_{ij} - z_{ij}$	$n_{11} - z_{11}$	$n_{12} - z_{12}$	$n_{21} - z_{21}$	$n_{22} - z_{22}$	...	$n_{k1} - z_{k1}$	$n_{k2} - z_{k2}$	
n_{ij}	n_{11}	n_{12}	n_{21}	n_{22}	...	n_{k1}	n_{k2}	N
$x_{ij}z_{ij}$	$x_{11}z_{11}$	$x_{12}z_{12}$	$x_{21}z_{21}$	$x_{22}z_{22}$	...	$x_{k1}z_{k1}$	$x_{k2}z_{k2}$	t
$z_{ij}(n_{ij} - z_{ij})$	$z_{11}(n_{11} - z_{11})$	$z_{12}(n_{12} - z_{12})$	$z_{21}(n_{21} - z_{21})$	$z_{22}(n_{22} - z_{22})$	...	$z_{k1}(n_{k1} - z_{k1})$	$z_{k2}(n_{k2} - z_{k2})$	s^{δ}

2.3.2 Exact Test: Variation of Correlation Structure Among Treatment Levels

We now further extend the exact trend test to allow the correlation structure (δ in Equation 2.6) to vary among treatment groups. While this modification can be applied to any number of treatment groups, we will focus on only two treatment groups since that is common for meta-analysis data. Thus, we have two correlation parameters given by δ_j ($j = 1, 2$). As such, Equation 2.6 is now written as

$$
\begin{aligned}
\Pr(\boldsymbol{Z}=\boldsymbol{z}|\boldsymbol{x}) &= \prod_{i=1}^{k}\prod_{j=1}^{2}\binom{n_{ij}}{z_{ij}}\exp\{(\alpha+\beta x_{ij})z_{ij}-\delta_j z_{ij}(n_{ij}-z_{ij})+A_i(\alpha,\beta,\delta_1,\delta_2,\boldsymbol{n})\} \\
&= \left[\prod_{i=1}^{k}\prod_{j=1}^{2}\binom{n_{ij}}{z_{ij}}\right]\prod_{i=1}^{k}\prod_{j=1}^{2}\exp\Bigg\{(\alpha+\beta x_{ij})z_{ij}-\delta_j z_{ij}(n_{ij}-z_{ij}) \\
&\qquad\qquad +A_i(\alpha,\beta,\delta_1,\delta_2,\boldsymbol{n})\Bigg\} \\
&= \left[\prod_{i=1}^{k}\prod_{j=1}^{2}\binom{n_{ij}}{z_{ij}}\right]\exp\Bigg\{\sum_{i=1}^{k}\sum_{j=1}^{2}\big[(\alpha+\beta x_{ij})z_{ij}-\delta_j z_{ij}(n_{ij}-z_{ij}) \\
&\qquad\qquad +A_i(\alpha,\beta,\delta_1,\delta_2,\boldsymbol{n})\big]\Bigg\} \\
&= \left[\prod_{i=1}^{k}\prod_{j=1}^{2}\binom{n_{ij}}{z_{ij}}\right]\exp\Bigg\{\sum_{i=1}^{k}\sum_{j=1}^{2}(\alpha+\beta x_{ij})z_{ij}-\sum_{i=1}^{k}\sum_{j=1}^{2}\delta_j z_{ij}(n_{ij}-z_{ij}) \\
&\qquad\qquad +\sum_{i=1}^{k}A_i(\alpha,\beta,\delta_1,\delta_2,\boldsymbol{n})\Bigg\} \\
&= \left[\prod_{i=1}^{k}\prod_{j=1}^{2}\binom{n_{ij}}{z_{ij}}\right]\exp\Bigg\{\sum_{i=1}^{k}\sum_{j=1}^{2}(\alpha z_{ij}+\beta x_{ij}z_{ij})-\sum_{i=1}^{k}\sum_{j=1}^{2}\delta_j z_{ij}(n_{ij}-z_{ij}) \\
&\qquad\qquad +\sum_{i=1}^{k}A_i(\alpha,\beta,\delta_1,\delta_2,\boldsymbol{n})\Bigg\} \\
&= \left[\prod_{i=1}^{k}\prod_{j=1}^{2}\binom{n_{ij}}{z_{ij}}\right]\exp\Bigg\{\alpha\sum_{i=1}^{k}\sum_{j=1}^{2}z_{ij}+\beta\sum_{i=1}^{k}\sum_{j=1}^{2}x_{ij}z_{ij} \\
&\qquad\qquad -\left(\delta_1\sum_{i=1}^{k}z_{i1}(n_{i1}-z_{i1})+\delta_2\sum_{i=1}^{k}z_{i2}(n_{i2}-z_{i2})\right) \\
&\qquad\qquad +\sum_{i=1}^{k}A_i(\alpha,\beta,\delta_1,\delta_2,\boldsymbol{n})\Bigg\} \\
&= \left[\prod_{i=1}^{k}\prod_{j=1}^{2}\binom{n_{ij}}{z_{ij}}\right]\exp\Bigg\{\alpha s^{\alpha}(\boldsymbol{z})+\beta t(\boldsymbol{z})-\Big(\delta_1 s^{\delta_1}(\boldsymbol{z})+\delta_2 s^{\delta_2}(\boldsymbol{z})\Big) \\
&\qquad\qquad +\sum_{i=1}^{k}A_i(\alpha,\beta,\delta_1,\delta_2,\boldsymbol{n})\Bigg\},
\end{aligned}
$$

where the sufficient statistics are

$$\boldsymbol{s} = \begin{cases} s^{\alpha}(\boldsymbol{z}) = \sum_{i=1}^{k} \sum_{j=1}^{2} z_{ij} & \text{for } \alpha \\ t(\boldsymbol{z}) = \sum_{i=1}^{k} \sum_{j=1}^{2} x_{ij} z_{ij} & \text{for } \beta \\ s^{\delta_1}(\boldsymbol{z}) = \sum_{i=1}^{k} z_{i1}(n_{i1} - z_{i1}) & \text{for } \delta_1 \\ s^{\delta_2}(\boldsymbol{z}) = \sum_{i=1}^{k} z_{i2}(n_{i2} - z_{i2}) & \text{for } \delta_2. \end{cases}$$

As before, we continue as outlined in Section 2.2 with the addition of conditioning on the n_{ij} from Section 2.3.1 and the new adjustment of conditioning on $s^{\delta_1}(\boldsymbol{z})$ and $s^{\delta_2}(\boldsymbol{z})$, as opposed to conditioning on just $s^{\delta}(\boldsymbol{z})$. Another adjustment to the network algorithm can be used to make this method computationally efficient, and this adjustment will be described in Section 2.3.3. C code for this method is provided in Appendix C. Table 2.3 illustrates a generic table for this framework.

2.3.3 Network Algorithm: Variation of Treatment Level Within Study

The network algorithm described in Section 2.2.4 can be modified to allow for treatment levels to vary within studies. As was mentioned in Section 2.3.1, the only difference between this test and the basic test described in 2.2 involves extra conditioning on the number of patients within each treatment group in each study n_{ij}. Note that this extra conditioning has no effect on the sufficient statistics s^{α} and s^{δ}. So, we can adjust the network algorithm to handle this extra conditioning by viewing the unit of analysis as treatment and study, as opposed to just the study. Working under the framework of Section 2.3.1, we adjust the network to consist of $2k + 1$ stages instead of $k + 1$ stages. We can then proceed with the same logic outlined in Section 2.2.4.

TABLE 2.3: Generic contingency table for a binary outcome with two treatments and two correlation structures per study.

Study (i)	1		2		...	k		Sum
Treatment (x_{ij})	0	1	0	1	...	0	1	
z_{ij}	z_{11}	z_{12}	z_{21}	z_{22}	...	z_{k1}	z_{k2}	s^{α}
$n_{ij} - z_{ij}$	$n_{11} - z_{11}$	$n_{12} - z_{12}$	$n_{21} - z_{21}$	$n_{22} - z_{22}$	...	$n_{k1} - z_{k1}$	$n_{k2} - z_{k2}$	
n_{ij}	n_{11}	n_{12}	n_{21}	n_{22}	...	n_{k1}	n_{k2}	N
$x_{ij}z_{ij}$	$x_{11}z_{11}$	$x_{12}z_{12}$	$x_{21}z_{21}$	$x_{22}z_{22}$	...	$x_{k1}z_{k1}$	$x_{k2}z_{k2}$	t
$z_{i1}(n_{i1} - z_{i1})$	$z_{11}(n_{11} - z_{11})$		$z_{21}(n_{21} - z_{21})$		...	$z_{k1}(n_{k1} - z_{k1})$		s^{δ_1}
$z_{i2}(n_{i2} - z_{i2})$		$z_{12}(n_{12} - z_{12})$		$z_{22}(n_{22} - z_{22})$	...		$z_{k2}(n_{k2} - z_{k2})$	s^{δ_2}

2.3.4 Network Algorithm: Variation of Correlation Structure Among Treatment Levels

We can now modify the network algorithm to let the correlation structure vary among treatment levels. As in Section 2.3.3, we need to condition on the n_{ij}. Additionally, we need to condition on the extra correlation parameters as described in Section 2.3.2.

Accordingly, we will have $2k+1$ stages in the network to allow for the treatment level varying within study. We proceed by following the steps outlined in Section 2.2.4 until we reach the backward pass where the s^{δ} information is needed. We will then arrange the data set by treatment level so that all information contributing to s^{δ_1} is listed first, and all information contributing to s^{δ_2} is listed second.

To create the $\Gamma(s^{\alpha}, s^{\delta_1}, s^{\delta_2})$ network, we first create the terminal node $(N, s^{\alpha}, s^{\delta_1}, s^{\delta_2})$, where $N = 2k$. Now, we proceed by working with the information contributing to s^{δ_2}, stage by stage in reverse, beginning at stage N and ending at Stage $T+1$, where T is the number of studies contributing information to s^{δ_1}. At the m^{th} stage,

1. Choose a node $(m, s^{\alpha,m}, s^{\delta_1,m}, s^{\delta_2,m})$.
2. For each node $(m-1, u) \in \mathcal{P}(m, s^{\alpha,m})$:
   (a) Create the quadruple $(m-1, u, s^{\delta_1}, v_1)$, where
   $v_1 = s^{\delta_2,m} - r_{2m} = s^{\delta_2,m} - (s^{\alpha,m} - u)(n_m - s^{\alpha,m} + u)$
   (b) If $m = T+1$ and $v_1 \neq 0$, then this node cannot be a member of the network $\Gamma(s^{\alpha}, s^{\delta_1}, s^{\delta_2})$ and is dropped.
   (c) For any value of m other than $T+1$, if $v_1 > s^{\delta_2}$ or $v_1 < 0$, then this node cannot be a member of the network $\Gamma(s^{\alpha}, s^{\delta_1}, s^{\delta_2})$ and is dropped.
   (d) If the quadruple $(k-1, u, s^{\delta_1}, v_1)$ passes the tests of the previous steps, then this node is feasible and is stored.

When this process is finished, the nodes that remain at stage T are of the form $(T, s^{\alpha,T}, s^{\delta_1}, 0)$ and are the terminal nodes for the next phase involving the information contributing to s^{δ_1}. We continue by working with the information related to s^{δ_1}, stage by stage in reverse, beginning at stage T and ending at Stage 1. At the m^{th} stage,

1. Choose a node $(m, s^{\alpha,m}, s^{\delta_1}, 0)$.
2. For each node $(m-1, u) \in \mathcal{P}(m, s^{\alpha,m})$:
   (a) Create the quadruple $(m-1, u, v_0, 0)$, where
   $v_0 = s^{\delta_1,m} - r_{2m} = s^{\delta_1,m} - (s^{\alpha,m} - u)(n_m + s^{\alpha,m} - u)$
   (b) If $m = 1$ and $v_0 \neq 0$, then this node cannot be a member of the network $\Gamma(s^{\alpha}, s^{\delta_1}, s^{\delta_2})$ and is dropped.
   (c) For any value of m other than 1, if $v_0 > s^{\delta_1}$ or $v_0 < 0$, then this node cannot be a member of the network $\Gamma(s^{\alpha}, s^{\delta_1}, s^{\delta_2})$ and is dropped.
   (d) If the quadruple $(m-1, u, v_0, 0)$ passes the tests of the previous steps, then this node is feasible and is stored.

We then continue as outlined in Section 2.2.4 starting with computing the rank length and probability length, and ending by calculating the exact p–value.

2.4 Application

We begin this section by comparing our new exact permutation–based method to several existing methods via a simulation study of rare event meta–analysis data. We then apply our method to two real meta–analysis data sets.

2.4.1 Simulation Study

We compare our new exact permutation–based method (Exact–Perm) with the following methods: inverse variance (INV), Mantel-Haenszel (MH), Peto, DerSimonian and Laird (DSL), and the exact p–value function combination approach of Liu et al. (2014) (Exact–pVal). These methods were chosen as they are readily available for use in software. We compare type I error rates, bias, and confidence interval coverage of our method to current methods under the following simulation study, similar to the simulation study in Bhaumik et al. (2012).

Consider the random–effects model

$$Y_{ij} \sim \text{Bernoulli}(\pi_{ij}),$$

such that

$$\text{logit}(\pi_{ij}) = \mu_i + X_{ij}\theta_i, \qquad \mu_i \sim N(\mu, \sigma_i^2), \qquad \theta_i \sim N(\theta, \tau^2),$$

and

$$\pi_{ij} = \frac{\exp\{\mu_i + X_{ij}\theta_i\}}{1 + \exp\{\mu_i + X_{ij}\theta_i\}},$$

where μ is the underlying event rate for the i^{th} study; X_{ij} is an indicator variable for treatment group; θ represents the true treatment effect (on the log odds ratio scale); and τ^2, the heterogeneity parameter, determines to what extent the treatment effects vary across studies.

We generate data under this framework. We use the nominal significance level of $\alpha = 0.05$, and we let $\theta = 0$, 0.5, 1, and 1.5 (corresponding odds ratios of 1, 1.6, 2.7, and 4.5); $\tau^2 = 0$, 0.2, 0.4 and 0.8; $\mu = -4$ and -3, with corresponding control group average

event rates of 1.8% and 4.7%; $\sigma^2 = 0.5$; and $k = 10$. Based on a rough scan of meta–analysis data sets, the combination of the values used for the between–study variance (τ^2) and the within–study variance (σ_i^2) seem to be realistic. For the i^{th} study, the sample sizes in the treatment and control group were independently generated from rounding a uniform distribution with a minimum sample size of 10 and a maximum sample size of 50.

We simulate 10,000 replications for each combination of μ and τ^2 to ensure precise estimations of statistical performance. Data sets are not included if they have zero events across all studies for either the treatment group or the control group. We also exclude data sets that generate test statistic distributions containing only one value of the test statistic. Additionally, we exclude data sets where the observed test statistic is at either extreme of the distribution. We exclude these data sets until we reach 10,000 non–trivial simulated data sets. Additionally, we chose to work with rare to very rare events due to the unrealistic computation time our exact permutation–based method requires when the number of events is large. While it would be feasible to apply our method to *one* meta–analysis with large underlying event rates, it was not feasible to use our method for *thousands* of simulated data sets with large event rates in a reasonable amount of time.

For our exact permutation–based method, we primarily used C code for the simulations, in addition to some R software (R Core Team, 2014) code (see Appendix C). Due to the computational power needed for the simulations of our method, we used the Center for High Performance Computing (CHPC) at the University of Utah. We utilized the Ember cluster, which consists of 18 nodes, each with 32 cores and 256 GB of RAM, that are designated for Utah State University researchers. Enumerating the reference set for our method requires substantial computer memory, and some of our simulations exceeded the 256 GB of memory provided by the 18 public nodes. Thankfully, two Utah State University professors who owned private nodes were kind enough to let us use their nodes, when needed. The support and resources from the CHPC at the University of Utah are gratefully acknowledged.

For all other methods, we used the R software for the simulations. The *metabin* function from the *meta* R package was used to obtain results from the four traditional methods. For the inverse variance and DerSimonian and Laird methods, we applied a 0.5 continuity correction to studies with zero events and included studies with zero events in both arms (*method = "inverse", incr = 0.5, allstudies = TRUE, method.tau = "DL"*). We applied the same adjustments for the Mantel–Haenszel method (*method = "MH", incr = 0.5, allstudies = TRUE*). The Peto method had no continuity correction applied (*method = "Peto", incr = 0*). Finally, the *gmeta* function from the *gmeta* R package was used for the exact *p*–value function combination method (*method = "exact1"*).

2.4.1.1 *Type I Error*

Figure 2.1 shows the simulated type I error rates for each method for different values of μ (baseline event rate) and τ^2 and when $\theta = 0$. For larger values of τ^2, the type I error rate becomes markedly larger than the nominal rate for the INV, MH, Peto, DSL and Exact–pVal methods. While the type I error rates for all methods hover around the nominal level when there is no heterogeneity, the presence of heterogeneity causes these comparison methods to fail greatly. It is interesting to note that the INV and DSL methods produce very similar type I error rates, even though the DSL method is attempting to estimate τ^2, which would suggest the DSL would perform better than the INV method. Their similar type I error rates are not surprising, though, in this case when the number of studies to be pooled is relatively small, causing the DSL estimator of τ^2 to be under-powered (Huedo-Medina et al., 2006). The DSL estimator of τ^2 is likely estimating the heterogeneity to be 0, even when there really is substantial heterogeneity in the data, thus producing results identical to the INV method.

The type I error rate, interestingly, also increases as the underlying event rate increases from approximately 1.8% to 4.7% for all methods. While this trend may seem counterintuitive, it is likely related to an issue of power. When $\tau^2 = 0.8$ and $\mu = -3$

(baseline event rate of 4.7%), the best of these comparison methods yields a type I error rate of about 30%! It is very clear that these methods do not produce reliable results when events are rare, and heterogeneity is present. Our new exact permutation-based approach (Exact-Perm) does markedly better than the comparison methods. The type I error rate hovers around the nominal rate, even when heterogeneity is present, and events are rare.

Overall, when there is heterogeneity between studies, the INV, MH, Peto, DSL, and Exact–pVal methods have highly inflated type I error rates, regardless of the underlying event rate, while the type I error rate of the Exact–Perm method stays closer to the nominal level. We believe this marked improvement in performance boils down to treating the correlation among the treatment and control groups separately. None of the methods, except for the DSL method, attempts to account for heterogeneity, and the DSL method suffers from insufficient power to detect heterogeneity. Additionally, as asymptotic methods are known to be unreliable when events are rare, it is no surprise that an exact approach outperforms the INV, MH, Peto, and DSL methods. The exact p–value function combination method, Exact–pVal, interestingly, did not outperform these asymptotic methods, which may be due to this method not incorporating a measure of heterogeneity. Our exact approach outperforms all these methods when events are rare, and heterogeneity is present among studies.

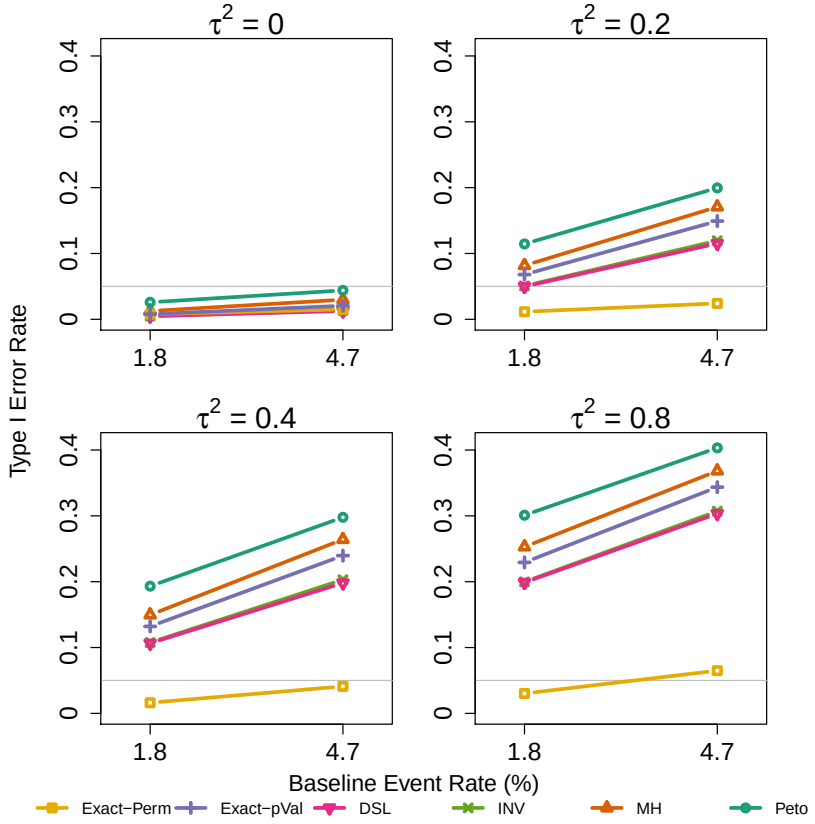

FIGURE 2.1: This figure compares the type I error rates of several existing methods to our new exact method with varied values of μ (baseline event rate) and τ^2. The horizontal, grey line marks the nominal level of 0.05.

2.4.1.2 *Bias*

Figure 2.2 compares the estimated log odds ratios with the true log odds ratios for each method and for different values of μ (baseline event rate), τ^2, and θ (log odds ratio). When the average baseline event rate is 4.7%, all methods perform similarly, including our exact method. There is some bias in all methods, especially as the true log odds ratio increases, which results in underestimation of the treatment effect. When the average baseline event rate decreases to 1.8%, all methods become more biased.

Our exact method fares about the same as the commonly used methods, excluding the Exact–pVal method, which has less bias overall than the other methods. Interestingly,

the presence of heterogeneity does not seem to affect the estimate of the log odds ratio as much as we thought it may, although when heterogeneity increases, all methods overestimate the treatment effect. The Exact–pVal method is less biased than the rest of the methods, and this may be due to heterogeneity apparently not having a large impact on estimation.

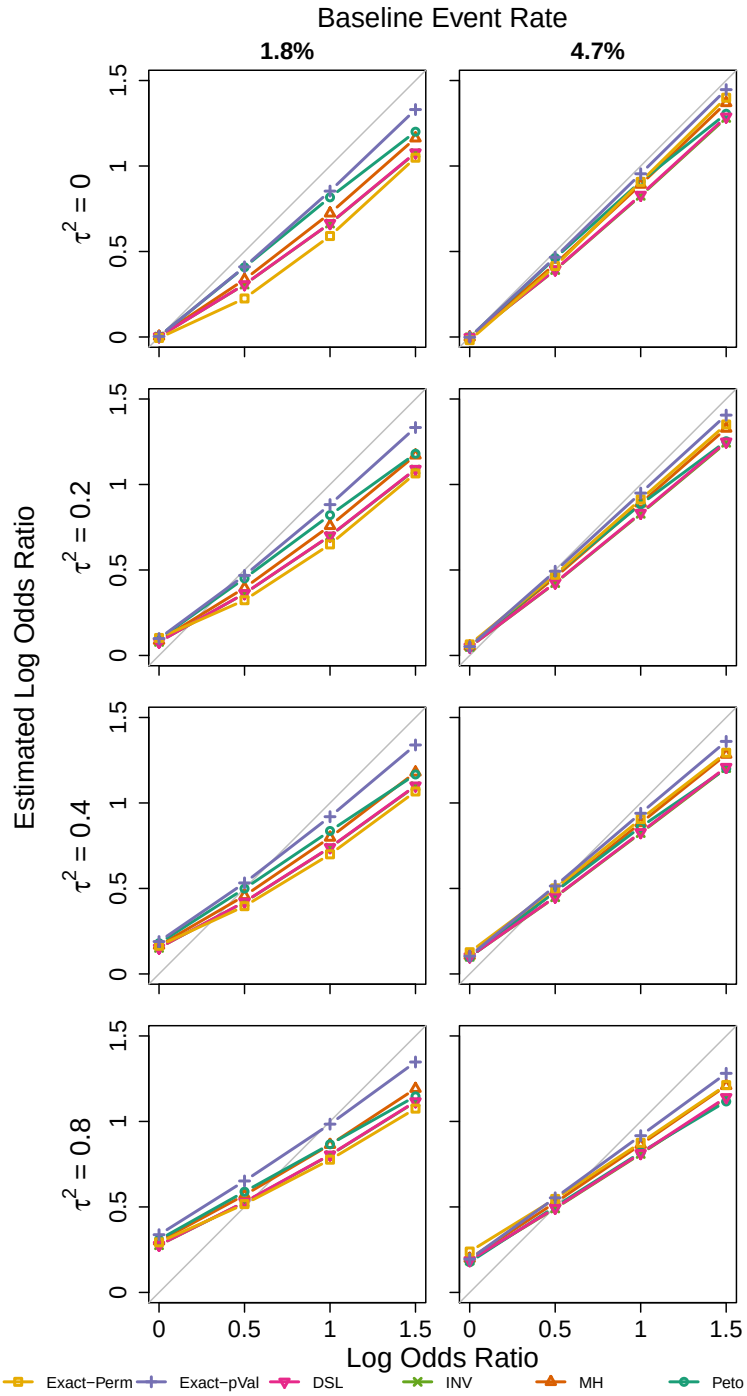

FIGURE 2.2: This figure compares bias of several existing methods to our new exact method with varied values of μ (baseline event rate), τ^2, and θ (log odds ratio). The diagonal, grey line indicates no bias, with the estimated log odds ratio equaling the true log odds ratio.

2.4.1.3 *Confidence Interval Coverage*

Figure 2.3 compares the confidence interval coverage for each method and for different values of μ (baseline event rate), τ^2, and θ (log odds ratio). As heterogeneity increases, the commonly used methods have very low coverage – as low as 50% with substantial heterogeneity and a baseline event rate of 4.7%. The Peto and Exact–pval methods are fairly consistent across all values of the true log odds ratio, in contrast to the other commonly used methods which result in a decrease in coverage probability as the true log odds ratio increases. Similarly to Figure 2.1, as the baseline event rate increases from 1.8% to 4.7%, the commonly used methods and the Exact–pVal method generally produce lower coverage. Interestingly, the commonly used methods perform extra poorly when the true log odds ratio is 1.5, and the average baseline event rate is 1.8%.

In contrast, our exact method never falls below a 95% coverage. The discrete nature of the data causes our method to be rather conservative because the method does not guarantee a 95% coverage, but rather, it guarantees at least a 95% coverage. Thus, the confidence intervals from our methods are generally much wider than the confidence intervals produced from the commonly used methods. However, we believe a wide confidence interval is preferred over the alternative of confidence intervals which do not contain the true log odds ratio.

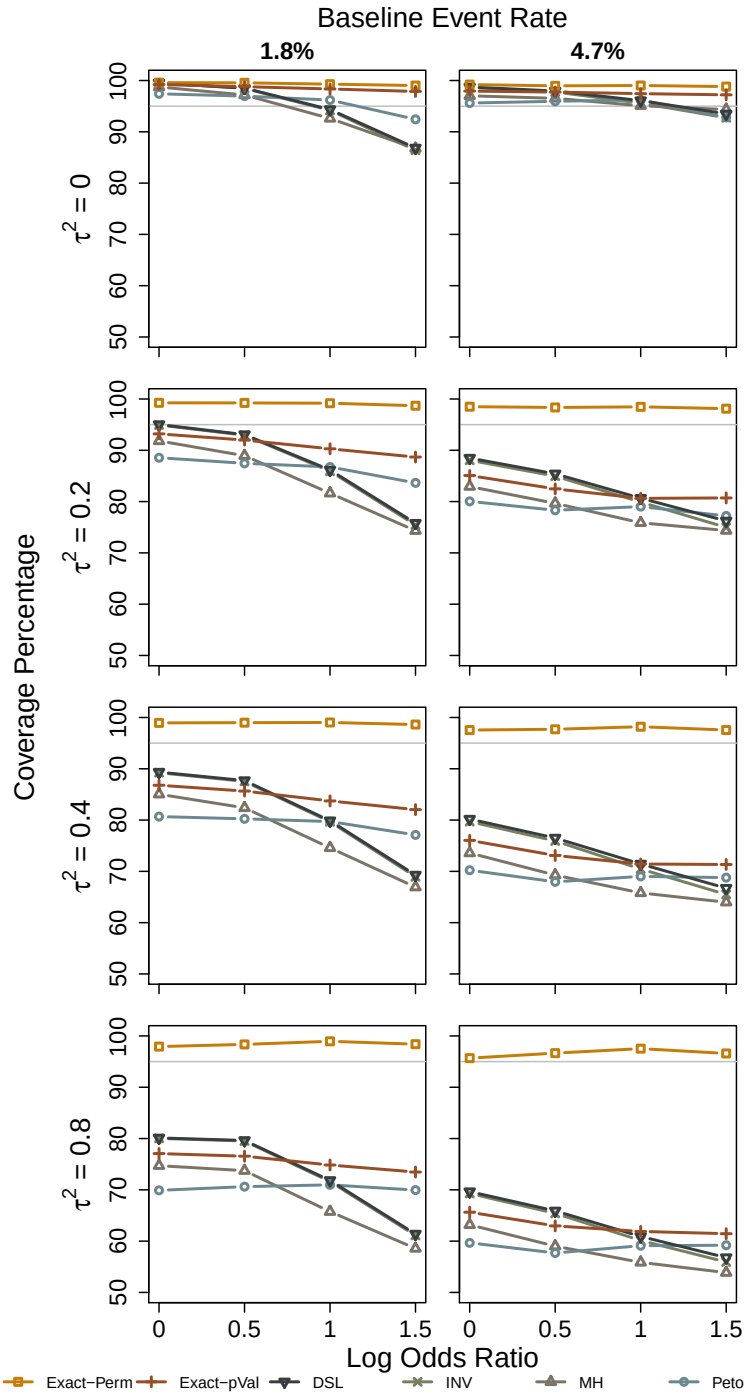

FIGURE 2.3: This figure compares the confidence interval coverage of several existing methods to our new exact method with varied values of μ (baseline event rate), τ^2, and θ (log odds ratio). The horizontal, grey line marks the nominal 95% coverage value.

2.4.2 Illustrative Examples

We now apply the methods described in Section 2.3 to obtain p–values, point estimates, and 95% confidence intervals for two real data sets. The first example illustrates the statistical power of our method to detect likely underlying treatment effects when other methods also detect an effect. The second example is one in which our method disagrees with other methods in concluding there is not a significant treatment effect. This is an indication that results from the other methods may be results of type I errors – especially in light of the simulation study in Section 2.4.

2.4.2.1 *Stomach Ulcers*

As our first example, we will use the stomach ulcer data set provided and used by Efron (1996) and used in Xie et al. (2011), shown in Table 2.4. A new surgical treatment for stomach ulcers is compared with an older surgical treatment in 41 independent randomized trials conducted between 1980 and 1989. The adverse event of recurrent bleeding is the outcome of interest. The average event rate with the new surgery is 0.21, and the average event rate with the old surgery is 0.46. Even though these average event rates are quite large, nine studies (studies 5, 6, 11, 25, 28, 29, 34, 40 and 41) have zero events in one or both treatment arms.

The DerSimonian and Laird estimate of τ^2 ranges from 0.45 when no continuity correction is used (thereby excluding the nine zero event studies) to 0.98 when a 0.5 continuity correction is used for studies with zero events in one treatment arm (including all studies except study 41, which has zero events in both treatment arms). Overall, this data set has non–negligible heterogeneity, some studies with low event rates and zero cells, a relatively large amount of studies, and a relatively large overall average event rate.

TABLE 2.4: Stomach ulcer data set. Source: Efron (1996)

Study	New Surgery		Old Surgery	
	Event	Non–Event	Event	Non–Event
1	7	8	11	2
2	8	11	8	8
3	5	29	4	35
4	7	29	4	27
5	3	9	0	12
6	4	3	4	0
7	4	13	13	11
8	1	15	13	3
9	3	11	7	15
10	2	36	12	20
11	6	6	8	0
12	2	5	7	2
13	9	12	7	17
14	7	14	5	20
15	3	22	11	21
16	4	7	6	4
17	2	8	8	2
18	1	30	4	23
19	4	24	15	16
20	7	36	16	27
21	6	34	13	8
22	4	14	5	34
23	14	54	13	61
24	6	15	8	13
25	0	6	6	0
26	1	9	5	10
27	5	12	5	10
28	0	10	12	2
29	0	22	8	16
30	2	16	10	11
31	1	14	7	6
32	8	16	15	12
33	6	6	7	2
34	0	20	5	18
35	4	13	2	14
36	10	30	12	8
37	3	13	2	14
38	4	30	5	14
39	7	31	15	22
40	0	34	34	0
41	0	9	0	16

For the stomach ulcer data, we compare results from our new exact permutation–based approach (Exact–Perm) with the same methods used in the simulation study: inverse variance (INV), DerSimonian and Laird (DSL), Mantel–Haenszel (MH), Peto, and the exact p–value function combination method (Exact–pVal). We used the R software and the same settings as used in the simulation study. This resulted in the Peto method excluding study 41, which has zero events in both treatment arms, since no continuity correction was used. Also note that the results from our exact method were produced using the median unbiased estimate. The results from these six methods are shown in Table 2.5.

TABLE 2.5: Meta–analysis results for the stomach ulcer data set. Note: the p–values are all two–sided.

	$\widehat{OR}$	95% CI	p–value
INV	0.41	$(0.32, 0.53)$	<0.0001 (significant)
MH	0.34	$(0.28, 0.42)$	<0.0001 (significant)
Peto	0.32	$(0.26, 0.40)$	<0.0001 (significant)
DSL	0.33	$(0.22, 0.50)$	<0.001 (significant)
Exact–pVal	0.29	$(0.23, 0.37)$	- (significant)
Exact–Perm	0.83	$(0.80, 0.86)$	<0.0001 (significant)

From Table 2.5, we can see that all methods, including our new exact permutation–based approach, yield the same substantive results. There is evidence that

the new surgical treatment significantly reduces the risk of recurrent bleeding. It appears the traditional methods and the p–value function combination method are not hampered by the rather substantial heterogeneity in this data set. This is likely due to the number of studies and average event rates both being large. Even with some studies having zero events and low event rates, these other methods seem to be resilient to their effects and do not seem to be results of a type I error. Our exact method agrees with these methods and has enough statistical power to detect the likely underling treatment effect, with the exact method even estimating a larger treatment effect than the other methods.

2.4.2.2 *Antibiotics*

We use the antibiotics data set provided by Spinks et al. (2013) and used in Friedrich et al. (2007) as our second example. The data are shown in Table 2.6. Sixteen studies, conducted from 1950 to 2000, compare the use of antibiotics to prevent acute rheumatic fever, a sore throat resulting from inadequately treated strep throat or scarlet fever, compared to a placebo. The event of interest is the occurrence of acute rheumatic fever. The average event rate for the antibiotics group is 0.0042, and the average event rate for the placebo group is 0.0111. These average event rates are relatively small, classifying rheumatic fever as a rare event. Eleven of the sixteen studies (69%) have zero events in one or both treatment arms (studies 1 through 9 having zero events in both arms, and studies 10 and 13 having zero events in one arm).

The DerSimonian and Laird estimate of τ^2 varies greatly, depending on the included studies. With no continuity correction, thus only including studies 11, 12, 14, 15 and 16, τ^2 is estimated to be 0.39. When using a 0.5 continuity correction to include studies 10 and 13, which have zero events in only one treatment arm, the estimate of τ^2 jumps to 0.43. When all studies are included, using a 0.5 continuity correction, τ^2 is estimated to be zero. Accordingly, we consider this data set to have non–negligible heterogeneity, despite the estimate of no heterogeneity when all studies are included in the analysis.

Continuity corrections are known to be less than ideal, and the DerSimonian and Laird heterogeneity estimator is known to be under–powered, so we feel comfortable assuming there is at least some non–negligible heterogeneity. As a summary, this data set has at least some heterogeneity, a rare event, a relatively small amount of studies, and several studies with zero cells.

TABLE 2.6: Antibiotics data set. Source: Spinks et al. (2013)

Study	Antibiotics		Placebo	
	Event	Non–Event	Event	Non–Event
1	0	121	0	118
2	0	59	0	58
3	0	358	0	164
4	0	454	0	216
5	0	238	0	268
6	0	62	0	59
7	0	186	0	97
8	0	87	0	94
9	0	369	0	386
10	0	257	2	109
11	2	798	17	804
12	5	978	35	996
13	0	605	2	608
14	2	277	5	198
15	2	157	1	50
16	26	650	12	220

For the antibiotics data, we compare results from our new exact permutation–based approach (Exact–Perm) again with the four traditional methods and the exact p–value function combination method (Exact–pVal), as used in the stomach ulcer analysis. The results are shown in Table 2.7.

TABLE 2.7: Meta–analysis results for the antibiotics data set. Note: the p–values are all two–sided.

	$\widehat{OR}$	95% CI	p–value
INV	0.37	$(0.23, 0.57)$	<0.0001 (significant)
MH	0.31	$(0.20, 0.46)$	<0.0001 (significant)
Peto	0.30	$(0.20, 0.45)$	<0.0001 (significant)
DSL	0.37	$(0.23, 0.57)$	<0.001 (significant)
Exact–pVal	0.30	$(0.19, 0.46)$	- (significant)
Exact–Perm	0.57	$(0.17, 1.76)$	0.3136 (non–significant)

Table 2.7 shows that the traditional and p–value function combination methods all indicate a significant effect of antibiotics in decreasing the occurrence of acute rheumatic fever. Our new exact permutation–based approach, however, suggests the observed treatment effect is not significant. This is due to the exact confidence interval covering the null value and being much wider than the confidence intervals of the other methods. Efthimiou (2018) states, "One additional issue with rare events is that, for the case of random effects meta–analysis, the estimation of the variance of random effects (heterogeneity) may be biased, which may lead to spuriously narrow confidence intervals." Given this, and our simulation study, it is very possible the results from the traditional methods and the p–value function combination method are results of a type I error. It is likely that our exact approach reflects the truth about the non–significance of the treatment effect.

2.4.3 Network Algorithm

We will now provide small example data sets to illustrate how the network algorithm works. We will first use the network algorithm for the original exact trend test for correlated data as outlined in Section 2.2.4. Next, we will show how the network algorithm works under the modifications of Sections 2.3.3 and 2.3.4.

2.4.3.1 *Original Exact Test for Correlated Data*

For this example, we will use the hypothetical data set shown in Table 2.8. Note that each study consists of one treatment, and there is one correlation structure for the entire data set.

TABLE 2.8: Example data set used to illustrate the original network algorithm as outlined in Section 2.2.4.

Study	1	2	3	4	Sum
Treatment	0	0	1	1	
z_i	2	1	1	2	6
$n_i - z_i$	1	2	4	4	11
n_i	3	3	5	6	17
$x_i z_i$	0	0	1	2	3
$z_i(n_i - z_i)$	2	2	4	8	16

The sufficient statistics for the data set shown in Table 2.8 are

$$\boldsymbol{s} = \begin{cases} s^{\alpha} = \sum_{i=1}^{k} z_i = 6 & \text{for } \alpha \\ t = \sum_{i=1}^{k} x_i z_i = 3 & \text{for } \beta \\ s^{\delta} = \sum_{i=1}^{k} z_i(n_i - z_i) = 16 & \text{for } \delta. \end{cases}$$

We now build the $\Gamma(s^{\alpha})$ network by moving forward through the network and conditioning on s^{α}. To start, there will be $k+1 = 4+1 = 5$ stages in the network, since there are 4 studies, starting at Stage 0 and ending at Stage 4. We begin building our network with the initial node $(0,0)$ and the terminal node $(4,6)$ since there are 4 studies and $s^{\alpha} = 6$. We can now determine how many nodes are at each stage. For Stage 1,

$$\begin{aligned} \mathcal{R}(m-1, s^{\alpha,m-1}) &= \Big\{(m,u) : \max\big(s^{\alpha,m-1}, s^{\alpha} - \sum_{l=m+1}^{k} n_l\big) \le u \le \min(s^{\alpha}, s^{\alpha,m-1} + n_m)\Big\} \\ \mathcal{R}(0,0) &= \Big\{(1,u) : \max\big(0, 6 - \sum_{l=2}^{4} n_l\big) \le u \le \min(6, 0+n_1)\Big\} \\ &= \Big\{(1,u) : \max\big(0, 6-(3+5+6)\big) \le u \le \min(4, 0+3)\Big\} \\ &= \Big\{(1,u) : 0 \le u \le 3\Big\}, \end{aligned}$$

so there are 4 nodes in Stage 1 (see Figure 2.4).

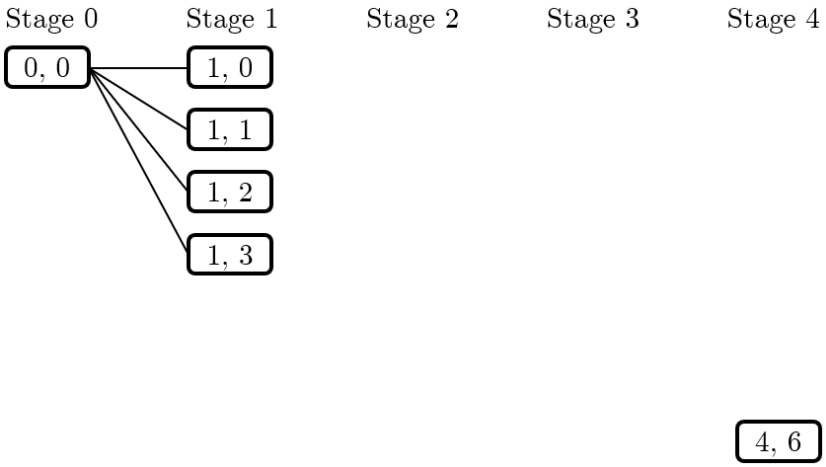

FIGURE 2.4: Building the $\Gamma(s^\alpha)$ network: number of nodes in Stage 1.

For the first node in Stage 2,

$$\mathcal{R}(m-1, s^{\alpha,m-1}) = \Big\{(m,u) : \max\big(s^{\alpha,m-1}, s^\alpha - \sum_{l=m+1}^{k} n_l\big) \le u \le \min(s^\alpha, s^{\alpha,m-1} + n_m)\Big\}$$

$$\begin{aligned}
\mathcal{R}(1,0) &= \Big\{(2,u) : \max\big(0, 6 - \sum_{l=3}^{4} n_l\big) \le u \le \min(6, 0+n_2)\Big\} \\
&= \Big\{(2,u) : \max\big(0, 6-(5+6)\big) \le u \le \min(6, 0+3)\Big\} \\
&= \Big\{(2,u) : 0 \le u \le 3\Big\},
\end{aligned}$$

so there are 4 nodes in Stage 2 that are connected to the first node in Stage 1 (see Figure 2.5).

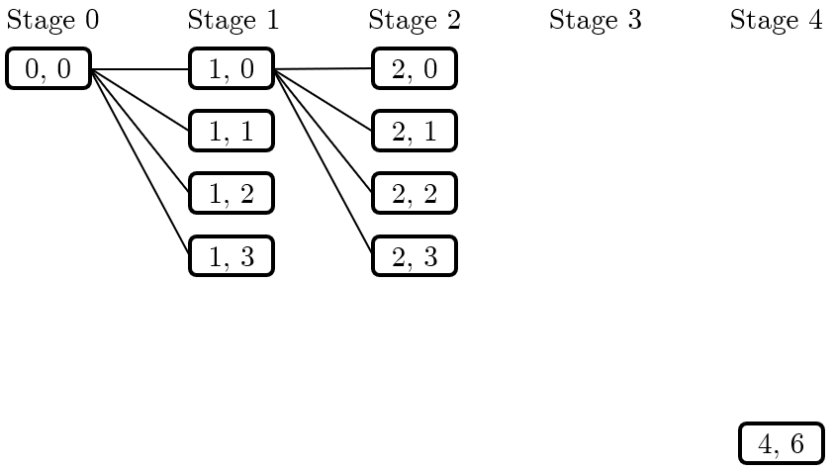

FIGURE 2.5: Building the $\Gamma(s^{\alpha})$ network: nodes in Stage 2 connected to node $(1, 0)$.

We repeat this process until we have the network shown in Figure 2.6. The data set shown in Table 2.8 is represented by the nodes and edges highlighted in red. There are 63 unique paths which represent 63 unique data sets that fit the constraints of having the same row and column totals and the total number of events equal to 6.

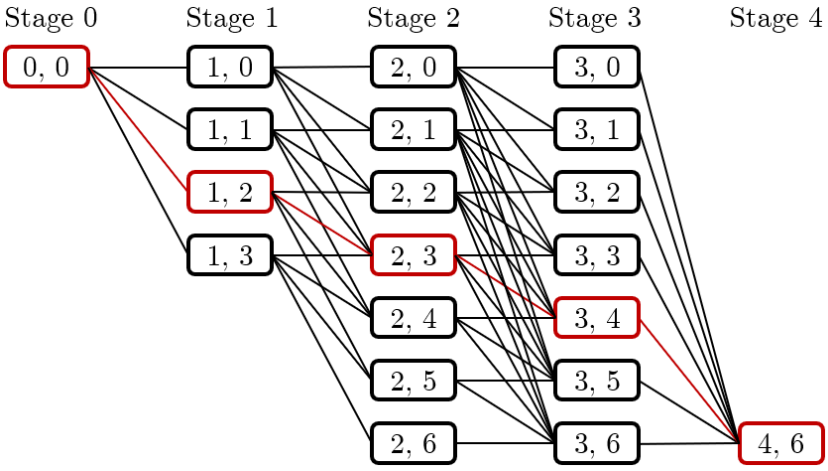

FIGURE 2.6: The $\Gamma(s^{\alpha})$ network with the nodes and edges representing the data in Table 2.8 in red.

We now need to build the $\Gamma(s^\alpha, s^\delta)$ network by moving backward and pruning the network according to the additional s^δ constraint. We start by creating the terminal node $(4, 6, 16)$ and the initial node $(0, 0, 0)$. We then proceed, stage by stage in reverse, beginning at Stage 4 and ending at Stage 1. For each predecessor node of the terminal node, we create new triple nodes.

For node $(3, 0)$, which is a predecessor node of $(4, 6)$,

$$\begin{aligned}\nu &= s^{\delta,m} - (s^{\alpha,m} - u)(n_m - s^{\alpha,m} + u) \\ &= 16 - (6 - 0)(6 - 6 + 0) \\ &= 16,\end{aligned}$$

so, we create the triple node $(3, 0, 16)$. We continue this process until we find all feasible predecessors of $(4, 6, 16)$. We do this for each stage until finishing Stage 1. In this example, there are many paths that do not ultimately connect to the initial node $(0, 0, 0)$ (like the $(3, 0, 16)$ node we just created). We finish this network by removing all paths that do not originate at $(0, 0, 0)$. The final $\Gamma(s^\alpha, s^\delta)$ network is shown in Figure 2.7. Again, the red nodes and edges represent the path of the data in Table 2.8.

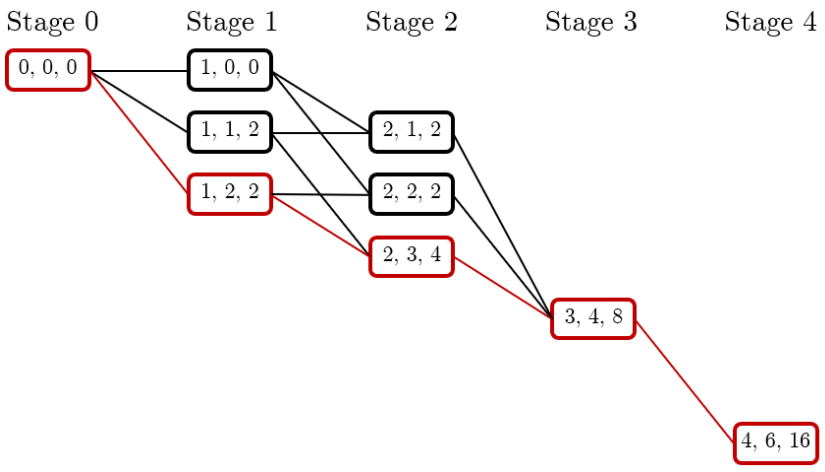

FIGURE 2.7: The $\Gamma(s^\alpha, s^\delta)$ network with the nodes and edges representing the data in Table 2.8 in red.

Each path in the $\Gamma(s^\alpha, s^\delta)$ network represents a possible meta-analysis data set subject to the sufficient statistics constraints. There are six such data sets.

Now, we move forward through the network to obtain information on the test statistics and the corresponding probabilities. To start, we compute the rank length and the probability length of the connecting arcs. From node $(3, 4, 8)$ to node $(4, 6, 16)$, the rank length is

$$\begin{aligned} r_{1m} &= x_m(s^{\alpha,m} - u) \\ &= 1(6 - 4) \\ &= 2, \end{aligned}$$

and the probability length is

$$\begin{aligned} c_{0m} &= \binom{n_m}{s^{\alpha,m} - u} \\ &= \binom{6}{6 - 4} \\ &= 15. \end{aligned}$$

We continue this process until we have the rank lengths and probability lengths for all connecting arcs, shown in blue in Figure 2.8. To compute the test statistic for our data set, we sum the rank lengths of all arcs in the path of our data set, so $t_{obs} = 0 + 0 + 1 + 2 = 3$. Similarly, the unnormalized probability of our data set is found by multiplying the probability lengths of all arcs in the path of our data set, $3 \times 3 \times 5 \times 15 = 675$.

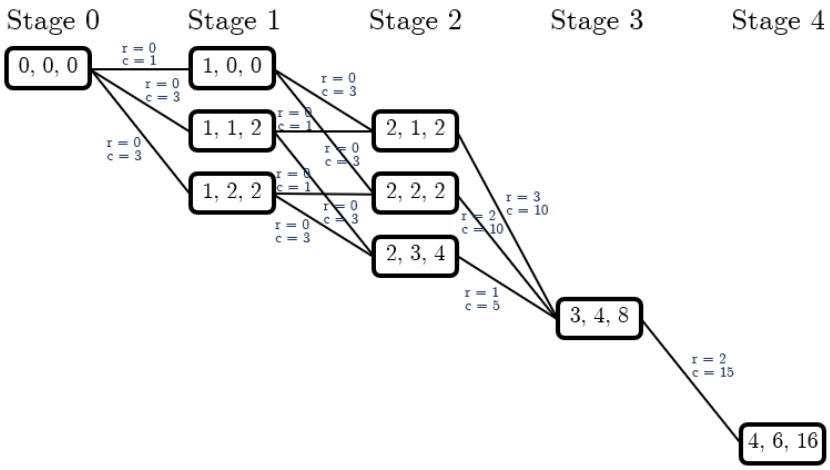

FIGURE 2.8: The $\Gamma(s^{\alpha}, s^{\delta})$ network with the rank lengths and probability lengths of the connecting arcs in blue.

We now need to compute the shortest and longest paths for the test statistic and the total probability for each connecting arc. We start with $SP_1(4,6,16) = LP_1(4,6,16) = 0$, $TP(4,6,16) = 1$. Now, the shortest and longest path lengths and the total probability for node $(3,4,8)$ are

$$\begin{aligned} SP_1(3,4,8) &= r_{1m} + SP_1(4,6,16) \\ &= 2 + 0 \\ &= 2 \\ LP_1(3,4,8) &= r_{1m} + LP_1(4,6,16) \\ &= 2 + 0 \\ &= 2 \\ TP(3,4,8) &= c_{0m} TP(4,6,16) \\ &= 15 \times 1 \\ &= 15. \end{aligned}$$

We continue this process until we reach Stage 1 (see Figure 2.9).

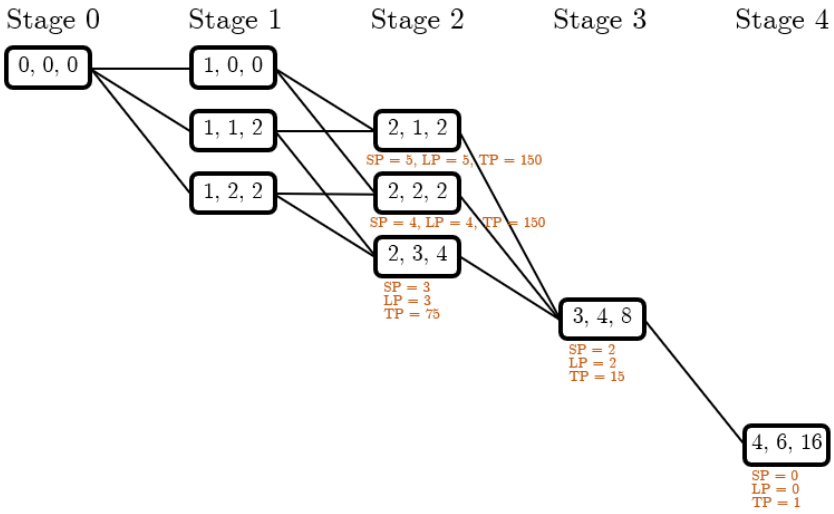

FIGURE 2.9: The $\Gamma(s^{\alpha}, s^{\delta})$ network with the shortest and longest path lengths and the total probability of the connecting arcs in orange for Stages 2 to 4.

Since the nodes in Stage 1 have multiple successor nodes, we need to incorporate information from both successor nodes. For example, the shortest and longest path lengths and the total probability for node $(1, 0, 0)$ are

$$\begin{aligned}
SP_1(1,0,0) &= \min\{r_{1m} + SP_1(2,1,2), r_{1m} + SP_1(2,2,2)\} \\
&= \min\{0+5, 0+4\} \\
&= 4 \\
LP_1(1,0,0) &= \max\{r_{1m} + LP_1(2,1,2), r_{1m} + LP_1(2,2,2)\} \\
&= \max\{0+5, 0+4\} \\
&= 5 \\
TP(1,0,0) &= c_{0m}TP(2,1,2) \times c_{0m}TP(2,2,2) \\
&= (3 \times 150) \times (3 \times 150) \\
&= 900.
\end{aligned}$$

Figure 2.10 shows the completed shortest and longest paths and total probability for all stages. The total probability for node $(0, 0, 0)$ is the normalizing constant.

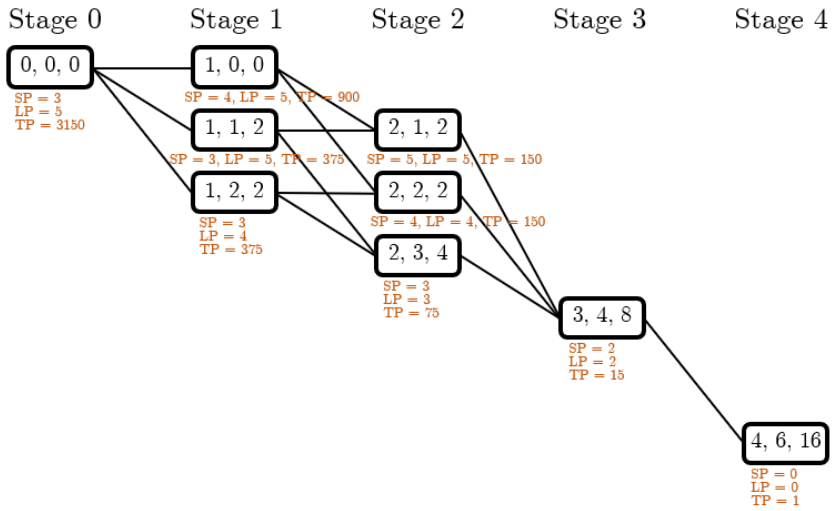

FIGURE 2.10: The $\Gamma(s^{\alpha}, s^{\delta})$ network with the shortest and longest path lengths and the total probability of the connecting arcs in orange.

We can now make the final pass through the network by moving forward, stage by stage, beginning at the initial node $(0, 0, 0)$. We carry forward a set of records $\mathcal{Y}_m$ at each stage for $m = 0, \ldots, k-1$ where each record $d \in \mathcal{Y}_m$ is of the form $d = (m, s^{\alpha,m}, s^{\delta,m}, t_m, h_m)$. We begin with the record $(0, 0, 0, 0, 1)$ associated with the initial node $(0, 0, 0)$. For $(1, 0, 0)$, a successor node of $(0, 0, 0)$, since

$$t_1 + r_{1,1+1} + LP_1\left(1 + 1, s^{\alpha,1+1}, s^{\delta,1+1}\right) = 0 + 0 + 5 \quad \not< \quad 3 = t_{obs},$$

we will keep this record and now must decide whether to merge this record or create a new record. Since there are no other records in $\mathcal{Y}_{1+1} = \mathcal{Y}_2$, we create a new record in $\mathcal{Y}_2$

$$\begin{aligned} d' &= \left(m+1, s^{\alpha,m+1}, s^{\delta,m+1}, t_m + r_{1,m+1}, h_m \binom{n_{m+1}}{s^{\alpha,m+1} - s^{\alpha,m}} \right) \\ &= \left(1, 0, 0, 0+0, 1 \times \binom{3}{0-0} \right) \\ &= (1, 0, 0, 0, 1). \end{aligned}$$

For $(1, 1, 2)$, a successor node of $(0, 0, 0)$, since

$$t_m + r_{1,m+1} + LP_1 \left(m+1, s^{\alpha,m+1}, s^{\delta,m+1} \right) = 0 + 0 + 5 \quad \not< \quad 3 = t_{obs},$$

we will keep this record. As there is a record currently in $\mathcal{Y}_2$, we make the following comparison

$$t_{m+1} = 0 \quad = \quad 0 + 0 = t_m + r_{1,m+1}.$$

Since this comparison results in an equality, we will merge the record from node $(1, 1, 2)$ with the record from the node $(1, 0, 0)$, since both paths, so far, yield the same test statistic. We will combine these records and add both paths' probabilities as such

$$\begin{aligned} d' &= \left(m+1, s^{\alpha,m+1}, s^{\delta,m+1}, t_{m+1}, h_{m+1} + h_m \binom{n_{m+1}}{s^{\alpha,m+1} - s^{\alpha,m}} \right) \\ &= \left(1, 1, 2, 0, 1+1 \times \binom{3}{1-0} \right) \\ &= (1, 1, 2, 0, 4). \end{aligned}$$

For the last node in Stage 1, $(1, 2, 2)$, since

$$t_m + r_{1,m+1} + LP_1 \left(m+1, s^{\alpha,m+1}, s^{\delta,m+1} \right) = 0 + 0 + 4 \quad \not< \quad 3 = t_{obs},$$

we keep this record. Since, for the previous combined record $d' = (1, 1, 2, 0, 4)$,

$$t_{m+1} = 0 \quad = \quad 0 + 0 = t_m + r_{1,m+1},$$

we merge this record from the node $(1, 2, 2)$ with the combined record from nodes $(1, 0, 0)$ and $(1, 1, 2)$ since all three paths, up to this point, yield the same test statistic. We combine these records and add all three paths' probabilities as such

$$\begin{aligned} d' &= \left(m+1, s^{\alpha,m+1}, s^{\delta,m+1}, t_{m+1}, h_{m+1} + h_m \binom{n_{m+1}}{s^{\alpha,m+1} - s^{\alpha,m}} \right) \\ &= \left(1, 2, 2, 0, 4 + 1 \times \binom{3}{2-0} \right) \\ &= (1, 2, 2, 0, 7). \end{aligned}$$

We continue this process until the following records remain

$$\begin{aligned} &d \in \mathcal{Y}_0 : (0, 0, 0, 0, 1), \\ &d \in \mathcal{Y}_1 : (1, 2, 2, 0, 7), \\ &d \in \mathcal{Y}_2 : (2, 3, 4, 0, 30), \\ &d \in \mathcal{Y}_3 : (3, 4, 8, 1, 90), (3, 4, 8, 2, 60), \text{ and } (3, 4, 8, 3, 60), \text{ and} \\ &d \in \mathcal{Y}_4 : (4, 6, 16, 3, 1350), (4, 6, 16, 4, 900), \text{ and } (4, 6, 16, 5, 900). \end{aligned}$$

The records in the last stage contain information about all data sets with test statistics at least as large (values of $t = 3, 4$, and 5) as our observed test statistic

($t_{obs} = 3$). The exact p–value is therefore

$$\begin{aligned}
\Pr(T \geq t_{obs}|H_0, s) &= \sum_{d \in \mathcal{Y}_k} \frac{h_k}{TP(0,0,0)} \\
&= \frac{1350}{3150} + \frac{900}{3150} + \frac{900}{3150} \\
&= 0.4286 + 0.2857 + 0.2857 \\
&= 1.
\end{aligned}$$

A p–value of 1 is consistent with all possible values of the test statistic being as large, or larger, than the observed test statistic.

2.4.3.2 *Variation of Treatment Level Within Study*

As described in Section 2.3.3, allowing for variation of treatment level within study boils down to extra conditioning on the n_{ij}. This adjustment has no effect on the sufficient statistics s^{α} and s^{δ}. As such, the network algorithm works analogous to what was shown in Section 2.4.3.1 with the distinction of having $2k + 1$ stages instead of $k + 1$ stages. Since the network algorithm does not change drastically under this framework, we will provide an application in the next section which illustrates both this adjustment and the adjustment to allow the correlation structure to vary among treatment levels, which is a non–trivial change to the network algorithm.

2.4.3.3 *Variation of Correlation Structure Among Treatment Levels*

For this example, we will use the hypothetical data set shown in Table 2.9. Note that each study now consists of two treatments (treatment and control), and there are now two correlation structures (one for each treatment group). Also, note that the data is sorted by treatment group instead of by study.

TABLE 2.9: Example data set used to illustrate the modified network algorithm as outlined in Sections 2.3.1 and 2.3.2.

Study	1	2	1	2	Sum
Treatment	0	0	1	1	
z_{ij}	2	0	1	1	4
$n_{ij} - z_{ij}$	3	3	1	3	10
n_{ij}	5	3	2	4	14
$x_{ij}z_{ij}$	0	0	1	1	2
$z_{i1}(n_{i1} - z_{i1})$	6	0			6
$z_{i2}(n_{i2} - z_{i2})$			1	3	4

The sufficient statistics for the data set shown in Table 2.9 are

$$\boldsymbol{s} = \begin{cases} s^{\alpha} = \sum_{i=1}^{k}\sum_{j=1}^{2} z_{ij} = 4 & \text{for } \alpha \\ t = \sum_{i=1}^{k}\sum_{j=1}^{2} x_{ij}z_{ij} = 2 & \text{for } \beta \\ s^{\delta_1} = \sum_{i=1}^{k} z_{i1}(n_{i1} - z_{i1}) = 6 & \text{for } \delta_1 \\ s^{\delta_2} = \sum_{i=1}^{k} z_{i2}(n_{i2} - z_{i2}) = 4 & \text{for } \delta_2. \end{cases}$$

We now build the $\Gamma(s^{\alpha})$ network as we did in Section 2.4.3.1, except we have $2k + 1 = 5$ stages instead of $k + 1 = 3$ stages. The resulting network is shown in Figure 2.11. The data set shown in Table 2.9 is represented by the nodes and edges highlighted in red. There are 30 unique paths, or data sets, that fit the constraints of having the same row and column totals and the total number of events equal to four.

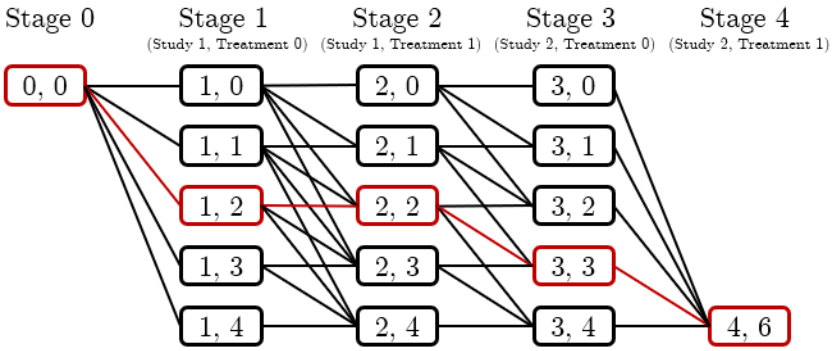

FIGURE 2.11: The $\Gamma(s^{\alpha})$ network with the nodes and edges representing the data in Table 2.9 in red.

To build the $\Gamma(s^{\alpha}, s^{\delta_1}, s^{\delta_2})$ network, we begin creating the terminal node $(4, 4, 6, 4)$. We now proceed by working with the information contributing to s^{δ_2}, stage by stage in reverse, and ending at stage $T + 1 = 2 + 1 = 3$, where T is the number of columns in the data set where the treatment $x = 0$.

Starting with Stage 4, and for node $(3, 0)$, which is a predecessor of node $(4, 4)$,

$$\begin{aligned}\nu_2 &= s^{\delta_2, m} - (s^{\alpha, m} - u)(n_m - s^{\alpha, m} + u) \\ &= 4 - (4 - 0)(4 - 4 + 0) \\ &= 4,\end{aligned}$$

so, we create the triple node $(3, 0, 6, 4)$. Note that all nodes in this phase will have the same third element equal to s^{δ_1}. We continue this process until we find all feasible predecessors of $(4, 4, 6, 4)$. This results in the following nodes at Stage 3: $(3, 0, 6, 4)$, $(3, 1, 6, 1)$, $(3, 2, 6, 0)$, $(3, 3, 6, 1)$, and $(3, 4, 6, 4)$. Note that many of these nodes will turn out to be orphan nodes and will be dropped. We repeat this process for Stage 3, dropping nodes if the partial sum of $s^{\delta_2} \neq 0$. This process results in the following nodes at Stage 2: $(2, 0, 6, 0)$ and $(2, 2, 6, 0)$. This completes the first part of the backward pass, where all

nodes at Stage 2 are in the form $(T, s^{\alpha,T}, s^{\delta_1}, 0)$. Figure 2.12 illustrates this first part of the backward pass.

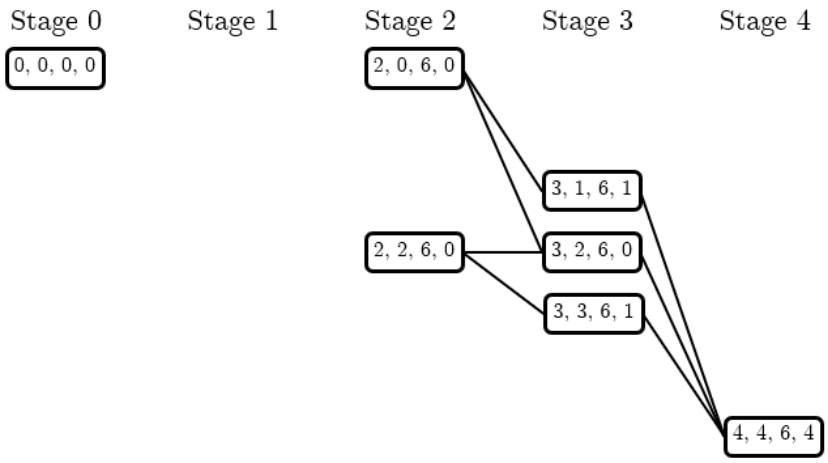

FIGURE 2.12: Building the $\Gamma(s^{\alpha}, s^{\delta_1}, s^{\delta_2})$ network: the first part of the backward pass, utilizing information from the treatment groups.

The two nodes at Stage 2 are the terminal nodes for the second part of the backward pass. Again, we proceed, stage by stage in reverse, until we end at Stage 1. Starting at Stage 2, and for node $(1, 0)$, which is a predecessor of node $(2, 0)$,

$$\begin{aligned}\nu_1 &= s^{\delta_1,m} - (s^{\alpha,m} - u)(n_m - s^{\alpha,m} + u) \\ &= 6 - (0 - 0)(3 - 0 + 0) \\ &= 6,\end{aligned}$$

so, we create the triple node $(1, 0, 6, 0)$. Note that all nodes in this phase will have the same fourth element equal to 0. We continue this process until we find all feasible predecessors of $(2, 0, 6, 0)$ and all feasible predecessors of $(2, 2, 6, 0)$. This results in the following nodes at Stage 1: $(1, 0, 6, 0)$, $(1, 0, 4, 0)$, $(1, 1, 4, 0)$, and $(1, 2, 6, 0)$. Again, many of these nodes end up being orphan nodes and will be dropped. We repeat this process for

Stage 1, dropping nodes if the partial sum of $s^{\delta_1} \neq 0$. This process results in the following nodes at Stage 1: $(1, 1, 4, 0)$ and $(1, 2, 6, 0)$. Both these nodes then connect to the initial node $(0, 0, 0, 0)$. The final $\Gamma(s^{\alpha}, s^{\delta_1}, s^{\delta_2})$ is shown in Figure 2.13. The grey arrows and linear divider are provided for understanding, and the red nodes and edges indicate the path that reflects the data set in Table 2.9.

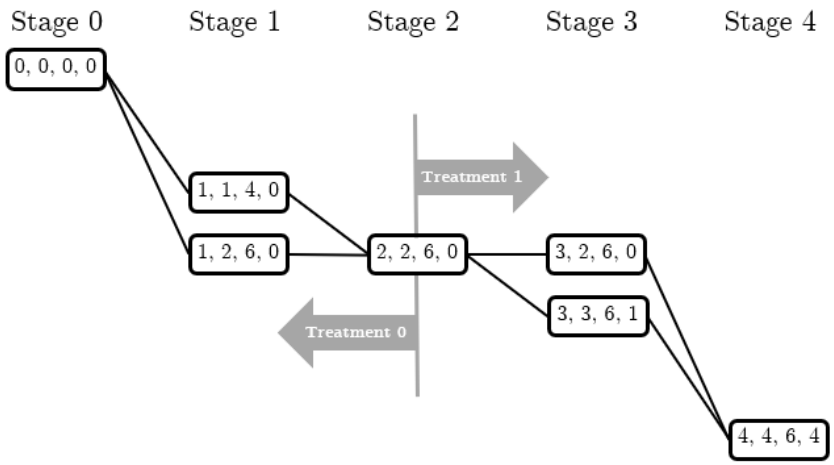

FIGURE 2.13: The $\Gamma(s^{\alpha}, s^{\delta_1}, s^{\delta_2})$ network with the nodes and edges in red representing the data in Table 2.9.

Now that we have the $\Gamma(s^{\alpha}, s^{\delta})$, we continue as shown in Section 2.4.3.1 to collect the rank lengths, probability lengths, and, ultimately, the exact p–value.

2.5 Conclusion

Clearly, small or sparse samples cause problems for many commonly used methods. An additional hindrance is non-negligible heterogeneity, which further encumbers these methods. Many researchers have focused on creating methods for small or sparse data, but few have tried to address these small sample issues coupled with heterogeneity. This,

in part, may be due to guidance from the *Cochrane Handbook for Systematic Reviews of Interventions*: "We would suggest that incorporation of heterogeneity into an estimate of a treatment effect should be a secondary consideration when attempting to produce estimates of effects from sparse data – the primary concern is to discern whether there is any signal of an effect in the data" (Higgins and Green, 2011, chap. 16.9.5). Considering our simulation studies, we would strongly argue against this suggestion, as unaccounted for heterogeneity in rare event data leads to highly inflated type I error rates and overly narrow confidence intervals.

There are a few things of which to take note regarding our simulations. First, our simulation study added a constant value of 0.5, when appropriate, to studies with zero cell counts for the INV, MH, and DSL methods. This continuity correction of 0.5, while commonly used, may not be the best correction. Alternative corrections have been proposed, and some suggest performing sensitivity analyses to determine which correction method is most appropriate (Sweeting et al., 2004). These methods may have lower type I error rates if a different continuity correction factor is used.

Second, for the exact p–value function combination approach, we used the default values in the *gmeta* package. The exact p–value function combination approach is a type of confidence distribution, and confidence distributions are quite general and can be modified to use different weights and linking functions. However, only two linking functions are built into the *gmeta* package; the use of any other linking functions would require additional coding. Nonetheless, this method could, perhaps, perform better if these parameters were modified and tuned.

Third, while our method drastically outperformed the other methods in terms of type I error and confidence interval coverage, there are some drawbacks to our method. Exact methods, in general, are known to be rather conservative when a significance level is used, and our method is no exception. This is illustrated by many of the type I error rates in

Figure 2.1 being below the nominal value and the confidence interval coverage percentages in Figure 2.3 being well over 95%. However, we believe, as does Paul (2018), that, "Given the various biases towards finding statistical significance prevalent in epidemiology today, a strong focus on maintaining a pre–specified level of Type I Error would seem critical." In terms of estimation, our method performed similarly to the commonly used methods and performed worse than the exact p–value function combination method. Despite the exact p-value function combination method being less biased than our method, we believe our method still outperforms this method due to the very poor confidence interval coverage probabilities of the exact p–value function combination method. Again, we prefer wider confidence intervals over confidence intervals that do not contain the true parameter value.

Additionally, in our simulation studies we had to remove data sets in which only one element remained in the reference set after conditioning. With exact methods, over conditioning yields this situation in which our method fails to produce an estimate. Finally, our exact method takes markedly longer computationally than the commonly used methods, and this would especially be true if the events were not rare. However, this method is designed for rare events, and we believe extra computation time is worth the benefit of much lower type I error rates. Additionally, our method is substantially faster than the exact p–value function combination method.

The long computation time is due to the need to enumerate the reference set, as outlined in the network algorithm. A sampling approach to the network algorithm, which involves rejection sampling, or importance sampling (Mehta et al., 1988), has been shown to produce results very similar to working with the entire network and would speed up computation time. The drawback to this sampling technique occurs when the number of paths in the $\Gamma(s(\alpha), s(\delta))$ network is small, which requires sampling a large amount of paths. As such, this would not work well with data sets having rare events because the sampling method would likely produce only samples from the non–event group, and it

would never sample from the event group. For these reasons, we did not attempt to speed up the computations via sampling.

The use of meta–analysis in the medical field has increased dramatically in recent years, and many of these analyses involve rare events. Non–negligible heterogeneity is also common and requires methods that can incorporate a measure of this heterogeneity. As such, it is vital for researchers to have methods that are reliable in these circumstances. In this chapter, we showed, via simulations, that when events are rare and there is a significant amount of heterogeneity present, the traditional methods for meta–analyses have highly inflated type I errors, low confidence interval coverage, and bias. We proposed a new exact method that allows all studies to be pooled without using any arbitrary continuity corrections. Our exact method has much lower type I error rates and higher coverage probabilities than traditional methods when heterogeneity is present. We propose our method be used for meta–analysis data sets when events are rare, and heterogeneity is present.

CHAPTER 3

EXACT NETWORK META–ANALYSIS

3.1 Background

In Chapter 2, we applied extensions to the exact trend test method for correlated binary data to meta–analyses with rare events and heterogeneity. We now apply our method to network meta–analysis. We begin by introducing network meta–analysis, along with the associated assumptions and commonly used methods. We then discuss how our method can be applied to network meta–analysis data, and we assess our method via a simulation study.

3.1.1 Introduction and Terminology

In Chapter 2, we reviewed how meta–analyses have become the standard for evidence–based healthcare decision making. However, there are several drawbacks to a traditional meta–analysis – especially when used in the healthcare industry. A conventional meta–analysis compares only two groups, but often there are more than just two treatments available to treat an illness or disease. In this case, we want to determine, by some means, which of all the possible treatments is most effective. While multiple pairwise meta–analyses could be performed to include all possible treatments, quite often studies directly comparing all combinations of possible treatments are not available. Typically, direct comparisons of two treatments have not been compared head–to–head in a randomized controlled trial. Rather, each treatment is compared with a placebo or standard care in a separate study. This is often the case since, according to the guidelines of the regualtory bodies of various nations, comparing a new treatment to a placebo is

usually sufficient to demonstrate efficacy (Jansen et al., 2011). In these settings, the network meta–analysis approach allows indirect comparisons between treatments. A network meta–analysis is a natural extension of traditional, pairwise meta–analysis, incorporating clinical evidence from both direct and indirect treatment comparisons in a network of trials to determine the effectiveness of multiple treatments. This makes it possible to determine the most effective treatment through effect size estimates and treatment rankings. Network meta–analysis is a relatively novel method, with growing application over the past several years. In early 2017, Tonin et al. (2017) found over 360 published network meta–analyses from over 30 different countries, relating primarily to biomedical studies of conditions, such as cardiovascular disease, oncological disorders, mental health disorders, and infectious diseases.

The starting point for a network meta–analysis often involves a visualization that displays the various studies using a graphical network. Figure 3.1 shows an example of such a network. In this example, there are six different treatments or interventions: A, B, C, D, E, and F. Each included intervention is represented by a treatment node, where the size of the node is determined by the relative sample size. Treatment C has the largest node, indicating this treatment has had the largest number of participants use this intervention; whereas, relatively few participants have ever been assigned to treatment A. Two treatment nodes can be connected by an edge, indicating the two treatments have been directly compared head–to–head in a clinical trial. The size of the edge is proportional to the number of studies which compared the two interventions. In the example shown in Figure 3.1, treatments C and D are connected with the thickest edge, indicating they are the most directly compared treatments. On the other hand, treatments C and F have never been compared directly in a trial. Treatments A and C have been directly compared, but by only a relatively few number of studies.

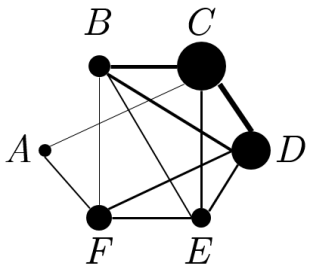

FIGURE 3.1: An example of a graphical network for six treatments: A, B, C, D, E, and F. Each treatment is represented by a node, and edges connecting two nodes indicate one or more studies directly compared the two treatments.

Well–connected networks, where most treatments are directly compared, produce more reliable estimates than poorly connected networks where relatively few direct comparisons have been made. Treatments that are not well connected, like treatment A in this example, have associated estimates that should be interpreted with caution (Mills et al., 2013). Additionally, if there is a severe imbalance in the amount of evidence available for each treatment, the reliability and power of the analysis can suffer (Mills et al., 2011; Thorlund and Mills, 2012). Graphical networks increase the transparency of results and help researchers determine if specific comparisons are selected, underrepresented, or avoided by trialists. This assists researchers in identifying if additional studies are needed and which treatments should be included in those studies. Alternatively, it also helps researchers avoid conducting unnecessary trials where the treatments have already been compared a sufficient number of times.

While traditional meta–analysis involves only direct treatment comparisons, network meta–analysis incorporates both direct and indirect treatment comparisons. This is beneficial since many treatments have not been compared directly, but estimates of a treatment effect can still be obtained indirectly. For example, in Figure 3.1, treatments A and B have never been directly compared, but an indirect comparison can be made by utilizing the studies comparing treatments A and C (AC studies) and the studies comparing treatments B and C (BC studies). Here, the treatment C is called the

"common comparator" or "linking treatment," and the AB comparison is "anchored" or "adjusted" on C.

It is critical to utilize the studies involving the common comparator to avoid "breaking randomization," which would occur if the total number of participants who responded favorably to treatment A from the AC studies were compared with the total number of participants who responded favorably to treatment B from the BC studies. This can lead to the simplistic and possibly erroneous conclusion that whichever treatment as the largest number of positive respondents is the one that is preferred or deemed more effective. Such a conclusion is called a "naïve indirect comparison" and should be avoided as it can lead to biased results (Higgins and Green, 2011, chap. 16.6.2). The trials of AC and BC must be used to help avoid bias from differences in baseline risks, placebo effects, administration of the treatments, etc.

While indirect comparisons can be included in a network meta–analysis, there are a few practical concerns that need to be addressed. First, it is well known that indirect estimates lend to less precision than direct estimates (Glenny et al., 2015). Second, if the two direct treatment comparisons used to compute the indirect comparison are underpowered, the resulting indirect treatment comparison will also be underpowered. Third, indirect treatment comparisons are not randomized. They are considered observational studies and can, therefore, be subject to bias and confounding (Higgins and Green, 2011, chap. 16.6.2). Fourth, the studies involved in the indirect treatment comparison must be similar in all factors that could influence the outcome (Higgins and Green, 2011, chap. 16.6.2). Fifth, some believe indirect comparisons systematically overestimate the treatment effects (Bucher et al., 1997).

Despite these concerns, many argue for the use of indirect comparisons. Jansen et al. (2011) even suggest that indirect comparisons should be considered even when direct comparisons are available. To be able to include an indirect estimate, along with a direct

estimate, a network must have a "closed loop." This occurs when the pairwise comparison has both direct and indirect evidence, causing part of the network to be a triangle, square, or other closed shape, as in the closed triangle made by treatments B, C, and D in Figure 3.1. As an example, the BD comparison has direct study evidence, and it also has indirect evidence from the BC and CD studies. When a network has a closed loop, the method that allows the combination of both direct and indirect evidence is called "multiple–treatments meta–analysis." Many also call this "mixed treatment meta–analysis" or "network meta–analysis." When both direct and indirect information are combined, precision of the estimated treatment effect is improved, and inference to the population sampled is broadened (Coleman et al., 2012). The *Cochrane Handbook for Systematic Reviews of Interventions* recommends reporting two separate analyses when both direct and indirect evidence is available: one with the results from the direct evidence only, the other with results from the combination of the direct and indirect evidence (Higgins and Green, 2011, chap. 16.6.2).

3.1.2 Basic Methods

Bucher et al. (1997) illustrate a way to obtain indirect treatment comparisons, which is often called an "adjusted indirect comparison" or an "anchored indirect comparison." If we let d_{AC} be the comparative effect size estimate of treatment A versus treatment C, and similarly for all treatment comparisons, then the indirect comparative effect size estimate of treatment A versus treatment B is calculated as

$$d_{AB} = d_{AC} - d_{BC},$$

with associated variance

$$\text{Var}\left(d_{AB}\right) = \text{Var}\left(d_{AC}\right) + \text{Var}\left(d_{BC}\right).$$

The trials are considered independent, eliminating the covariance and allowing inclusion of studies with only two treatments. If there is no common comparator for two treatments, a series can be created with multiple common comparators. However, these "multiple adjusted indirect comparisons" suffer from significant uncertainty, which uncertainty is correlated with the number of common comparators in the series (Kim et al., 2013).

One drawback of the method by Bucher et al. (1997) is that it can be used only with two–arm trials and a total of three treatments. A slightly more sophisticated method by Lumley (2002) accounts for both direct and indirect evidence simultaneously and allows for three or more treatments in a network of closed loops. Lumley (2002) also was the one to first use the term "network meta–analysis" to refer to the setting where an indirect treatment comparison can be obtained through more than one common comparator. If the indirect estimate is similar regardless of the common comparator used, Lumley (2002) indicates this is strong evidence that the indirect estimate reflects the true relationship between treatments. However, incoherence occurs if there is discrepancy in the results between common comparators.

Lu and Ades (2004) propose an even more sophisticated model that accommodates three or more treatments for a network of any shape. They call this method "multiple treatment comparison" or "mixed treatment comparison". The method uses a Bayesian framework so that inference can be made simultaneously for all treatments, and treatments can be ranked according to the probability of being the "best" using graphical "rankograms."

The framework of these three methods is often collectively grouped as "network meta–analysis," and throughout the rest of this chapter, we will use this label to refer to

the combination of both direct and indirect estimates used to determine the relative effects of multiple treatments on the same health condition, as does Tonin et al. (2017). Many other methods have been developed under both the frequentist and Bayesian frameworks, but we will not consider these here. It is interesting, though, that the most common software for network meta–analysis (WinBUGS, OpenBUGS, ADDIS, and JAGS) all utilize Bayesian approaches (Tonin et al., 2017). Since intuitively appealing probabilistic treatment rankings are available with the Bayesian framework, this preference is not surprising.

Additionally, both fixed–effect and random–effects methods can be used for network meta–analysis. Under the random–effects framework for traditional meta–analyses, a distribution of treatment effects is assumed, and a measure of heterogeneity is incorporated. Under the random–effects framework for network meta–analysis, another assumption is that effect size estimates differ across comparisons, in addition to differing across studies (Tonin et al., 2017).

3.1.3 Assumptions and Validity Considerations

A network meta–analysis (including the methods of Bucher et al. (1997), Lumley (2002), and Lu and Ades (2004)) employs all the assumptions of a traditional meta–analysis, along with others. These assumptions can be broadly classified relative to *homogeneity*, *similarity*, and *consistency/coherence*. *Homogeneity* requires that there be no relevant differences between studies that are to be pooled for pairwise comparisons. Each pairwise comparison should be assessed both qualitatively, by reviewing the study characteristics, and quantitatively, by using metrics like the I^2 statistic discussed in Chapter 2. Additionally, the two groups of pairwise comparisons used for an indirect comparison should be homogeneous.

Similarity relates to the validity of making indirect comparisons. This assumption requires the studies used in an indirect comparison to be similar in terms of study population, design, outcome measures, and effect modifiers (e.g. age, disease severity, dosage, etc.). This assumption cannot be assessed quantitatively. Researchers should determine, for example, if the treatment chosen as a common comparator is similar enough to be considered a common comparator between the two pairwise comparisons used in an indirect comparison. When this assumption is not met, bias is introduced, and heterogeneity and inconsistency can result (Tonin et al. (2017)).

The *consistency*, or *coherence*, assumption requires there to be no relevant discrepancy, or inconsistency, between direct and indirect evidence. Consistency can be assessed both qualitatively and quantitatively. A closed loop is needed to check for consistency as both direct and indirect evidence is needed. When this assumption is not met, "transitivity" is broken, meaning that if studies show treatment C is better than treatment B, and other studies show treatment B is better than treatment A, one cannot conclude treatment C is better than treatment A. If there is evidence of inconsistency, the *Cochrane Handbook for Systematic Reviews of Interventions* states that the direct estimate should take precedence over the indirect estimate when forming conclusions (Higgins and Green, 2011, chap. 16.6.2).

3.2 Methodology and Innovation

While the graphical networks for network meta–analyses range in complexity, we have chosen to focus on the case where two or more treatments have all been compared to a common treatment (i.e. a control or placebo), but none of the treatments has been directly compared to each other. In this setting, no direct treatment comparisons exist.

Our goal is to obtain indirect treatment comparisons for each treatment comparison using the information provided by the control group comparisons.

We propose applying the modified exact trend test, as described in Chapter 2, to this network meta–analysis situation. Assuming independent trials, we can directly apply this method to each of the direct study comparisons. Once we have an exact estimate of the direct treatment effect for the direct treatment comparisons, we then repeatedly apply the method of Bucher et al. (1997) to obtain the desired exact indirect treatment comparisons.

Without loss of generality, consider as an example Figure 3.2, which displays a hypothetical network for a network meta–analysis including three different treatments, A, B, and D all compared to a common control C (as shown by the solid lines), but never compared to each other (as indicated by the dotted lines). Here, there is direct evidence from studies comparing A to C (AC studies), direct evidence from studies comparing B to C (BC studies), and direct evidence from studies comparing D to C (DC studies), but there are no studies directly comparing any pair of the three treatments A, B, and D. We would like to use the direct AC, BC, and DC study information to obtain indirect information about the following relationships: AB, AD, and BD.

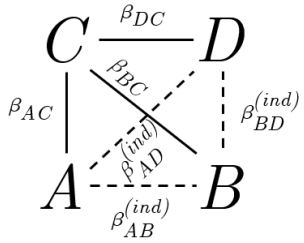

FIGURE 3.2: Small example network with three treatments, A, B, and D, all directly compared to the control, C, but never directly compared to each other. The solid lines indicate the direct comparison, and the dotted lines indicate no direct comparison has been made.

By applying our exact method separately to the AC, BC, and DC studies, the exact estimates of β_{AC}, β_{BC}, and β_{DC} can be obtained. We can then apply the method of Bucher et al. (1997) to obtain the indirect exact estimates of β_{AB}, β_{AD}, and β_{BD}, as

$$\begin{aligned}\hat{\beta}_{AB}^{(ind)} &= \hat{\beta}_{AC} - \hat{\beta}_{BC},\\ \hat{\beta}_{AD}^{(ind)} &= \hat{\beta}_{AC} - \hat{\beta}_{DC}, \text{ and}\\ \hat{\beta}_{BD}^{(ind)} &= \hat{\beta}_{BC} - \hat{\beta}_{DC}.\end{aligned}$$

To obtain an interval estimates of AB, AD, and BD, we utilize the conditional probability for both direct comparisons used in creating the indirect exact point estimates. We will illustrate how this is done for the indirect AB comparison using the direct AC and BC comparisons.

From Chapter 2 Equation 2.10, recall that the conditional probability is given by

$$f_\beta(t(\boldsymbol{z}_{obs})|s^\alpha(\boldsymbol{z}), s^\delta(\boldsymbol{z})) = \frac{C(t(\boldsymbol{z}_{obs})|s^\alpha(\boldsymbol{z}), s^\delta(\boldsymbol{z}))\exp\{\beta t(\boldsymbol{z}_{obs})\}}{\sum_{u=\min(t(\boldsymbol{z}))}^{\max(t(\boldsymbol{z}))} C(u|s^\alpha(\boldsymbol{z}), s^\delta(\boldsymbol{z}))\exp\{\beta u\}}.$$

We use both the conditional probability for the AC comparison, $f_\beta^{AC}(t(\boldsymbol{z}_{obs})|s^\alpha(\boldsymbol{z}), s^\delta(\boldsymbol{z}))$, and the conditional probability for the BC comparison, $f_\beta^{BC}(t(\boldsymbol{z}_{obs})|s^\alpha(\boldsymbol{z}), s^\delta(\boldsymbol{z}))$. Additionally, for both treatment comparisons we define the left and right tails of the distribution of the test statistic given $s^\alpha(\boldsymbol{z})$ and $s^\delta(\boldsymbol{z})$, as defined in Chapter 2, to be, respectively,

$$L_\beta(t(\boldsymbol{z}_{obs})) = \frac{\sum_{u^*=\min(t(\boldsymbol{z}))}^{t(\boldsymbol{z}_{obs})} C(u^*|s^\alpha(\boldsymbol{z}), s^\delta(\boldsymbol{z}))\exp\{\beta u^*\}}{\sum_{u=\min(t(\boldsymbol{z}))}^{\max(t(\boldsymbol{z}))} C(u|s^\alpha(\boldsymbol{z}), s^\delta(\boldsymbol{z}))\exp\{\beta u\}},$$

and

$$R_\beta(t(\boldsymbol{z}_{obs})) = \frac{\sum_{u^*=t(\boldsymbol{z}_{obs})}^{\max(t(\boldsymbol{z}))} C(u^*|s^\alpha(\boldsymbol{z}), s^\delta(\boldsymbol{z}))\exp\{\beta u^*\}}{\sum_{u=\min(t(\boldsymbol{z}))}^{\max(t(\boldsymbol{z}))} C(u|s^\alpha(\boldsymbol{z}), s^\delta(\boldsymbol{z}))\exp\{\beta u\}},$$

giving us $L_\beta^{AC}(t(\boldsymbol{z}_{obs}))$, $R_\beta^{AC}(t(\boldsymbol{z}_{obs}))$, $L_\beta^{BC}(t(\boldsymbol{z}_{obs}))$ and $R_\beta^{BC}(t(\boldsymbol{z}_{obs}))$. From here, to find the lower bound of a two–sided $(1-\alpha)\%$ confidence interval for β for the AB comparison, we combine $R_\beta^{AC}\left(t\left(\boldsymbol{z}_{obs}\right)\right)$ and $R_\beta^{BC}\left(t\left(\boldsymbol{z}_{obs}\right)\right)$ using the p–value combination method of Fisher (1932)

$$R_\beta^{AB}\left(t\left(\boldsymbol{z}_{obs}\right)\right) = \Pr\left\{\chi_{2k}^2 \geq -2\left(\log\left(R_\beta^{AC}\left(t\left(\boldsymbol{z}_{obs}\right)\right)\right) + \log\left(R_\beta^{BC}\left(t\left(\boldsymbol{z}_{obs}\right)\right)\right)\right)\right\},$$

where $k = 2$, the number of p–values to combine. Similarly, for the upper bound of a two–sided $(1-\alpha)\%$ confidence interval for β for the AB comparison, we combine $L_\beta^{AC}\left(t\left(\boldsymbol{z}_{obs}\right)\right)$ and $L_\beta^{BC}\left(t\left(\boldsymbol{z}_{obs}\right)\right)$ as

$$L_\beta^{AB}\left(t\left(\boldsymbol{z}_{obs}\right)\right) = \Pr\left\{\chi_{2k}^2 \geq -2\left(\log\left(L_\beta^{AC}\left(t\left(\boldsymbol{z}_{obs}\right)\right)\right) + \log\left(L_\beta^{BC}\left(t\left(\boldsymbol{z}_{obs}\right)\right)\right)\right)\right\}.$$

Letting β_-^{AB} be the lower bound of the confidence interval for β for the AB comparison and β_+^{AB} be the upper bound, then β_-^{AB} satisfies

$$R_{\beta_-^{AB}}(t(\boldsymbol{z}_{obs})) = \frac{\alpha}{2},$$

and β_+^{AB} satisfies

$$L_{\beta_+^{AB}}(t(\boldsymbol{z}_{obs})) = \frac{\alpha}{2}.$$

Thus, we can find an indirect confidence interval for the AB comparison using information from the AC and BC comparisons.

3.3 Application

In this section, we analyze our method via a simulation study, which is based on a simulation study conducted by Mills et al. (2011). We generate networks with two treatments, A and B, compared to a common control, C, but not directly compared to each other, as illustrated in Figure 3.3. We define k_{AC} as the number of studies comparing treatments A and C; k_{BC} as the number of studies comparing treatments B and C; $n_{\min}$ as the minimum number of participants in a study; $n_{\max}$ as the maximum number of participants in a study; p_{nC} the proportion of participants in the control group C; π_C as the true average event rate in the common comparator group C; OR_{AC} as the true relative effect of AC, quantified as an odds ratio; OR_{BC} as the true relative effect of BC, quantified as an odds ratio; and τ^2 as the between–study variance (assumed constant across the AC and BC comparisons). We generate data sets under the following model (written for the BC comparisons, with the AC comparison written similarly), where i indexes study.

$$
\begin{aligned}
Y_{Ci} &\sim \text{Binomial}\left(n_{Ci}, \pi_{Ci}\right), \\
Y_{Bi} &\sim \text{Binomial}\left(n_{Bi}, \pi_{Bi}\right), \\
n_i &\sim \text{Uniform}\left(n_{\min}, n_{\max}\right), \\
n_{Ci} &= n_i\left(p_{nC}\right), \\
n_{Ci} &= n_i - n_{Ci}, \\
\pi_{Ci} &\sim \text{Uniform}\left(\pi_C - \frac{\pi_C}{2}, \pi_C + \frac{\pi_C}{2}\right), \\
\pi_{Bi} &= \frac{\pi_{Ci} \exp \ln\left(OR_{BC,i}\right)}{1 - \pi_{Ci} + \pi_{Ci} \exp \ln\left(OR_{BC,i}\right)}, \text{ and} \\
\ln\left(OR_{BC,i}\right) &\sim \text{Normal}\left(\ln\left(OR_{BC}\right), \tau^2\right).
\end{aligned}
$$

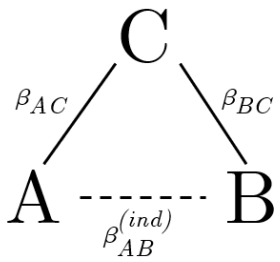

FIGURE 3.3: The graphical network on which the simulation study is based.

All data sets were generated under the null, $OR_{AC} = 1$ and $OR_{BC} = 1$, with $n_{\min} = 10$ and $n_{\max} = 50$. We vary k_{AC} while holding constant $k_{BC} = 5$. We let $\pi_C = 0.10$, and we use the nominal significance level of $\alpha = 0.05$. Table 3.1 shows the eight scenarios, each with varying values of k_{AC}, p_{nC}, and τ^2, that we use for our simulation study.

TABLE 3.1: Combination of values for the parameters used in eight simulation study scenarios.

Scenario	k_{AC}	p_{nC}	τ^2
1	5	0.50	0
2	5	0.50	0.4
3	5	0.25	0
4	5	0.25	0.4
5	10	0.50	0
6	10	0.50	0.4
7	10	0.25	0
8	10	0.25	0.4

Using the formula from Bucher et al. (1997),

$OR_{AB} = \exp\{log\,(OR_{AC}) - log\,(OR_{BC})\}$, knowing $OR_{AC} = 1$ and $OR_{BC} = 1$ determines

$OR_{AB} = 1$. We can also determine the true average event rates for treatment A, π_A, and for treatment B, π_B, just by knowing $\pi_C = 0.10$ using these formulas

$$\pi_A = \frac{OR_{AC}\frac{\pi_C}{1-\pi_C}}{1 + OR_{AC}\frac{\pi_C}{\pi_C}}, \text{ and}$$
$$\pi_B = \frac{OR_{BC}\frac{\pi_C}{1-\pi_C}}{1 + OR_{BC}\frac{\pi_C}{\pi_C}},$$

resulting in $\pi_A = 0.10$ and $\pi_B = 0.10$. Thus, all treatments have the same true average event rate of 0.10, and the true odds ratios between each pair of treatments is 1.

For each scenario, we simulate 2,500 data sets. As with the simulation study in Chapter 2, we remove data sets having zero events across all studies for either the treatment group or the control group, and we exclude data sets that generate test statistic distributions containing only one value of the test statistic or where the observed test statistic is at either extreme of the distribution.

We then apply our exact permutation–based approach to these simulated data sets, as outlined in Section 3.2. We compare our method with the random–effects DerSimonian and Laird (DSL) method. The DSL method was used to estimate $\log(OR_{AC})$ and $\log(OR_{BC})$ with their associated standard errors. Bucher's method was then used to obtain an adjusted indirect comparison for AB. We use the same C code for our simulations as used in Chapter 2 (see Appendix C), along with utilizing the computational power of the CHPC at the University of Utah. We use R code to process the results of our method and to compute the results from the DSL method (see Appendix D).

Figure 3.4 compares the bias of our exact permutation–based method and the DSL method under the eight simulation settings. For Scenarios 1 and 2, our method performs similarly to the DSL method, with both methods having very little bias. Scenarios 3 and 4 indicate that both methods are biased, with our method yielding slightly more bias. Scenarios 7 and 8 show similar results, except the magnitude of the difference in bias

between the two methods is even greater. Scenarios 5 and 6 indicate our method is very biased when compared with the DSL method which has minimal bias. Across the board, it appears that the presence of heterogeneity did not substantially influence the results of these two methods. The DSL method seems to perform best when the proportion of participants in the control group versus the treatment group is 0.50. When this proportion shrinks to 0.25, the DSL method's performance weakens. The imbalance in the number of studies in the AC and BC treatment comparison groups (Scenarios 5 through 8) also may have slightly increased the bias in the DSL method. We cannot determine any noticeable patterns that cause our exact permutation–based method to have substantial bias.

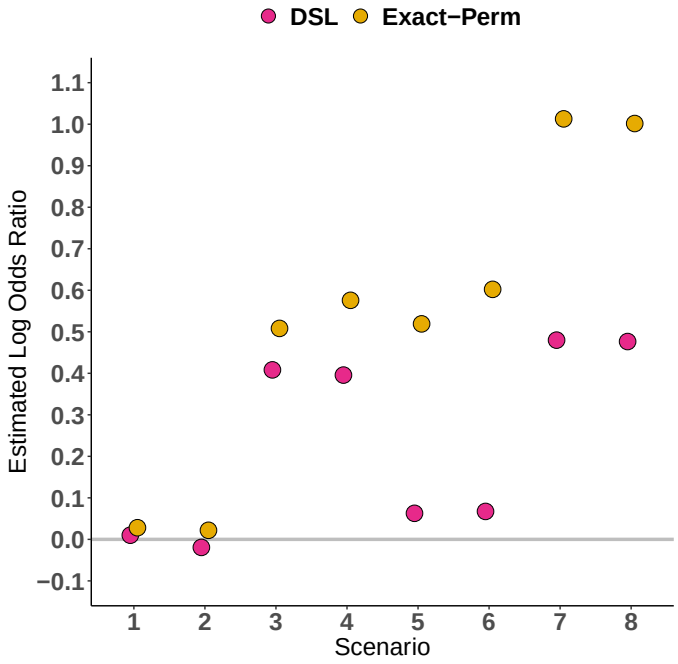

FIGURE 3.4: This figure compares the bias of our exact permutation–based method and the DerSimonian and Laird (DSL) method under the eight network meta–analysis simulation scenarios. The horizontal, grey line at an odds ratio equal to one indicates no bias.

Figure 3.5 compares the confidence interval coverage of our exact permutation–based

method and the DSL method under the eight simulation settings. For Scenarios 1 through 5 and Scenario 7, both methods have average confidence interval coverages that do not exceed the nominal coverage value of 95%. The exact permutation–based method produces low coverage in Scenarios 6 and 8, indicating that the presence of heterogeneity, combined with an imbalance in the number of studies between the AC and BC treatment comparison groups, cause this method to underperform. The DSL method only slightly drops below the nominal level in Scenario 8.

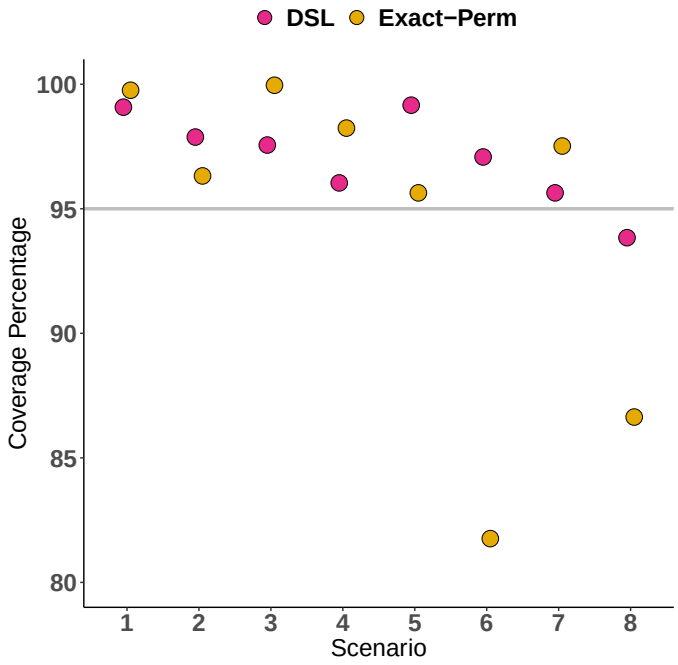

FIGURE 3.5: This figure compares the confidence interval coverage of our exact permutation–based method and the DerSimonian and Laird (DSL) method under the eight network meta–analysis simulation scenarios. The horizontal, grey line marks the nominal 95% coverage value.

3.4 Conclusion

In this chapter, we focused on a relatively simple network for network meta–analysis where treatments are directly compared to a common control but not to each other. We applied our exact permutation–based test and modified exact interval estimation procedure to such data and compared these approaches to the standard DerSimonian and Laird method through simulation. Although the DerSimonian and Laird method appears to have outperformed the exact method, for computational reasons we were unfortunately constrained in our simulation to use baseline response rates in the neighborhood of 10%. It was only at this setting that we were able to generate a sufficiently informative number of samples to yield reliable results. As permutation–based procedures generally provide greater advantages for relatively sparse samples, additional study under small–sample settings would provide more useful insights between these two approaches.

Consistent with results from other published work, as well as with our findings within Chapter 2 (particularly within Section 2.4.1), there is appreciable bias in certain settings. The permutation–based methods for network meta–analysis yielded comparatively lower confidence coverage in some scenarios, relative to the good confidence coverage of our method in Chapter 2. However, as noted above, for practical reasons, our simulations for this chapter were carried out with an average 10% response rate, whereas our simulations in Chapter 2 used much smaller event rates of 1.8% and 4.7%. Nevertheless, our simulation study provides some direction with regard to how we can further improve our method. We plan to investigate alternative methods to Bucher et al. (1997)'s method for computing indirect estimates. We also plan to apply Monte Carlo sampling to the exact test, which should allow us to use smaller event rates in our simulations.

CHAPTER 4

COMBINING CONFIDENCE DISTRIBUTIONS FOR RARE EVENT META–ANALYSIS

4.1 Background

When considering statistical inference, a common approach is to compute a single point estimate of the parameter of interest. Interval estimates are also used, providing a probable range of values for the parameter of interest. A step further would be to not only estimate an interval based on some level of confidence, but to estimate the whole distribution of the parameter of interest. Distribution estimators come in a variety of familiar forms, including the bootstrap distribution, the Bayesian posterior distribution, the empirical likelihood, and the p–value function (also called "significance functions," see Fraser (1991)). A distribution estimator can alternatively be called a confidence distribution (CD) since all levels of confidence can be represented in this distribution. A CD yields richer information about the parameter of interest than point estimates, interval estimates, or single p–values since this information, and more, is contained within CDs (see Figure 4.1). The shape of the distribution can provide information about the size of the study and precision of the point estimate – a narrow distribution indicates a large study with high precision; whereas, a wide distribution indicates a small study with low precision. Succinctly, a confidence distribution is a sample–dependent distribution function on the parameter space that represents all possible confidence intervals for the parameter.

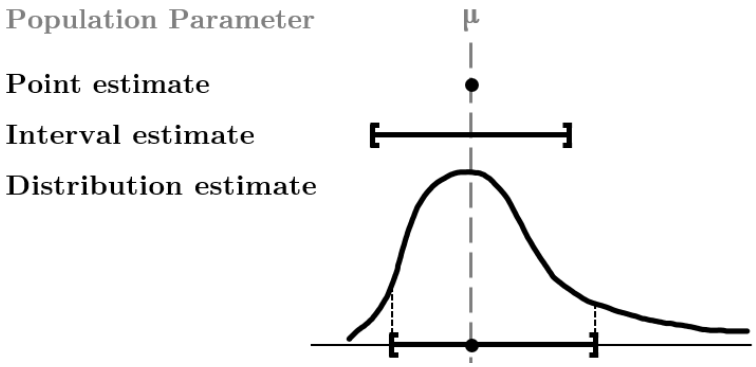

FIGURE 4.1: This figure illustrates different ways to estimate the population parameter of interest. Point estimates, such as the log odds ratio or the sample average, and interval estimates, like a 95% confidence interval, are commonly used in statistical inference. An estimate of the distribution of the population parameter, such as a confidence distribution, can also be obtained. Distribution estimators are appealing since they contain information about both the point estimate and the confidence interval, as shown in the bottom curve, in addition to much more information.

The idea of CDs can be traced back to Ronald Fisher (Fisher, 1973); however, until recently, CDs have received little attention. It wasn't until the 1990s that more attention and development was placed on CDs (Efron, 1993), and, in 2005, Singh et al. (2005) introduced the idea of combining CDs. This approach is especially applicable when combining information from multiple studies, as in a meta–analysis. Since CDs provide more information than point estimates, confidence intervals, or p–values alone, it seems smart to apply them to a meta–analysis with rare events or small sample sizes where it is important to utilize and accurately summarize the available data. Under the CD framework, each study is summarized using a CD, which can be based on an exact test, and the individual studies' CDs are then pooled into one combined CD. If an exact test is used, the CD preserves the possible asymmetry of the distribution. Another appealing trait of a CD is that it is always obtainable even if a study has zero events.

CDs have become the generalized linear model for combining information from different sources. Xie et al. (2011) refer to CDs as "a unifying framework for

meta–analysis," and they show how most existing methods for meta–analysis fit nicely under this framework as special types of CDs. The classical p–value combination approaches (Fisher's method, Stouffer's method, etc.) and the common model–based approaches (fixed–effect, random–effects, etc.) all fit in this unifying framework of CDs.

To illustrate how CDs are used to combine information, consider the results shown in Table 4.1 from two hypothetical studies. We will use these studies to illustrate how CDs can be used to both summarize and combine the two studies. The plots shown throughout this example were created using code from the *gmeta* R package (Yang et al., 2016a), and we will focus on combining p–value functions, which are a type of CD and will be discussed more later.

TABLE 4.1: Results from two hypothetical studies that will be used to illustrate the use of CDs.

Study 1	Event	Non-Event	Total
Trt	3	1	4
Ctrl	1	3	4
Total	4	4	8

Study 2	Event	Non-Event	Total
Trt	2	3	5
Ctrl	2	4	6
Total	4	7	11

The p–values for the two tables based on Fisher's exact test with the mid–p correction are 0.13 and 0.43, respectively. These were computed under the hypotheses $H_0 : \log(OR) = 0$ and $H_A : \log(OR) > 0$. To create a p–value function, we obtain p–values for many different null values (not just 0). The resulting p–value functions for our example, $p_1(\cdot)$ and $p_2(\cdot)$, are shown in Figure 4.2 as cumulative distribution functions (CDFs). For this example, the null value for the log odds ratio was varied from -4 to 8. The dashed magenta line marks the p–value under the null value of 0, and the dotted teal line indicates the median log odds ratio of the p–value function. The median of the i^{th}

p–value function is defined as $p_i^{-1}(0.5)$. The median of a CD is often used as the point estimate for the parameter of interest.

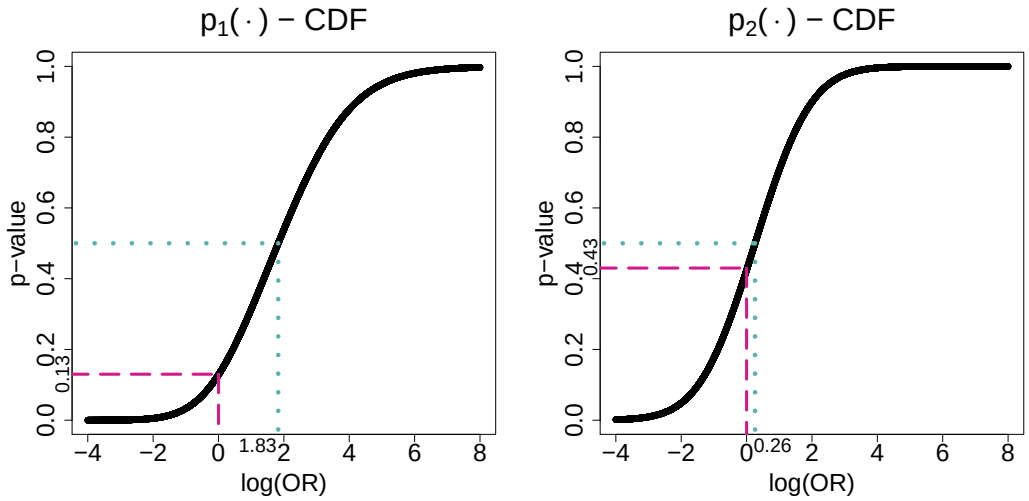

FIGURE 4.2: Individual p–value functions plotted as CDFs for the two studies shown in Table 4.1. The dashed magenta lines indicate the p–value under the null value of 0, and the dotted teal lines indicate the median of the p–value function.

An alternative way to plot a p–value function is to plot the "confidence curve" (CV) which shows both the p–value and all levels of confidence (see Figure 4.3). Here, the median is at the peak of the distribution, indicated by the dotted teal line. A 95% confidence interval is shown by the dashed gold lines. A $(1-\alpha)\%$ confidence interval for the i^{th} p–value function is given by $\left(p_i^{-1}\left(\frac{\alpha}{2}\right), p_i^{-1}\left(1-\frac{\alpha}{2}\right)\right)$. With this representation, it is easy to see all levels of confidence and gain a rich view of the distribution, which, for Study 1, has preserved the inherent asymmetry. Additionally, since Study 2 has a narrower curve, it is more precise than Study 1, due to its larger sample size. Study 2 also indicates stronger evidence of no treatment effect than Study 1; however, since 0, emphasized by the dashed black line, is contained in both confidence intervals, both studies suggest a non–significant effect.

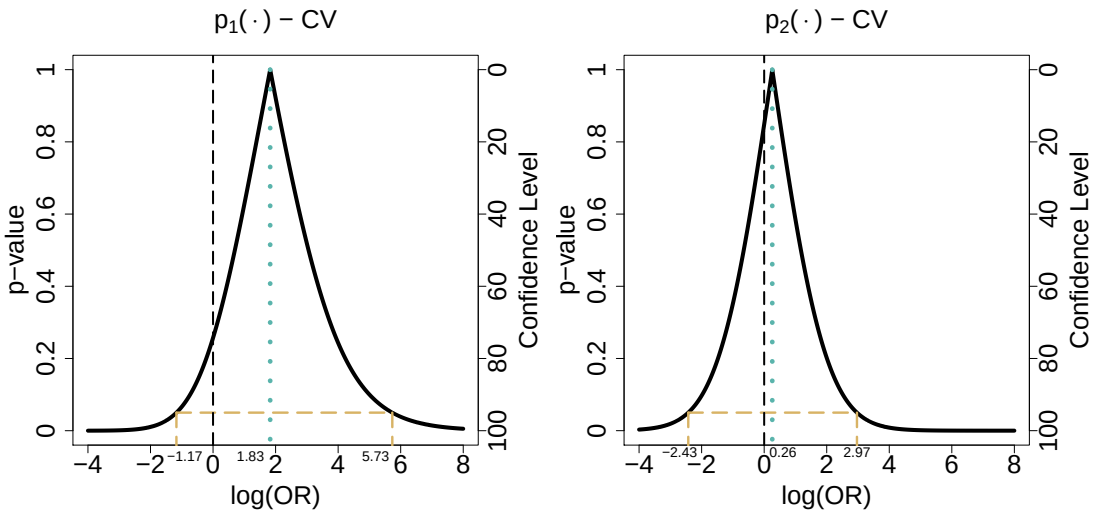

FIGURE 4.3: Individual p–value functions plotted as "confidence curves" for the two studies shown in Table 4.1. The dotted teal line indicates the median of the p–value function, and the dashed gold lines show the 95% confidence interval. The dashed black line indicates the null value of 0, which both confidence intervals cover, indicating no significant effect.

Now that we have p–value functions for the individual studies, we can combine them into one p–value function $H^{(c)}(\cdot)$. For this example, we will use the combination method described later in Section 4.2.3. Figure 4.4 shows the combined CDF with the CDFs of the two individual studies overlaid in gray. Again, the median is marked in a dotted teal line and is used as the point estimate for the combined effect. The p–value under the null value of 0 is marked in the dashed magenta line.

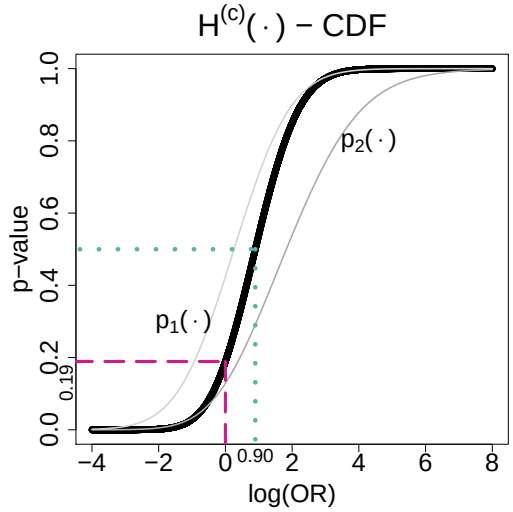

FIGURE 4.4: The combined p–value function of the two studies shown in Table 4.1 plotted as a CDF. The dotted teal line indicates the median of the combined p–value function. The dashed magenta line shows the p–value under the null value of 0.

The combined p–value function can also be displayed as a CV, as shown in Figure 4.5. From this plot, it is easy to see the combined estimate of 0.90 (the median marked with the dotted teal line) with a 95% confidence interval of $(-1.06, 3.04)$ (marked in the dashed gold lines). This curve is also narrower than the two individual study CVs, which is indicative of pooling studies together, providing a more precise estimate than the individual studies. Additionally, since the black dashed line, indicating the null value of 0, is contained in the interval, these two studies, pooled together, provide no evidence of a significant effect.

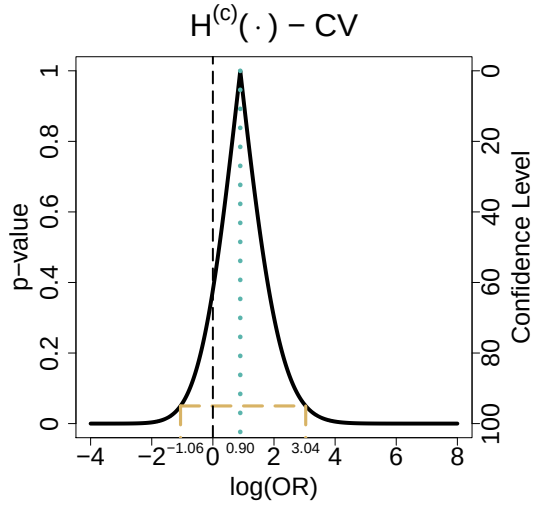

FIGURE 4.5: The combined p–value function of the two studies shown in Table 4.1 plotted as a CV. The dotted teal line indicates the median of the combined p–value function, and the dashed gold lines show the 95% confidence interval for the combined function. The dashed black line indicates the null value of 0, which is contained in the confidence interval, indicating no significant effect.

The simple example just illustrated outlines the basic approach of combining CDs. The methodology and the mathematics on how to combine these CDs is the focus of this chapter. We are interested in discovering a combination approach that works well in the rare event meta–analysis setting. We believe CDs have the potential to perform better in the rare event setting than the traditional methods used for meta–analyses, which were shown in Chapter 2 to perform poorly. As such, we will utilize two articles that have applied CDs for meta–analyses with rare events. Tian et al. (2009) developed an exact method of combining confidence intervals which has been shown to fall under the CD framework (Yang et al., 2016b). Liu et al. (2014) combined p–value functions using Fisher's exact test, which are a type of CD, and which we reviewed briefly in Chapter 1. Additionally, the well–known Fisher's p–value combination method (Fisher, 1932) is a type of CD and can be used in this setting.

While several confidence distribution methods exist, no comparisons have been made

to determine which method is best suited for meta-analyses with rare events and non–negligible heterogeneity. This chapter compares the performance of these three CD approaches. We also propose and compare modifications of these CD methods to better handle situations with rare events or heterogeneity. We begin by providing a formal definition of a CD. We then summarize each of the three CD methods. Next, we outline some modifications to these procedures, and we end by comparing these methods and recommending one to be used for meta–analyses with rare events and significant heterogeneity.

4.2 Methodology

In this section, we start by providing a formal definition of a CD, as well as a general formula to combine CDs for a meta–analysis. Next, we briefly outline each of the three CD approaches we are considering: the exact method of combining confidence intervals as in Tian et al. (2009), the exact method of combining p–value functions as in Liu et al. (2014), and the method of combining p–values as in Fisher (1932). We show how these methods fall under the unifying CD framework.

4.2.1 Confidence Distribution Definition

The following is a formal definition of a CD, as given in Singh et al. (2005). Let Θ be the parameter space of the unknown parameter θ, and let $\mathcal{X}$ be the sample space corresponding to the sample $x = \{x_1, \ldots, x_n\}$. A function $H_n(\cdot) = H_n(x, \cdot)$ on $\mathcal{X} \times \Theta \to [0, 1]$ is a CD for θ if

1. for each given $x \in \mathcal{X}$, $H_n(\cdot)$ is a continuous cumulative distribution function on Θ

2. when $\theta = \theta_0$ (the true parameter value), $H_n(\theta_0) \equiv H_n(x, \theta_0) \sim U[0,1]$, thus providing confidence intervals for all levels for θ_0.

In the meta–analysis setting, suppose there are k independent studies that estimate a common parameter of interest θ. Let x_i represent the data of the i^{th} study. We first construct a CD for each of the i studies $H_i(\cdot) = H_i(x_i, \cdot)$ for θ. The combined CD, which contains information from all k studies, as defined by Singh et al. (2005) and Xie et al. (2011), is

$$H^{(c)}(\theta) = \Pr\left\{g_c\left(U_1, \ldots, U_k\right) \leq g_c\left(H_1\left(\theta\right), \ldots, H_k\left(\theta\right)\right)\right\}, \tag{4.1}$$

where $g_c(u_1, \ldots, u_k)$ is a monotonic function on $[0,1] \to \mathbb{R}$, and $U_1, \ldots, U_k$ are independent $U[0,1]$ random variables. In the meta–analysis framework, an appealing choice for g_c is

$$g_c(u_1, \ldots, u_k) = w_1 a_0(u_1) + \cdots + w_k a_0(u_k), \tag{4.2}$$

where $a_0(\cdot)$ is any monotonically increasing function (transformation function), and w_i are the weights for each study given $w_i > 0$. Throughout the rest of this chapter, we will refer to Equations 4.1 and 4.2 as the "general CD framework."

4.2.2 Combining Confidence Intervals as in Tian et al. (2009)

In their paper, Tian et al. (2009) present an exact meta–analysis approach that combines confidence intervals with the risk difference as the effect size estimator. They do not present their method under the general CD framework, but Yang et al. (2016b) illustrate how this method does fit nicely within the general CD framework. Here, for consistency, we will present the method of Tian et al. (2009) in the context of the general CD framework using Equations 4.1 and 4.2 as demonstrated by Yang et al. (2016b).

Letting $a_0(\cdot) = \text{logit}(\cdot)$, the method of Tian et al. (2009) under the general CD framework is written as

$$\begin{aligned}
H^{(c)}(s) &= \Pr\left\{g_c\left(U_1,\ldots,U_k\right) \le g_c\left(H_1(s),\ldots,H_k(s)\right)\right\} \\
&= \Pr\left\{w_1 a_0(u_1) + \cdots + w_k a_0(u_k) \le w_1 a_0(H_1(s)) + \cdots + w_k a_0(H_k(s))\right\} \\
&= \Pr\Bigg\{w_1 \log\left(\frac{u_1}{1-u_1}\right) + \cdots + w_k \log\left(\frac{u_k}{1-u_k}\right) \\
&\qquad\qquad \le w_1 \log\left(\frac{H_1(s)}{1-H_1(s)}\right) + \cdots + w_k \log\left(\frac{H_k(s)}{1-H_k(s)}\right)\Bigg\} \\
&= \Pr\left\{\sum_{i=1}^{k} w_i \log\left(\frac{u_i}{1-u_i}\right) \le \sum_{i=1}^{k} w_i \log\left(\frac{H_i(s)}{1-H_i(s)}\right)\right\}. \qquad (4.3)
\end{aligned}$$

George and Mudholkar (1977) state that the sum of k logits will follow the t-distribution $-\pi\sqrt{\frac{k(5k+2)}{3(5k+4)}}t_\lambda$ with degrees of freedom $\lambda = 5k+4$. The weighted version of this combination method also follows a t-distribution $-\pi\sqrt{\frac{\lambda-2}{\lambda}\sum_{i=1}^{k}\frac{w_i^2}{3}}t_\lambda$ with degrees of freedom

$$\lambda = 4 + \frac{5}{\sum_{i=1}^{k}\frac{w_i^4}{(\sum_{i=1}^{k} w_i^2)^2}}.$$

So, given this distribution and continuing from Equation 4.3,

$$\begin{aligned}
H^{(c)}(s) &= \Pr\left\{\sum_{i=1}^{k} w_i \log\left(\frac{u_i}{1-u_i}\right) \le \sum_{i=1}^{k} w_i \log\left(\frac{H_i(s)}{1-H_i(s)}\right)\right\} \\
&= \Pr\Bigg\{-\frac{1}{\pi\sqrt{\frac{\lambda-2}{\lambda}\sum_{i=1}^{k}\frac{w_i^2}{3}}}\sum_{i=1}^{k} w_i \log\left(\frac{u_i}{1-u_i}\right) \\
&\qquad\qquad \ge -\frac{1}{\pi\sqrt{\frac{\lambda-2}{\lambda}\sum_{i=1}^{k}\frac{w_i^2}{3}}}\sum_{i=1}^{k} w_i \log\left(\frac{H_i(s)}{1-H_i(s)}\right)\Bigg\} \\
&= \Pr\left\{-\frac{1}{\pi\sqrt{\frac{\lambda-2}{\lambda}\sum_{i=1}^{k}\frac{w_i^2}{3}}}\sum_{i=1}^{k} w_i \log\left(\frac{H_i(s)}{1-H_i(s)}\right) \le t_\lambda\right\},
\end{aligned}$$

where

$$\lambda = 4 + \frac{5}{\sum_{i=1}^{k} \frac{w_i^4}{(\sum_{i=1}^{k} w_i^2)^2}},$$

and the weight for each study, as defined by Tian et al. (2009), is based solely on the sample size of the study.

This is the method of Tian et al. (2009) under the general CD framework. The original method of Tian et al. (2009) is programmed in the *exactmeta* package, and a slightly modified version of their original method is programmed in the *gmeta* package in order to be contained with the other CD approaches.

4.2.3 Combining p–value Functions as in Liu et al. (2014)

Liu et al. (2014) propose the combination of p–value functions, which are a special type of CD, for an exact meta–analysis. We start by providing the definition of a p–value function, and we then show how this combination method can be written in the general CD framework. We provided additional details of this method in Chapter 1, and we only reproduce the relevant details here.

Let x, y denote the sample, then $p = p(s; x, y)$ denotes a p–value computed based on a given exact test (Liu et al. (2014) use the mid–p correction of Fisher's exact test) for testing $H_0 : \theta = s$ versus $H_A : \theta > s$ where s is some fixed value in the parameter space. Here, we see the p–value depends on both the sample x, y and the value of s, just like a CD does. Given the sample x, y, as the value of s varies, $p(s) = p(s; x, y)$ is a p–value function on the parameter space of θ, and it is a CD (Singh et al., 2005; Xie et al., 2011). To combine these p–value functions under the general CD framework, Liu et al. (2014) use $a_0(\cdot) = \Phi^{-1}(\cdot)$, where $\Phi^{-1}(\cdot)$ represents the inverse cumulative distribution function of the

standard normal distribution. The weights used for the general CD framework are given by

$$w_i \propto \sqrt{\frac{1}{n_i\pi_{1i}(1-\pi_{1i})} + \frac{1}{n_i\pi_{0i}(1-\pi_{0i})}},$$

where π_{1i} and π_{0i}, the probabilities of an event in the treatment and control group, respectively, are estimated empirically by borrowing information from both the i^{th} study itself, as well as other studies by utilizing the "average level" of the event rates in the other studies. For details on how π_{1i} and π_{0i} are computed, refer to the Appendix in Liu et al. (2014).

Thus, under the general CD framework, the overall combined p–value function, or CD, is

$$\begin{aligned}
H^{(c)}(s) &= \Pr\left\{g_c\left(U_1,\ldots,U_k\right) \le g_c\left(H_1(s),\ldots,H_k(s)\right)\right\} \\
&= \Pr\left\{w_1a_0(u_1)+\cdots+w_ka_0(u_k) \le w_1a_0(p_1(s))+\cdots+w_ka_0(p_k(s))\right\} \\
&= \Pr\left\{w_1\Phi^{-1}(u_1)+\cdots+w_k\Phi^{-1}(u_k) \le w_1\Phi^{-1}(p_1(s))+\cdots+w_k\Phi^{-1}(p_k(s))\right\} \\
&= \Pr\left\{\sum_{i=1}^{k} w_i\Phi^{-1}(u_i) \le \sum_{i=1}^{k} w_i\Phi^{-1}(p_i(s))\right\}. \qquad (4.4)
\end{aligned}$$

From here, we can use a well–known theorem which states

If $X_1, X_2, \ldots, X_n$ are mutually independent normal random variables with means $\mu_1, \mu_2, \ldots, \mu_n$ and variances $\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2$, then the linear combination $Y = \sum_{i=1}^n c_iX_i$ follows the normal distribution $N(\sum_{i=1}^n c_i\mu_i, \sum_{i=1}^n c_i^2\sigma_i^2)$.

We know $\Phi^{-1}(u_i) \sim N(0,1)$, so let $X_i = \Phi^{-1}(u_i)$ and $Y = \sum_{i=1}^k w_iX_i$, then, by the theorem above, $Y \sim N(\sum_{i=1}^n w_i \times 0, \sum_{i=1}^n w_i^2 \times 1) \Rightarrow Y \sim N(0, \sum_{i=1}^n w_i^2)$.

Now, by the same theorem, if we multiply Y by the square root of the inverse of its variance, then that product is normally distributed with mean 0 and variance

$\left(\frac{1}{\sqrt{\sum_{i=1}^{n} w_i^2}}\right)^2 \times \sum_{i=1}^{n} w_i^2 = 1$. So, $Z = \frac{1}{\sqrt{\sum_{i=1}^{n} w_i^2}} \sum_{i=1}^{k} w_i \Phi^{-1}(u_i)$ follows a standard normal distribution.

Given these results and continuing from Equation 4.4,

$$\begin{aligned} H^{(c)}(s) &= \Pr\left\{ \sum_{i=1}^{k} w_i \Phi^{-1}(u_i) \leq \sum_{i=1}^{k} w_i \Phi^{-1}(p_i(s)) \right\} \\ &= \Pr\left\{ \frac{1}{\sqrt{\sum_{i=1}^{n} w_i^2}} \sum_{i=1}^{k} w_i \Phi^{-1}(u_i) \leq \frac{1}{\sqrt{\sum_{i=1}^{n} w_i^2}} \sum_{i=1}^{k} w_i \Phi^{-1}(p_i(s)) \right\} \\ &= \Pr\left\{ Z \leq \frac{\sum_{i=1}^{k} w_i \Phi^{-1}(p_i(s))}{\sqrt{\sum_{i=1}^{n} w_i^2}} \right\} \\ &= \Phi\left\{ \frac{\sum_{i=1}^{k} w_i \Phi^{-1}(p_i(s))}{\sqrt{\sum_{i=1}^{n} w_i^2}} \right\}. \end{aligned}$$

This is the method of Liu et al. (2014) under the general CD framework. This method provides effect size estimates on the log odds ratio scale and is programmed in the *gmeta* R package (Yang et al., 2016a) under the general CD framework.

4.2.4 Combining p–values as in Fisher (1932)

Given the hypotheses $H_0 : \theta \leq s_0$ vs $H_A : \theta > s_0$ for some fixed s_0, Fisher (1932) proposes the following p–value combination method

$$p^{(c)} = \Pr\left\{ \chi^2_{2k} \geq -2 \sum_{i=1}^{k} \log(p_i) \right\}, \tag{4.5}$$

where p_i is the one–sided p–value from the i^{th} study.

If $p_i(s)$ is the p–value function for the i^{th} study, then $p_i = p_i(s_0)$ is the p–value of the of the hypotheses for Fisher's p–value combination method. We can think of combining

p–values as a special case of combining p–value functions. Now, if we combine these individual p–value functions under the general CD framework with $a_0(\cdot) = \log(\cdot)$ and $w_i = 1$, we obtain the following combined CD

$$\begin{aligned}
H^{(c)}(s) &= \Pr\left\{g_c\left(U_1, \ldots, U_k\right) \leq g_c\left(H_1(s), \ldots, H_k(s)\right)\right\} \\
&= \Pr\left\{w_1 a_0(u_1) + \cdots + w_k a_0(u_k) \leq w_1 a_0(p_1(s)) + \cdots + w_k a_0(p_k(s))\right\} \\
&= \Pr\{\log(u_1) + \cdots + \log(u_k) \leq \log(p_1(s)) + \cdots + \log(p_k(s))\} \\
&= \Pr\left\{ -2\sum_{i=1}^{k} \log(u_i) \geq -2\sum_{i=1}^{k} \log(p_i(s)) \right\} \\
&= \Pr\left\{ \chi^2_{2k} \geq -2\sum_{i=1}^{k} \log(p_i(s)) \right\}, \qquad (4.6)
\end{aligned}$$

since $U_1, \ldots, U_k \overset{iid}{\sim} U[0,1] \Rightarrow -2\sum_{i=1}^{k} \log(u_i) \sim \chi^2_{2k}$. This is in the same form as Fisher's p–value combination method, with the alteration of letting s vary instead of being held fixed. Thus, the $p^{(c)}$ in Equation 4.5 equals $H^{(c)}(s_0)$ in Equation 4.6 for the fixed value s_0. As such, we can consider this method to be under the general CD framework of combining p–value functions. Fisher's p–value combination method is also incorporated in the *gmeta* R package. Throughout this chapter, we obtain the individual p–values via asymptotics using a 0.5 continuity correction when needed.

4.3 Innovation

We now consider three alternative approaches to the methods described above. These modifications are aimed to improve these methods when conducting a meta–analysis with rare events and heterogeneity. When applying these three methods, we noticed the method of Tian et al. (2009) (as implemented in the *exactmeta* package) was extremely slow computationally, with the method of Liu et al. (2014) (as implemented in the *gmeta* package) faster by about an order of magnitude, and the method of Fisher (1932) (as

implemented in the *gmeta* package) was very fast. Additionally, we believe the method of Tian et al. (2009) to be more reasonable in the rare event setting since they use the logit function, as opposed to the normal distribution as used in Liu et al. (2014). Accordingly, we make the following modifications.

4.3.1 Modification 1: Use the Logit Function Under the General CD Framework without Weights

Our first modification is to use the logit function under the general CD framework without weighting the studies. We begin by excluding weights based on the argument described in Chapter 1 of Bhaumik et al. (2012), Shuster et al. (2012), and Stijnen et al. (2015) that weights introduce bias when events are rare. This proposed method is akin to the method outlined in Section 4.2.2 (with the exception of not using weights for the studies), but this method is not currently programmed under the general CD framework with the logit transformation function. For this modification, we utilize the *gmeta* code for the method of Liu et al. (2014), which is programmed under the general CD framework, but we use the logit function and no weights. Since the method of Liu et al. (2014) runs remarkably faster than the method of Tian et al. (2009), we would like to use the logit function under this much faster general CD framework to decrease computation time.

Thus, this method under the general CD framework is written as

$$
\begin{aligned}
H^{(c)}(s) &= \Pr\left\{g_c\left(U_1,\ldots,U_k\right) \le g_c\left(H_1(s),\ldots,H_k(s)\right)\right\} \\
&= \Pr\left\{w_1 a_0(u_1) + \cdots + w_k a_0(u_k) \le w_1 a_0(p_1(s)) + \cdots + w_k a_0(p_k(s))\right\} \\
&= \Pr\Bigg\{\log\left(\frac{u_1}{1-u_1}\right) + \cdots + \log\left(\frac{u_k}{1-u_k}\right) \\
&\qquad\qquad \le \log\left(\frac{H_1(s)}{1-H_1(s)}\right) + \cdots + \log\left(\frac{H_k(s)}{1-H_k(s)}\right)\Bigg\} \\
&= \Pr\left\{\sum_{i=1}^{k}\log\left(\frac{u_i}{1-u_i}\right) \le \sum_{i=1}^{k}\log\left(\frac{H_i(s)}{1-H_i(s)}\right)\right\} \\
&= \Pr\left\{-\frac{1}{\pi\sqrt{\frac{k(5k+2)}{3(5k+4)}}}\sum_{i=1}^{k}\log\left(\frac{u_i}{1-u_i}\right) \ge -\frac{1}{\pi\sqrt{\frac{k(5k+2)}{3(5k+4)}}}\sum_{i=1}^{k}\log\left(\frac{H_i(s)}{1-H_i(s)}\right)\right\} \\
&= \Pr\left\{-\frac{1}{\pi\sqrt{\frac{k(5k+2)}{3(5k+4)}}}\sum_{i=1}^{k}\log\left(\frac{H_i(s)}{1-H_i(s)}\right) \le t_\lambda\right\},
\end{aligned}
$$

based on the results from George and Mudholkar (1977), and where $\lambda = 5k + 4$.

4.3.2 Modification 2: Use the Logit Function Under the General CD Framework with Weights Defined by Liu et al. (2014)

Our second modification is to use the logit function under the general CD framework, as described in Section 4.3.1, but this time we include weights as defined by Liu et al. (2014). This model, under the general CD framework, is written exactly as in Section 4.2.2 where the weights are defined differently. Our innovation involves coding this method under the general CD framework of the *gmeta* package.

4.3.3 Modification 3: Use the Tian et al. (2009) Framework with Weights Defined by

Liu et al. (2014)

Our third modification is to use the Tian et al. (2009) method with the weights presented by Liu et al. (2014). This method uses the logit function, which we believe is desirable, while also utilizing more sophisticated weights, as opposed to weighting based on the sample size of the study as in the method of Tian et al. (2009). We modified the *gmeta* code for this modification instead of using the *exactmeta* code because the *gmeta* code was already programmed under the general CD framework. This method, under the general CD framework, is written as in Section 4.2.2 with the w_i defined as in Section 4.2.3.

4.3.4 Summary

As a summary, Table 4.2 presents the six methods discussed above and their characteristics under the general CD framework.

4.4 Application

The six methods discussed will first be compared via a simulation study. Then, they will be applied to a real data set involving cerebral microbleeds. See Appendix E for the relevant code.

4.4.1 Simulation Results

For our simulation study, we used the same framework of the simulation study described in Chapter 2 Section 2.4.1. We use an $\alpha = 0.05$ significance level, and we let θ,

TABLE 4.2: Summary of the six methods discussed in this chapter.

	Tian et al. (2009)	Liu et al. (2014)	Fisher (1932)	Innovation Modification 1	Innovation Modification 2	Innovation Modification 3
s	risk difference	log odds ratio	log odds ratio	log odds ratio	log odds ratio	risk difference
$H_i(s)$	combined confidence intervals using an exact method	combined p–value functions using Fisher's exact test (mid–p)	combined p–values obtained using asymptotics	combined p–value functions using Fisher's exact test (mid–p)	combined p–value functions using Fisher's exact test (mid–p)	combined confidence intervals using an exact method
$a_0(\cdot)$	logit	inverse Normal CDF	log	logit	logit	logit
w_i	based on sample size	based on event rates	none	none	based on event rates	based on event rates

the true treatment effect, equal 0. We let $\tau^2 = 0$, 0.2, 0.4 and 0.8, and we set $\mu = -5$. This will simulate a situation with very rare events, with the baseline probability of an event of 0.7%. Choosing a small event rate is partly due to computational reasons as these methods take very long to compute – especially as the number of events increase. We generated 10 studies for simulation, with the sample sizes for the 10 studies independently generated to be between 50 and 200. We simulated 2,000 replications for each value of τ^2, where data sets were not included if they had zero events across all studies for either the treatment group or the control group for the same reasons mentioned in Chapter 2.

As a baseline, Figure 4.6 shows how four traditional methods (inverse variance, DerSimonian and Laird, Mantel–Haenszel, and Peto, all described in Chapter 1) perform under these settings. These results are analogous to the results in Chapter 2, but are provided here for ease of comparison. Each of these methods has very high type I error rates – especially as the amount of heterogeneity increases. Figure 4.7 shows how the three CD methods perform relative to the traditional methods. The CD methods perform similarly to the inverse variance and DerSimonian and Laird methods, but the CD method of Tian et al. (2009) does have a slightly lower type I error rate. The method of Liu et al. (2014) does not perform any better than the traditional methods, which is presumably due to them using the Normal distribution as the transformation function.

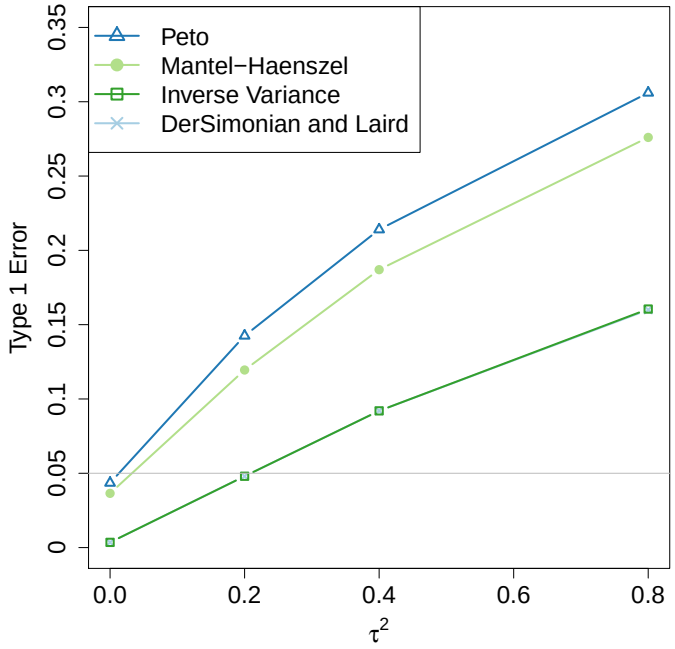

FIGURE 4.6: Simulation results of four common methods under the rare event meta–analysis setting. The grey horizontal line indicates the nominal significance level of 0.05.

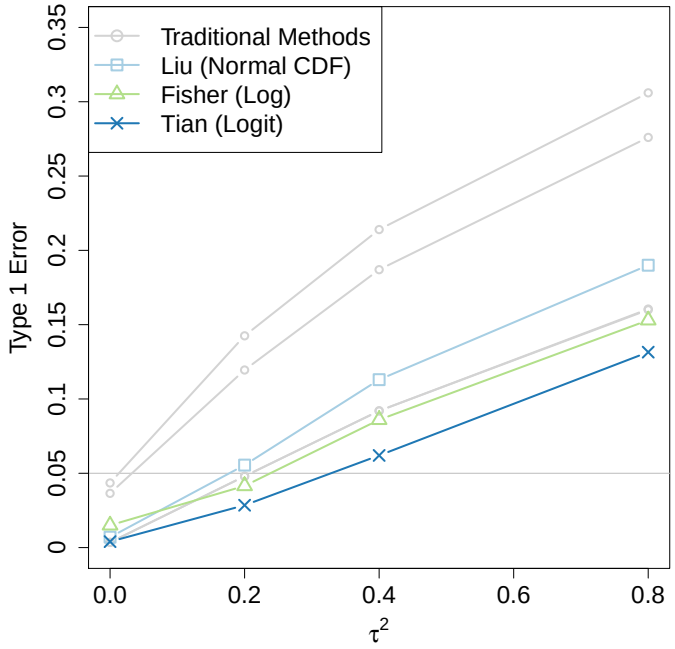

FIGURE 4.7: Simulation results of the three CD methods under the rare event meta–analysis setting with the four traditional methods grayed out in the background for reference. The grey horizontal line indicates the nominal significance level of 0.05.

We then applied our three modifications to this setting, and the results are shown in Figure 4.8. The first two of our modifications perform similarly to the existing CD methods. These two methods were programmed under the general CD framework, and both use the logit as the transformation function. Our third modification, however, has the lowest type I error rate out of all the methods. This modification is under the framework of Tian et al. (2009) using the more sophisticated weights of Liu et al. (2014). The logit transformation function is used, which is fitting for binary outcomes. This modification brings down the type I error rate to acceptable levels when $\tau^2 = 0$, 0.2, and 0.4, and is under 10% when $\tau^2 = 0.8$.

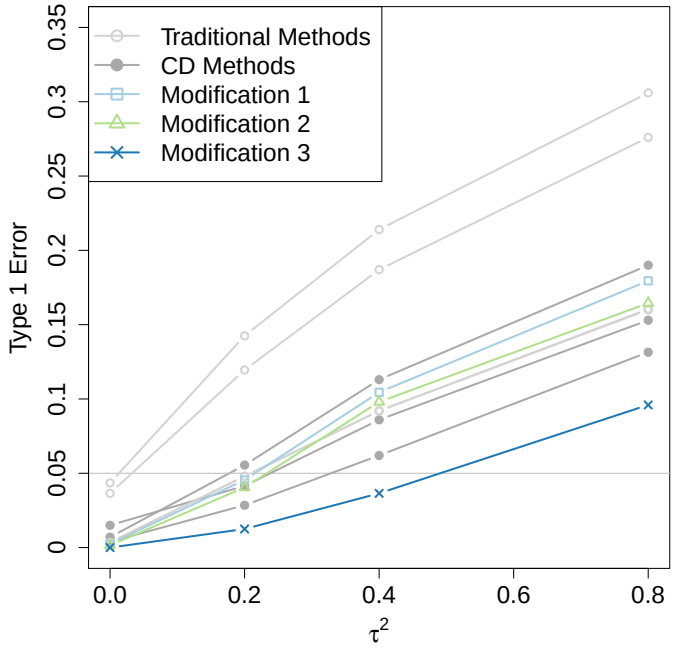

FIGURE 4.8: Simulation results of the three new modifications to the CD methods under the rare event meta–analysis setting with the four traditional methods and the three CD methods grayed out in the background for reference. The grey horizontal line indicates the nominal significance level of 0.05.

4.4.2 Example Data Set: Cerebral Microbleeds

This example uses the cerebral microbleed (CMB) data set, which was analyzed using a meta–analysis in Tsivgoulis et al. (2016). CMBs, as visualized on an MRI, are indicators of future bleeding cerebral vessels. Previous researchers have found that the presence of CMBs are associated with hemorrhagic stroke and hemorrhagic complications following antithrombotic medications, which reduce the formation of blood clots (Kim and Lee, 2013). Given that association, Tsivgoulis et al. (2016) were interested in the association between CMBs and the risk of symptomatic intracerebral hemorrhage (sICH) in patients with acute ischemic stroke treated with intravenous thrombolysis. Patients with AIS experience a sudden loss of blood circulation to part of the brain and are often treated with intravenous thrombolysis. sICH is “the most feared complication” of intravenous

thrombolysis treatment and has a mortality rate of almost 50% (Yaghi et al., 2014). Tsivgoulis et al. (2016) conducted a meta–analysis to determine the risk of sICH when CMBs are present in patients using intravenous thrombolysis to treat AIS. The nine studies in this data set are shown in Table 4.3. The average sICH rate in patients with CMBs present is 9.3%, and the average sICH rate in patients without CMBs present is 3.5%, so sICH is a fairly rare event in these studies.

TABLE 4.3: CMB data set. Source: Tsivgoulis et al. (2016)

Study	CMB Present		CMB Absent	
	Event	Total	Event	Total
1 Dannenberg et al.	7	81	3	245
2 Derex et al.	1	8	2	36
3 Fiehler et al.	5	86	13	484
4 Goyal et al.	1	3	0	18
5 Gratz et al.	2	38	4	136
6 Kakuda et al.	0	11	5	59
7 Kimura et al.	4	72	2	152
8 Turc et al.	12	150	52	567
9 Yan et al.	6	132	2	201

Figure 4.9 shows a "forest" plot for the nine studies plotted as confidence curves. Study 1 is the only study suggesting a significant effect of CMBs on sICH risk, while all other studies indicate no significant effect of CMBs on sICH risk at the 0.05 significance level. The 9th study also may suggest a detrimental effect of the presence of CMBs on sICH, although the effect is not quite statistically significant. Since Studies 4 and 6 have a zero event, the confidence curves do not look like the confidence curves of the other studies; for more information on this see Liu et al. (2014). Additionally, from Figure 4.9,

we see that Studies 3 and 8 are the most precise out of the nine studies.

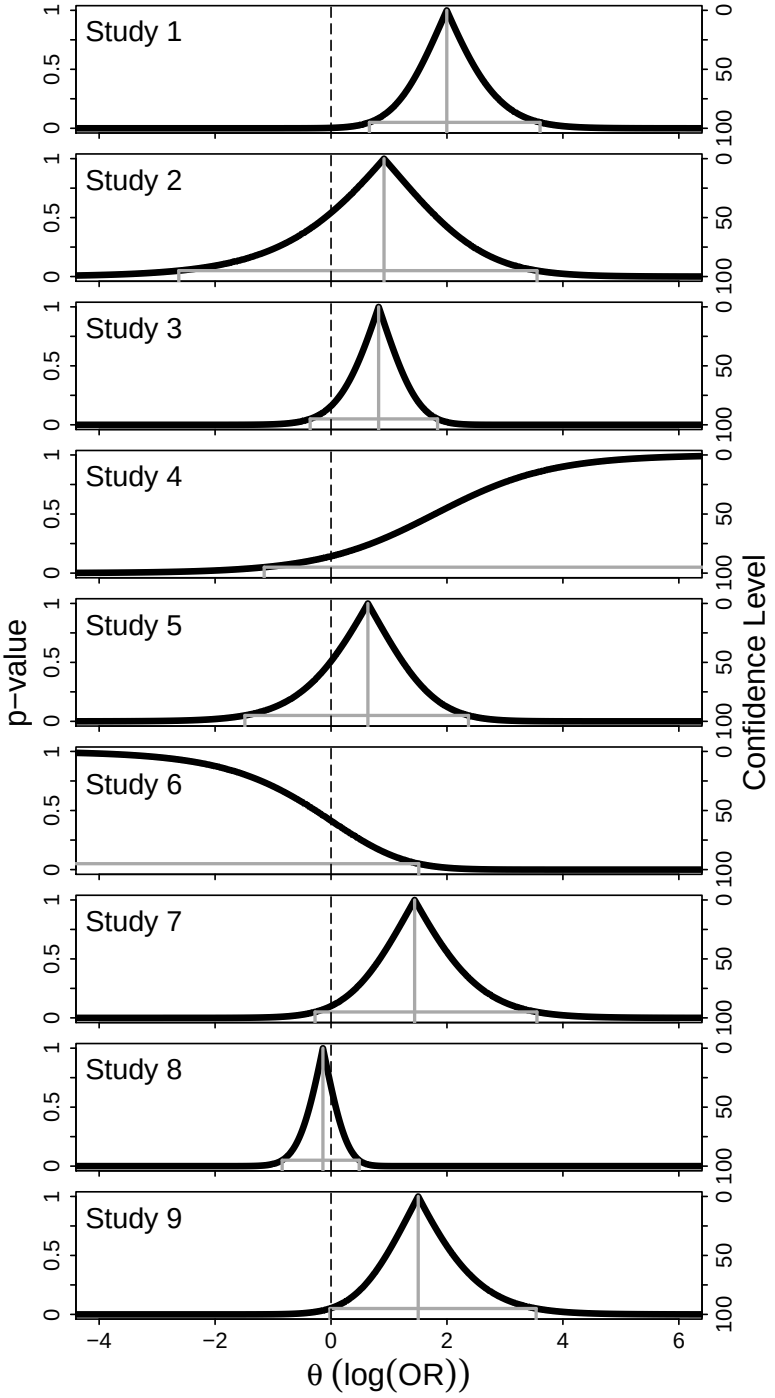

FIGURE 4.9: "Forest" plot of the nine CMB studies plotted as confidence curves. The solid grey vertical lines on each plot indicate the point estimate (median), and the other solid grey lines mark the 95% confidence interval for each study. The black vertical dashed line marks the null value of 0.

Table 4.4 shows the results after combining the nine CMB studies using the four traditional methods. Since there were two studies with zero events (studies 4 and 6), a 0.5 continuity correction was added to all cells in those studies for the inverse variance and DerSimonian and Laird methods in order to include all nine studies in the analysis. All nine methods are unanimous, and one would conclude the risk of sICH may be greater when CMBs are present. This is what Tsivgoulis et al. (2016) concluded, as well. Note that the DerSimonian and Laird and inverse variance approaches produce different results. This is because the between–study heterogeneity was estimated to be non–zero ($\tau^2 = 0.45$).

TABLE 4.4: Meta–analysis results from four traditional methods for the CMB data set.

	$\hat{\theta}$	95% CI	p–value
Inverse Variance (0.5 Correction)	Odds Ratio 1.80	$(1.16, 2.80)$	0.009 (significant)
Mantel–Haenszel	Odds Ratio 1.71	$(1.14, 2.57)$	0.010 (significant)
Peto	Odds Ratio 1.83	$(1.17, 2.87)$	0.008 (significant)
DerSimonian and Laird (0.5 Correction)	Odds Ratio 2.46	$(1.22, 4.97)$	0.012 (significant)

Table 4.5 shows the results from the three CD methods. The Tian et al. (2009) results came from the *exactmeta* R package using the "inverse–variance" weighting method. The methods from Liu et al. (2014) and Fisher (1932) both agree with the traditional methods, indicating the presence of CMBs may increase the risk of sICH. The method of Tian et al. (2009), however, disagrees with these methods and suggests the risk

of sICH may not increase with the presence of CMBs. The CD approaches provide conflicting results as to the effect of the presence of CMBs.

TABLE 4.5: Meta–analysis results from the three CD methods for the CMB data set.

	$\hat{\theta}$	95% CI	p–value
Tian et al. (2009)	Risk Difference 0.03	$(-0.0002, 0.0576)$	0.054 (non–significant)
Liu et al. (2014)	Odds Ratio 1.73	$(1.12, 2.62)$	0.014 (significant)
Fisher (1932)	-	-	0.0002 (significant)

We now apply our three new modifications to the CMB data set, and the results are shown in Table 4.6. Modifications 1 and 2, which were shown to have type I error rates similar to the three CD methods from our simulations, suggest the presence of CMBs may increase the risk of sICH. However, Modification 3, which had the lowest type I error rate of all the methods considered in our simulations, suggests the risk of sICH does not increase when CMBs are present. Due to the results shown in Figure 4.8, it is possible that the results of the traditional methods and the two CD methods of Liu et al. (2014) and Fisher (1932) could be type I errors, and our new CD method, Modification 3, more accurately reflects the true effect of the presence of CMBs on sICH.

TABLE 4.6: Meta–analysis results from the new modifications of the CD methods for the CMB data set.

	$\hat{\theta}$	95% CI	p–value
Modification 1	Odds Ratio 1.69	$(1.16, 2.33)$	0.010 (significant)
Modification 2	Odds Ratio 2.11	$(1.37, 3.07)$	0.001 (significant)
Modification 3	Risk Difference 0.01	$(-0.008, 0.054)$	- (non–significant)

4.5 Conclusion

CDs are a rich, unifying tool that have recently been used in the meta–analysis setting. They provide information on the distribution of the effect size, and they have been shown to work well in most settings. However, in the rare event meta–analysis setting when heterogeneity is present, the common CD methods perform rather poorly, with unacceptably large type I error rates. This chapter focused on improving those existing methods to perform better when events are rare and there is a non–negligible amount of heterogeneity. We showed that our Modification 3 outperforms the existing CD methods in terms of lowering the type I error to a more acceptable rate. Given that more than half of the reviews from a random sample in 2009 contained rare events (Vandermeer et al., 2009), it is critical to develop methods that can analyze these data properly. The cerebral microbleeds example of Section 4.4.2 suggests a troublesome thought that many of the analyses conducted and resulting conclusions could be results of type I errors, if heterogeneity is present, and patients could be receiving non–beneficial treatments. We hope our modification will be considered and more methods will continue to evolve to better analyze meta–analyses with rare events in the presence of heterogeneity.

CHAPTER 5
CONCLUSION

Meta-analyses have become increasingly popular to conduct, especially in public health and medicine, where they are the ideal form of "evidence–based medicine." Often, medical data consists of rare outcomes or small sample sizes where conventional methods, which rely on large–sample approximations, fail. Recently, methods have been developed that provided options for sparse data, but many of these methods are hampered when heterogeneity across the available studies differs according to the treatment group.

We have developed an exact permutation–based method that performs well when events are rare, and non–negligible heterogeneity is present. Our exact method produces much lower type I errors and has better confidence interval coverage probabilities than commonly used methods. In the future, we would like to extend our simulation studies to cover broader values of each parameter and to include some newer methods in our comparisons. Additionally, we are considering implementing a sensitivity analysis to our method to identify influential studies.

We have also extended this approach to the novel method of network meta–analysis. While our method did not perform better than the DerSimonian and Laird method with a simulated average event rate of 10%, we plan to apply Monte Carlo sampling, which would make a smaller event rate feasible. Exact tests generally provide the greatest advantage when samples are small or sparse, so further simulation studies with smaller event rates would better compare these two methods. Additionally, we plan to investigate alternative indirect estimation methods. We also plan to extend this method to work with more complex network meta–analysis structures by incorporating multiple correlation structures in our exact method.

We also utilized confidence distributions, which have received increasing attention lately in their application to meta–analysis. Currently, several different confidence distributions are available, and it is important to identify which method is best suited for rare events and heterogeneity. We compared the performance of these methods under the rare event and heterogeneity setting, and we found that these methods produce inflated type I error rates with non–negligible heterogeneity. Accordingly, we made several modifications to these methods to better handle situations with rare events or heterogeneity.

Upon further investigation, we discovered that our method, which produced the lowest type I error of all considered methods, actually performed very similarly to the method of Tian et al. (2009) as implemented in the *gmeta* R package, as opposed to the *exactmeta* R package on which our simulations were based. It is very interesting that Tian et al. (2009)'s method produces lower type I error rates when using an R package not developed by them. Further study is needed to determine why the *gmeta* package systematically produces lower type I error rates, especially since the developers of the *gmeta* package show their implementation to be identical to the implementation programmed in the *exactmeta* package (Yang et al., 2016b). In the future, we plan to create a confidence distribution using our exact permutation–based method, which we think may rival all confidence distribution methods considered.

With the plethora of data in the world today, meta–analyses are often employed to pool the wealth of information available to facilitate decision making. We have witnessed a troubling trend that suggests many of the commonly used methods for meta–analyses of rare event data with heterogeneity produce results that are likely type I errors. In these cases, treatments may be provided to patients, perhaps at great cost, that are not actually beneficial. In this light, we greatly hope our developments will assist researchers in better analyzing meta–analyses with rare events in the presence of non–negligible heterogeneity.

REFERENCES

Akobeng, A. (2005), "Understanding randomised controlled trials," *Archives of Disease in Childhood*, 90, 840–844.

Bhaumik, D. K., Amatya, A., Normand, S., Greenhouse, J., Kaizar, E., Neelon, B., and Gibbons, R. D. (2012), "Meta–Analysis of Rare Binary Adverse Event Data," *Journal of the American Statistical Association*, 107, 498, 555–567.

Böhning, D., Mylona, K., and Kimber, A. (2015), "Meta–analysis of clinical trials with rare events," *Biometrical Journal*, 57, 4, 633–648.

Bradburn, M. J., Deeks, J. J., Berlin, J. A., and Localio, A. R. (2007), "Much ado about nothing: a comparison of the performance of meta–analytical methods with rare events," *Statistics in Medicine*, 26, 53–77.

Bucher, H. C., Guyatt, G. H., Griffith, L. E., and Walter, S. D. (1997), "The Results of Direct and Indirect Treatment Comparisons in Meta–Analysis of Randomized Controlled Trials," *Journal of Clinical Epidemiology*, 50, 6, 683–691.

Cai, T., Parast, L., and Ryan, L. (2010), "Meta–analysis for rare events," *Statistics in Medicine*, 29, 2078–2089.

Cochran, W. G. (1954), "The Combination of Estimates from Different Experiments," *Biometrics*, 10, 1, 101–129.

Cochrane (2019), "Our vision, mission, and principles," Available at http://www.cochrane.org/about-us/our-vision-mission-and-principles.

Coleman, C. I., Phung, O. J., Cappelleri, J. C., Baker, W. L., Kluger, J., White, C. M., and Sobieraj, D. M. (2012), "Use of Mixed Treatment Comparisons in Systematic Reviews," *US Agency for Healthcare Research and Quality*.

Corcoran, C., Ryan, L., Senchaudhuri, P., Mehta, C., Patel, N., and Molenberghs, G. (2001), "An Exact Trend Test for Correlated Binary Data," *Biometrics*, 57, 941–948.

DerSimonian, R. and Laird, N. (1986), "Meta–Analysis in Clinical Trials," *Controlled Clinical Trials*, 7, 177–188.

Efron, B. (1993), "Bayes and likelihood calculations from confidence intervals," *Biometrika*, 80, 3–26.

——— (1996), "Empirical Bayes Methods for Combining Likelihoods," *Journal of the American Statistical Association*, 91, 434, 538–550.

Efthimiou, O. (2018), "Practical guide to the meta–analysis of rare events," *Evid Based Mental Health*, 21, 2, 72–76.

Fisher, R. A. (1932), *Statistical Methods for Research Workers*, 4 ed.

——— (1973), *Statistical Methods and Scientific Inference*, 3 ed.

Fraser, D. A. S. (1991), "Statistical Inference: Likelihood to Significance," *Journal of the American Statistical Association*, 86, 414, 258–265.

Friedrich, J. O., Adhikari, N. K. J., and Beyene, J. (2007), "Inclusion of zero total event trials in meta–analyses maintains analytic consistency and incorporates all available data," *BMC Medical Research Methodology*, 7, 5.

George, E. O. and Mudholkar, G. S. (1977), "The logit method for combining independent tests," *Institute of Mathematical Statistics Bulletin*, 6, 212.

Glenny, A., Altman, D. G., Song, F., Sakarovitch, C., Deeks, J., D'Amico, R., Bradburn, M., and Eastwood, A. (2015), "Indirect comparisons of competing interventions," *Health Technology Assessment*, 9, 26.

Greenland, S. and Salvan, A. (1990), "Bias in the one–step method for pooling study results," *Statistics in Medicine*, 9, 3, 247–252.

Higgins, J. and Green, S. (2011), Wiley Publication, Chichester, UK.

Higgins, J. P. T. and Thompson, S. G. (2002), "Quantifying heterogeneity in a meta–analysis," *Statistics in Medicine*, 21, 1539–1558.

Huedo-Medina, T., Sanchez-Meca, J., Marin-Martinez, F., and Botella, J. (2006), "Assessing heterogeneity in meta–analysis: Q statistic or I2 index?" *CHIP Documents*, 19.

Hunter, J. E. and Schmidt, F. L. (2000), "Fixed Effects vs. Random Effects Meta–Analysis Models: Implications for Cumulative Research Knowledge," *International Journal of Selection and Assessment*, 8, 4, 275–292.

Ioannidis, J. P. (2006), "Meta–analysis in public health: potentials and problems," *Italian Journal of Public Health*, 3, 2, 9–14.

Jansen, J. P., Fleurence, R., Devine, B., Itzler, R., Barrett, A., Hawkins, N., Lee, K., Boersma, C., Annemans, L., and Cappelleri, J. C. (2011), "Interpreting Indirect Treatment Comparisons and Network Meta–Analysis for Health–Care Decision Making: Report for the ISPOR Task Force on Indirect Treatment Comparisons Good Research Practices: Part 1," *Value in Health*, 14, 417–428.

Kim, B. J. and Lee, S. H. (2013), "Cerebral microbleeds: their associated factors, radiologic findings, and clinical implications," *J Stroke*, 15, 3, 153–163.

Kim, H., Gurrin, L., and Ademi, Z. (2013), "Overview of methods for comparing the efficacies of drugs in the absence of head–to–head clinical trial data," *British Journal of*

Clinical Pharmacology, 77, 1, 116–121.

Kim, M., Wang, X., Liu, C., Dorris, K., Fouladi, M., and Song, S. (2016), "Random–effects meta–analysis for systematic reviews of phase I clinical trials: Rare events and missing data," *Research Synthesis Methods*.

Kuss, O. (2015), "Statistical methods for meta–analyses including information from studies without any events – add nothing to nothing and succeed nevertheless," *Statistics in Medicine*, 34, 1097–1116.

Liu, D., Liu, R. Y., and Xie, M. (2014), "Exact Meta–Analysis Approach for Discrete Data and its Application to 2 x 2 Tables With Rare Events," *Journal of the American Statistical Association*, 109, 508, 1450–1465.

Lu, G. and Ades, A. E. (2004), "Combination of direct and indirect evidence in mixed treatment comparisons," *Statistics in Medicine*, 23, 3105–3124.

Lumley, T. (2002), "Network meta–analysis for indirect treatment comparisons," *Statistics in Medicine*, 21, 2313–2324.

Mantel, N. and Haenszel, W. (1959), "Statistical Aspects of the Analysis of Data From Retrospective Studies of Disease," *J Natl Cancer Inst*, 22, 4, 719–748.

Mehta, C. R., Patel, N., and Senchaudhuri, P. (1992), "Exact Stratified Linear Rank Tests for Ordered Categorical and Binary Data," *Journal of Computational and Graphical Statistics*, 1, 1, 21–40.

Mehta, C. R. and Patel, N. R. (1995), "Exact Logistic Regression: Theory and Examples," *Statistics in Medicine*, 14, 2143–2160.

Mehta, C. R., Patel, N. R., and Senchaudhuri, P. (1988), "Importance Sampling for Estimating Exact Probabilities in Permutational Inference," *Journal of the American Statistical Association*, 83, 404, 999–1005.

Mills, E. J., Ghement, I., O'Regan, C., and Thorlund, K. (2011), "Estimating the Power of Indirect Comparisons: A Simulation Study," *PLoS One*, 6:e16237.

Mills, E. J., Thorlund, K., and Ioannidis, J. P. A. (2013), "Demystifying trial networks and network meta–analysis," *BMJ*, 346:f2914.

Molenberghs, G. and Ryan, L. M. (1999), "An Exponential Family Model for Clustered Multivariate Binary Data," *Environmentrics*, 10, 279–300.

Moreno, S. G., Sutton, A. J., Thompson, J. R., Ades, A. E., Abrams, K. R., and Cooper, N. J. (2012), "A generalized weighting regression–derived meta-analysis estimator robust to small–study effects and heterogeneity," *Statistics in Medicine*, 31, 1407–1417.

O'Rourke, K. (2007), "An historical perspective on meta–analysis: dealing quantitatively with varying study results," *Journal of the Royal Society of Medicine*, 100, 579–582.

Paul, L. M. (2018), "Cannons and sparrows: an exact maximum likelihood non–parametric test for meta–analysis of k 2×2 tables," *Emerging Themes in Epidemiology*, 15, 9.

Paule, R. C. and Mandel, J. (1982), "Consensus Values and Weighting Factors," *Journal of Research of the National Bureau of Standards*, 87, 5, 377–385.

R Core Team (2014), *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria.

Sankey, S. S., Weissfeld, L. A., Fine, M. J., and Kapoor, W. (1996), "An assessment of the use of the continuity correction for sparse data in meta–analysis," *Communications in Statistics – Simulation and Computation*, 25, 1031–1056.

Shuster, J. J., Guo, J. D., and Skyler, J. S. (2012), "Meta–analysis of safety for low event–rate binomial trials," *Research Synthesis Methods*, 3, 30–50.

Shuster, J. J. and Walker, M. A. (2016), "Low–event–rate meta–analyses of clinical trials: implementing good practices," *Statistics in Medicine*, 35, 2467–2478.

Sidik, K. and Jonkman, J. N. (2005), "Simple heterogeneity variance estimation for meta–analysis," *Journal of Royal Statistical Society (Series C)*, 54, 367–384.

——— (2007), "A comparison of heterogeneity variance estimators in combining results of studies," *Statistics in Medicine*, 26, 1964–1981.

Singh, K., Xie, M., and Strawderman, W. E. (2005), "Combining Information from Independent Sources through Confidence Distributions," *The Annals of Statistics*, 33, 1, 159–183.

Spinks, A., Glasziou, P., and Del Mar, C. B. (2013), "Antibiotics for sore throat (Review)," *Cochrane Database of Systematic Reviews*, 11.

Stijnen, T., Hamza, T. H., and Özdemir, P. (2015), "Random effects meta–analysis of event outcome in the framework of the generalized linear mixed model with applications in sparse data," *Statistics in Medicine*, 34, 1097–1116.

Sutton, A. J. and Higgins, J. P. T. (2008), "Recent developments in meta–analysis," *Statistics in Medicine*, 27, 625–650.

Sweeting, M. J., Sutton, A. J., and Lambert, P. C. (2004), "What to add to nothing? Use and avoidance of continuity corrections in meta-analysis of sparse data," *Statistics in Medicine*, 23, 1351–1357.

T. C. Smith, A. T., D. J. Spiegelhalter (1995), "Bayesian approaches to random–effects meta–analysis: a comparative study," *Statistics in Medicine*, 14, 2685–2699.

Thorlund, K. and Mills, E. J. (2012), "Sample size and power considerations in network meta–analysis," *Systematic Reviews*, 1:41.

Tian, L., Cai, T., Pfeffer, M. A., Piankov, N., Cremieux, P.-y., and Wei, L. J. (2009), "Exact and efficient inference procedure for meta–analysis and its application to the analysis of independent 2 x 2 tables with all available data but without artificial continuity correction," *Biostatistics*, 10, 2, 275–281.

Tonin, F. S., Rotta, I., Mendes, A. M., and Pontarolo, R. (2017), "Network meta–analysis: a technique to gather evidence from direct and indirect comparisons," *Pharmacy Practice*, 15, 1, 943.

Tsivgoulis, G., Zand, R., Katsanos, A. H., Turc, G., Nolte, C. H., Jung, S., Cordonnier, C., Fiebach, J. B., Scheitz, J. F., Klinger-Gratz, P. P., Oppenheim, C., Goyal, N., Safouris, A., Mattle, H. P., Alexandrov, A. W., Schellinger, P. D., and Alexandrov, A. V. (2016), "Risk of Symptomatic Intracerebral Hemorrhage After Intravenous Thrombolysis in Patients With Acute Ischemic Stroke and High Cerebral Microbleed Burden: A Meta–analysis," *JAMA Neurol*, 73, 6, 675–683.

Vandermeer, B., Bialy, L., Hooton, N., Hartling, L., Klassen, T. P., Johnston, B. C., and Wiebe, N. (2009), "Meta–analysis of safety data: a comparison of exact versus asymptotic methods," *Statistical Methods in Medical Research*, 18, 421–432.

Whitehead, A. and Whitehead, J. (1991), "A general parametric approach to the meta–analysis of sparse data," *Statistics in Medicine*, 10, 1665–1677.

Xie, M., Singh, K., and Strawderman, W. E. (2011), "Confidence Distributions and a Unifying Framework for Meta–Analysis," *Journal of American Statistical Association*, 106, 493, 320–333.

Yaghi, S., Eisenberger, A., and Willey, J. Z. (2014), "Symptomatic Intracerebral Hemorrhage in Acute Ischemic Stroke After Thrombolysis With Intravenous Recombinant Tissue Plasminogen Activator: A Review of Natural History and Treatment," *JAMA Neurol*, 71, 9, 1181–1185.

Yang, G., Cheng, J. Q., and Xie, M. (2016a), *gmeta: Meta-Analysis via a Unified Framework of Confidence Distribution*, r package version 2.2-6.

Yang, G., Liu, D., Wang, J., and Xie, M. (2016b), "Meta–Analysis Framework for Exact Inferences with Application to the Analysis of Rare Events," *Biometrics*, 72, 1378–1386.

Yusuf, S., Peto, R., Lewis, J., Collins, R., and Sleight, P. (1985), "Beta Blockade During and After Myocardial Infarction: An Overview of the Randomized Trials," *Progress in Cardiovascular Diseases*, 27, 5, 335–371.

APPENDICES

Appendix A: Information on the Confidence Interval Bounds of Section 2.2.3

There are several different methods of finding confidence intervals. For our exact estimation method (described in Section 2.2.3), we utilize the method known as "inverting a test statistic." This method capitalizes on the strong relationship between hypothesis testing and interval estimation. Hypothesis tests hold the parameter fixed and determine what sample values are consistent with that fixed parameter. Confidence intervals hold the sample value fixed and determine what values of the parameter makes that sample value most likely. To obtain a $(1-\alpha)$ confidence interval, a hypothesis test can be inverted, which effectively inverts the acceptance region of the α level hypothesis test. This process could be repeated for each possible value of the parameter in the parameter space. However, to avoid unnecessarily testing each possible value of the parameter in the parameter space, we can start at one end of a distribution and carry out these tests until the probability exceeds some threshold. At that point, we know subsequent values would fall in the acceptance region, and no further testing is needed.

As an example, consider, the one–sided hypothesis test $H_0 : \theta = \theta_0$ and $H_a : \theta > \theta_0$. In this case, to create a $(1-\alpha)$ one–sided confidence interval we use the lower tail of the distribution to find the upper bound of the confidence interval. To avoid unnecessarily testing each possible value of the parameter in the parameter space, we work from the left of the distribution to the right, stopping when we reach the point at which $P(\theta < \theta_0) > 1-\alpha$. Figure A.1 shows what this would look like for a one–sided 95% confidence interval for normally distributed data.

Similarly, for the one–sided hypothesis test $H_0 : \theta = \theta_0$ and $H_a : \theta < \theta_0$, we use the upper tail of the distribution to find the lower bound of the confidence interval. For a $(1-\alpha)$ one–sided confidence interval, we work from the right of the distribution to the left, stopping when we reach the point at which $P(\theta > \theta_0) > 1-\alpha$. Figure A.2 shows what this would look like for a one–sided 95% confidence interval for normally distributed data.

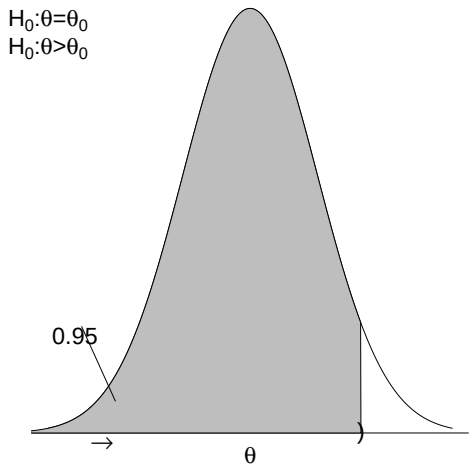

FIGURE A.1: This figure illustrates how to find the one–sided 95% confidence interval given the corresponding one–sided hypothesis test.

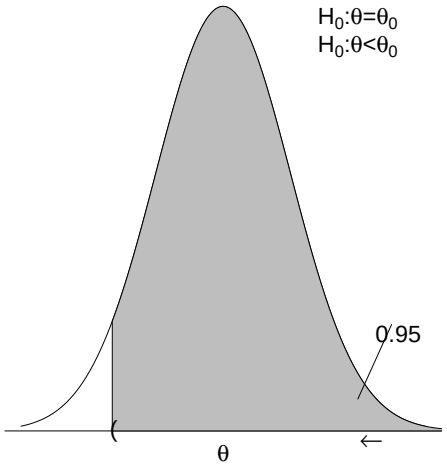

FIGURE A.2: This figure illustrates how to find the one–sided 95% confidence interval given the corresponding one–sided hypothesis test.

We utilize this method when creating our exact confidence intervals. Note that for

two sided intervals, the two one–sided calculations (using $\frac{\alpha}{2}$) are essentially combined to produce lower and upper limits.

Appendix B: Adjustments to the Conditional Probability of Section 2.2.3

The conditional probability shown in Equation 2.10 is used to compute exact estimates and exact confidence intervals. Often, the distribution of test statistics is scaled to start at zero since large values of the test statistic quickly produce values that are too large to be stored in computer memory. Accordingly, the following adjustment can be made to Equation 2.10.

$$
\begin{aligned}
f_\beta(t(\boldsymbol{z}_{obs})|s^\alpha(\boldsymbol{z}), s^\delta(\boldsymbol{z})) &= \frac{C(t(\boldsymbol{z}_{obs})|s^\alpha(\boldsymbol{z}), s^\delta(\boldsymbol{z}))\exp\{\beta t(\boldsymbol{z}_{obs})\}}{\sum_{u=\min(t(\boldsymbol{z}))}^{\max(t(\boldsymbol{z}))} C(u|s^\alpha(\boldsymbol{z}), s^\delta(\boldsymbol{z}))\exp\{\beta u\}} \\
&= \frac{C(t(\boldsymbol{z}_{obs})|s^\alpha(\boldsymbol{z}), s^\delta(\boldsymbol{z}))\exp\{\beta t(\boldsymbol{z}_{obs})\}}{\sum_{u=\min(t(\boldsymbol{z}))}^{\max(t(\boldsymbol{z}))} C(u|s^\alpha(\boldsymbol{z}), s^\delta(\boldsymbol{z}))\exp\{\beta u\}} \times \frac{\exp\{-\beta\min(t(\boldsymbol{z}))\}}{\exp\{-\beta\min(t(\boldsymbol{z}))\}} \\
&= \frac{C(t(\boldsymbol{z}_{obs})|s^\alpha(\boldsymbol{z}), s^\delta(\boldsymbol{z}))\exp\{\beta\left(t(\boldsymbol{z}_{obs}) - \min(t(\boldsymbol{z}))\right)\}}{\sum_{u=\min(t(\boldsymbol{z}))}^{\max(t(\boldsymbol{z}))} C(u|s^\alpha(\boldsymbol{z}), s^\delta(\boldsymbol{z}))\exp\{\beta\left(u - \min(t(\boldsymbol{z}))\right)\}} \\
&= \frac{C(t(\boldsymbol{z}_{obs})|s^\alpha(\boldsymbol{z}), s^\delta(\boldsymbol{z}))\exp\{\beta t(\boldsymbol{z}^{\text{scaled}}_{obs})\}}{\sum_{u^{\text{scaled}}=\min(t(\boldsymbol{z}^{\text{scaled}}))}^{\max(t(\boldsymbol{z}^{\text{scaled}}))} C(u|s^\alpha(\boldsymbol{z}), s^\delta(\boldsymbol{z}))\exp\{\beta u^{\text{scaled}}\}},
\end{aligned}
$$

The unnormalized probabilities $C(\cdot)$ are also often replaced by the normalized probabilities $N(\cdot)$ by dividing the numerator and denominator by the normalizing constant (NC)

$$
\begin{aligned}
f_\beta(t(\boldsymbol{z}_{obs})|s^\alpha(\boldsymbol{z}), s^\delta(\boldsymbol{z})) &= \frac{C(t(\boldsymbol{z}_{obs})|s^\alpha(\boldsymbol{z}), s^\delta(\boldsymbol{z}))\exp\{\beta t(\boldsymbol{z}^{\text{scaled}}_{obs})\}}{\sum_{u^{\text{scaled}}=\min(t(\boldsymbol{z}^{\text{scaled}}))}^{\max(t(\boldsymbol{z}^{\text{scaled}}))} C(u|s^\alpha(\boldsymbol{z}), s^\delta(\boldsymbol{z}))\exp\{\beta u^{\text{scaled}}\}} \times \frac{NC}{NC} \\
&= \frac{\frac{C(t(\boldsymbol{z}_{obs})|s^\alpha(\boldsymbol{z}), s^\delta(\boldsymbol{z}))}{NC}\exp\{\beta t(\boldsymbol{z}^{\text{scaled}}_{obs})\}}{\sum_{u^{\text{scaled}}=\min(t(\boldsymbol{z}^{\text{scaled}}))}^{\max(t(\boldsymbol{z}^{\text{scaled}}))} \frac{C(u|s^\alpha(\boldsymbol{z}), s^\delta(\boldsymbol{z}))}{NC}\exp\{\beta u^{\text{scaled}}\}} \\
&= \frac{N(t(\boldsymbol{z}_{obs})|s^\alpha(\boldsymbol{z}), s^\delta(\boldsymbol{z}))\exp\{\beta t(\boldsymbol{z}^{\text{scaled}}_{obs})\}}{\sum_{u^{\text{scaled}}=\min(t(\boldsymbol{z}^{\text{scaled}}))}^{\max(t(\boldsymbol{z}^{\text{scaled}}))} N(u|s^\alpha(\boldsymbol{z}), s^\delta(\boldsymbol{z}))\exp\{\beta u^{\text{scaled}}\}}.
\end{aligned}
$$

Working with the log of the conditional probability also decreases computational issues that arise with large data sets. Accordingly,

$$\begin{aligned}
\log\left(f_\beta(\cdot)\right) &= \log\left(\frac{N(t(\boldsymbol{z}_{obs})|s^\alpha(\boldsymbol{z}),s^\delta(\boldsymbol{z}))\exp\{\beta t(\boldsymbol{z}_{obs}^{\text{scaled}})\}}{\sum_{u^{\text{scaled}}=\min(t(\boldsymbol{z}^{\text{scaled}}))}^{\max(t(\boldsymbol{z}^{\text{scaled}}))} N(u|s^\alpha(\boldsymbol{z}),s^\delta(\boldsymbol{z}))\exp\{\beta u^{\text{scaled}}\}}\right) \\
&= \log\left(N(t(\boldsymbol{z}_{obs})|s^\alpha(\boldsymbol{z}),s^\delta(\boldsymbol{z}))\exp\{\beta t(\boldsymbol{z}_{obs}^{\text{scaled}})\}\right) \\
&\quad - \log\left(\sum_{u^{\text{scaled}}=\min(t(\boldsymbol{z}^{\text{scaled}}))}^{\max(t(\boldsymbol{z}^{\text{scaled}}))} N(u|s^\alpha(\boldsymbol{z}),s^\delta(\boldsymbol{z}))\exp\{\beta u^{\text{scaled}}\}\right) \\
&= \log\left(N(t(\boldsymbol{z}_{obs}^{s})|s^\alpha(\boldsymbol{z}),s^\delta(\boldsymbol{z}))\right) + \beta t(\boldsymbol{z}_{obs}^{\text{scaled}}) \\
&\quad - \log\left(\sum_{u^{\text{scaled}}=\min(t(\boldsymbol{z}^{\text{scaled}}))}^{\max(t(\boldsymbol{z}^{\text{scaled}}))} N(u|s^\alpha(\boldsymbol{z}),s^\delta(\boldsymbol{z}))\exp\{\beta u^{\text{scaled}}\}\right).
\end{aligned}$$

A common formula for rewriting the log of a summation is

$$\log(x+y) = \log(x) + \log(1+\exp\{\log(y)-\log(x)\}).$$

We can iteratively apply this formula to obtain an exact estimate.

Appendix C: Code for Chapter 2

This appendix contains selected code for Chapter 2, and includes the following:

- clustexamp.c
- trend.c
- numrout.c
- nrutil.h
- ExactEstimation.R
- CMLE.R
- ma.sim.dat.R
- Generate_Data.R

The files clustexamp.c, trend.c, numrout.c, and nrutil.h are used to compute the exact distribution. Once the distribution is computed, an exact estimate and exact confidence interval can be computed with ExactEstimation.R and CMLE.R. For the simulations in Chapter 2, ma.sim.dat.R and Generate_Data.R were used to simulate the meta–analysis data sets. These data sets were then run through the exact distribution C code and the exact estimation R code to obtain exact estimates, confidence intervals, and p–values.

Exact Distribution C Code - clustexamp.c:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <limits.h>
#include "nrutil.h"

// Note: treatment groups must be labeled with 1s (treatment) and 0s (control)
// The meta-analysis data set should be a .txt file.
// The number of studies * 2 is on the first line, with the
// indicator for treatment, the number of events, and the sample size following
// Example data set with NUMCLUST = 10:
/*
10
0 2 4
0 3 7
0 0 2
0 2 6
0 1 7
1 3 4
1 0 2
1 1 4
1 1 6
1 2 5
*/
#define NUMCLUST 10  // number of studies * 2

int main (int argc, char *argv[]) {

                int dose[NUMCLUST], litter[NUMCLUST], i, rowm;
                int obscorr_0, obscorr_1, obsstat;
                int sampsz, ncol, numclust;
                int x, y, n, num, yij[NUMCLUST];
                int switchtrt = 0;
                int m_trt_switch;
                double pval;

                rowm = 0;
                obscorr_0 = 0;
```

```
                obscorr_1 = 0;
                obsstat = 0;
                sampsz = 0;
                pval = 0;
                ncol = NUMCLUST;

                FILE *fin;
        char ifile[250];
                fin = fopen(ifile, "r");

                if (fin != NULL) {
                        fscanf(fin, "%d", &numclust);
                        for (i = 0; i < numclust; i++) {
                                num = fscanf(fin, "%d %d %d", &x, &y, &n);
                                dose[i] = x;
                                yij[i] = y;
                                obsstat += x*y;
                                if (x == 0) {
                                        obscorr_0 += y*(n - y);
                                }
                                else if (x == 1) {
                                        obscorr_1 += y*(n - y);
                                        if (switchtrt == 0) {
                                                m_trt_switch = i;
                                                switchtrt = 1;
                                        }
                                }
                                sampsz += n;
                                rowm += y;
                                litter[i] = n;
                        }
                        fclose(fin);

                        trstat(&ncol, litter, dose, &sampsz, &rowm,
                                &m_trt_switch, &obsstat, &obscorr_0,
                                &obscorr_1, &pval);
                }
  return 0;
}
```

Exact Distribution C Code - trend.c:

```
// TREND.C: BUILDS NETWORK USING ALLOCATED VECTORS OF SUBNODES

#include <stdio.h>
#include <math.h>

typedef struct arc_tag ARC;
typedef struct snd_tag SUBND;
typedef struct rec_tag REC;

struct rec_tag {
        int pastr;  // past record
        double pastp;  // past probability
        REC *nextrec;
};

struct arc_tag {
        int arc;
        double pr;  // probability length
        ARC *nextarc;
        SUBND *child;
};

struct snd_tag {
        int parcorr_0;  // partial sum for the quadratic term when x = 0
        int parcorr_1;  // partial sum for the quadratic term when x = 1
        int spl;  // shortest path length (for the test statistic)
        int lpl;  // longest path length (for the test statistic)
        double tp;  // total probability
        int numpred;  // number of predecessor nodes created
        ARC *arc;
        REC *rec;
};

typedef struct node_tag {
        int splcorr;  // shortest path length for quadratic term
        int lplcorr;  // longest path length for quadratic term
        int upper;
        int lower;
```

```
        int numsucc;  // number of successor nodes
        SUBND *subnodes;
}NODE;

// trstat = trend statistic
void trstat(int *ncol, int table[], int uwt[], int *sampsz,
        int *rowm, int *m_trt_switch, int *obsstat,
        int *obscorr_0, int *obscorr_1, double *pval);
// ncol = number of columns in data = numclusters
// uwt = u weight = scores
// sampsz = total sample size
// rowm = number successes across all clusters - first sufficient stat (sum yi)
// obstat = observed linear rank test statistic
// obscorr_0 = observed correlation suff. statistic (quadratic term) when x = 0
// obscorr_1 = observed correlation suff. statistic (quadratic term) when x = 1

void trstat(int *ncol, int table[], int uwt[], int *sampsz,
        int *rowm, int *m_trt_switch, int *obsstat,
        int *obscorr_0, int *obscorr_1, double *pval) {

        // lowval = test statistic value (for p-value calculation, add up
        // everything with test stats larger than this value)
        int i, nnodes, high, lowval;
        // colm = num responses in ith cluster
        // wt = score for that dose (x-value)
        // stpos = starting position at each stage
        // table2 = sample size for each cluster, stored backwards (ex: for 3
        // clusters, if colm = [0,2,3,3], then sample size for cluster 1 = 3,
        // cluster 2 = 3, cluster 3 = 2)
        // minvl = min number of responses at that stage. For terminal node
        // (last stage), it should be equal to the first sufficient
        // statistic and it should be zero for all other stages)
        int *colm, *cumcol, *wt, *stpos, *minvl, *table2;
        // normcon = normalizeng constant - total probability for initial node
        double *fact, normcon;
        REC *currec;
        SUBND *cursnode;
        NODE *nodes;
```

```
FILE *fout, *distout;

// factorial log for total probability
void faclog(int sampsz, double *fact);
// calculate nodes
void calnds(int ncol, int *table, int rowm, int *nnodes, int *colm,
        int *cumcol, int *stpos, int *minvl);
// NDvector = create a vector of nodes
NODE *NDvector(long nl, long nh);
// forind = forward induction/pass - create network based on sum yi
void forind(int ncol, int rowm, int nnodes, int sampsz, int *colm,
        int *cumcol, double *fact, int *stpos, int *minvl, NODE *nodes);
// backind = backward induction/pass - create network based on quadratic
// term (prune results from forward pass to only include nodes that
// satisfy the quadratic condition)
void backind(int ncol, int rowm, int nnodes, int sampsz, int m_trt_switch,
        int obscorr_0, int obscorr_1, int *colm, int *cumcol, int *wt,
        double *fact, int *stpos, int *minvl, NODE *nodes);
void printnd(int nnodes, int obscorr_0, int obscorr_1, NODE *nodes);
// finalpass = final pass - calculate the p-value as the final forward
// pass through the network
void finalpass(int nnodes, int ncol, int *minvl, int *stpos,
        int obsstat, int lowval, int obscorr_0, int obscorr_1,
        NODE *nodes, double *rtail);
void free_NDvector(NODE *v, long nl, long nh);
void free_SNvector(SUBND *v, long nl, long nh);
void free_arc(ARC *arc);
void freerec(REC *rec);

if (*rowm == 0 || *rowm == *sampsz) {
        *pval = 1.0;
        return;
}

table2 = ivector(0, *ncol - 1);
colm = ivector(1, *ncol + 1);
cumcol = ivector(1, *ncol + 1);
wt = ivector(1, *ncol + 1);
stpos = ivector(1, *ncol + 1);
```

```
minvl = ivector(1, *ncol + 1);
fact = dvector(1, *sampsz + 1);

faclog(*sampsz, fact);
wt[1] = 0;
for (i = 2; i <= *ncol + 1; i++) {
        table2[i - 2] = table[*ncol - i + 1];  \
        wt[i] = uwt[i - 2];
}

calnds(*ncol, table2, *rowm, &nnodes, colm, cumcol, stpos, minvl);
//printf("\nnnodes = %d", nnodes);
nodes = NDvector(1, nnodes);
forind(*ncol, *rowm, nnodes, *sampsz, colm, cumcol,
        fact, stpos, minvl, nodes);
for (i = 1; i <= nnodes; i++) {
        nodes[i].upper = nodes[nnodes - i + 1].lplcorr;
        nodes[i].lower = nodes[nnodes - i + 1].splcorr;
}
// calculate nodes
calnds(*ncol, table, *rowm, &nnodes, colm, cumcol, stpos, minvl);
forind(*ncol, *rowm, nnodes, *sampsz, colm, cumcol,
        fact, stpos, minvl, nodes);

//printf("\nFinished forward induction...\n\n");
backind(*ncol, *rowm, nnodes, *sampsz, *m_trt_switch,
        *obscorr_0, *obscorr_1, colm, cumcol, wt, fact,
        stpos, minvl, nodes);
//printf("\nFinished backward induction...\n\n");

lowval=nodes[stpos[1]].subnodes[*obscorr_0 + *obscorr_1].spl;
// if you want the whole distribution of the test stat, then
// uncomment the following line (smallest value of test stat)
//lowval = *obsstat;
normcon = nodes[stpos[1]].subnodes[*obscorr_0 + *obscorr_1].tp;
//printf("\nNormalizing Constant = %f", normcon);

*pval = (double)0.0;
// this is where the final p-value is calculated:
```

```
finalpass(nnodes, *ncol, minvl, stpos, *obsstat, lowval,
        *obscorr_0, *obscorr_1, nodes, pval);
// cursnode = current subnode
cursnode = &(nodes[stpos[*ncol + 1]].subnodes[0]);
// currec = current record - last record that is processed
currec = cursnode->rec;
// if lowval is uncommented above, then uncomment the following lines to
// get the entire distribution of the test statistic
/*
printf("\n\nPERMUTATION DISTRIBUTION");
printf("\n\nt(x)                Pr(T=t)         \n");
printf(    "------------------------\n");
fprintf(distout,"\n\nPERMUTATION DISTRIBUTION");
fprintf(distout,"\n\nt(x)               Pr(T=t)         \n");
fprintf(distout,    "------------------------\n");
*/
while (currec != NULL) {
        fprintf(distout, "%6d %18.5f %18.5f\n", currec->pastr,
                (currec->pastp), exp(currec->pastp - normcon));
        if (currec->pastr >= *obsstat)
                *pval = *pval + exp(currec->pastp - normcon);
        currec = currec->nextrec;
}
fclose(distout);

cursnode = &(nodes[stpos[*ncol + 1]].subnodes[0]);
currec = cursnode->rec;
freerec(currec);
high = imin(nodes[stpos[*ncol + 1]].upper, *obscorr_0 + *obscorr_1);
free_SNvector(nodes[stpos[*ncol + 1]].subnodes,
        nodes[stpos[*ncol + 1]].lower, high);
free_NDvector(nodes, 1, nnodes);
free_ivector(table2, 0, *ncol - 1);
free_ivector(colm, 1, *ncol + 1);
free_ivector(cumcol, 1, *ncol + 1);
free_ivector(wt, 1, *ncol + 1);
free_ivector(stpos, 1, *ncol + 1);
free_ivector(minvl, 1, *ncol + 1);
free_dvector(fact, 1, *sampsz + 1);
```

```
        return;
}

void finalpass(int nnodes, int ncol, int *minvl, int *stpos, int obsstat,
        int lowval, int obscorr_0, int obscorr_1, NODE *nodes, double *rtail) {

        int pos, hlim, llim, r, stage, high, k, j;
        double npr;
        SUBND *cursnode, *succ;
        ARC *carc;
        REC *newrec, *crec, *curr, *nxt;
        // FILE *fout;

        REC *crerec(int r, double pr);  // create record
        double addlog(double num1, double num2);
        void freerec(REC *rec);
        void free_SNvector(SUBND *v, long nl, long nh);
        void free_arc(ARC *arc);

        // fout=fopen("finalpass.log","w");
        newrec = crerec(0, (double)0.0);
        nodes[stpos[1]].subnodes[obscorr_0 + obscorr_1].rec = newrec;
        for (stage = 1; stage <= ncol; stage++) {
                llim = minvl[stage];
                if (stage == 1) hlim = llim;
                else hlim = llim + stpos[stage - 1] - stpos[stage] - 1;
                /* fprintf(fout,"\n\nstage=%d, llim=%d, hlim=%d\n",stage,
                   llim,hlim); */
                for (pos = stpos[stage]; pos <= stpos[stage] + hlim - llim;
                        pos++) {
                //fprintf(fout,"\n   pos (node)=%d",pos);
                if (nodes[pos].subnodes == NULL) continue;
                high = imin(nodes[pos].upper, /*obscorr*/obscorr_0 + obscorr_1);
                // fprintf(fout,", low=%d, high=%d",nodes[pos].lower,high);
                for (k = nodes[pos].lower; k <= high; k++) {
                cursnode = &(nodes[pos].subnodes[k]);
                if (cursnode->parcorr_0 < 0 || cursnode->parcorr_1 < 0) {
                continue;
                }
```

```
// fprintf(fout,"\n      cursnode=%d %d %d %7.3f",
// cursnode->parcorr,cursnode->spl,cursnode->lpl,cursnode->tp);
carc = cursnode->arc;  //carc = current subnode arc
while (carc != NULL) {
succ = carc->child;
// fprintf(fout,"\n          curchild=%d %d %d %7.3f",
// succ->parcorr,succ->spl,succ->lpl,succ->tp);
crec = cursnode->rec;
while (crec != NULL) {
// from Formula: t_k + r_(1,k+1) (part 2, part a)
r = crec->pastr + carc->arc;
// from Formula: t_k + r_(1,k+1) +
// LP_1(k+1, s_(1,k+1), s_(2,k+1)) < t_obs
// (part 2, part a - in reverse (if NOT part a,
// then move on))
if (r + succ->lpl >= lowval) {
if (crec->pastp < 0.0000001)
npr = carc->pr;
else npr = crec->pastp + carc->pr;
newrec = crerec(r, npr);
if (succ->rec == NULL)
succ->rec = newrec;
else if (newrec->pastr < succ->rec->pastr) {
newrec->nextrec = succ->rec;
succ->rec = newrec;
}
else {
curr = succ->rec;
nxt = curr->nextrec;
while (nxt != NULL) {
if (r == curr->pastr ||
(nxt != NULL && (r < nxt->pastr))) {
        break;
}
curr = nxt;
nxt = curr->nextrec;
}
if (r == curr->pastr) {
curr->pastp = addlog(curr->pastp,
```

```
                newrec->pastp);
                free(newrec);
                }
                else {
                newrec->nextrec = nxt;
                curr->nextrec = newrec;
                }
                }
                }
                // move on to the next record and don't transfer this
                // record (if part 2, part a is true)
                crec = crec->nextrec;
                }
                carc = carc->nextarc;
                }
                if (cursnode->rec != NULL)
                freerec(cursnode->rec);
                }
                if (nodes[pos].subnodes != NULL) {
                for (j = nodes[pos].lower; j <= high; j++) {
                if (nodes[pos].subnodes[j].arc != NULL)
                free_arc(nodes[pos].subnodes[j].arc);
                }
                }
                free_SNvector(nodes[pos].subnodes, nodes[pos].lower, high);
                }
        }
        //fclose(fout);
}

REC * crerec(int r, double pr) {  // create record

        REC *record;

        record = (REC *)malloc(sizeof(REC));
        record->pastr = r;  // t_k = partial rank length
        // h_k = unnormalized sum of probabilites of all partial paths
    // with rank length r_1k
        record->pastp = pr;
```

```
        record->nextrec = NULL;
        return(record);
}

void freerec(REC *rec) {

        void freerec(REC *rec);

        if (rec->nextrec != NULL)
                freerec(rec->nextrec);
        free(rec);
        return;
}

void free_arc(ARC *arc) {

        void free_arc(ARC *arc);

        if (arc->nextarc != NULL)
                free_arc(arc->nextarc);
        free(arc);
        return;
}

void backind(int ncol, int rowm, int nnodes, int sampsz, int m_trt_switch,
        int obscorr_0, int obscorr_1, int *colm, int *cumcol, int *wt,
                double *fact, int *stpos, int *minvl, NODE *nodes) {

        // parm = partial sum of yi
        int llim, hlim, k, j, npos, parm, llm, hlm, ill, ihh, count;
        int num, ipl, remcorr_0, remcorr_1, ipl1, nodind, ref, high, up;
        double prarc, newprob;  // prarc = probability length
        SUBND *cursnode;
        ARC *newarc, *carc, *narc;
        SUBND *SNvector(long nl, long nh);
        void init(int lower, int upper, SUBND *subnodes);
        SUBND *crenode(/*int remcorr,*/ int remcorr_0, int remcorr_1, int lp,
                int sp, int ipl, double prarc);
        double prbar0(int coltot, int numsucc, double *fact);
```

```
// add all of the log-combinatorics terms
double addlog(double num1, double num2);
void dropnd(SUBND *cursnode);

npos = stpos[ncol + 1];
nodes[npos].subnodes = SNvector(0, 0);
nodes[npos].subnodes[0].parcorr_0 = obscorr_0;
nodes[npos].subnodes[0].parcorr_1 = obscorr_1;
nodes[npos].subnodes[0].lpl = 0;
nodes[npos].subnodes[0].spl = 0;
// number of predecessor nodes created
nodes[npos].subnodes[0].numpred = 0;
nodes[npos].subnodes[0].tp = 0.0;
nodes[npos].subnodes[0].arc = NULL;
nodes[npos].subnodes[0].rec = NULL;

// in for loop, j is decreasing -> going backwards through the network
for (j = ncol + 1; j >= m_trt_switch + 2; j--) {
        /* printf("\nAt the beginning of stage = %d with wt = %d",
                j, wt[j]); */
        llim = minvl[j];
        hlim = llim + stpos[j - 1] - stpos[j] - 1;
        // go through the number of nodes (in the gamma1 network)
        // at Stage j-1
        for (parm = llim; parm <= hlim; parm++, npos++) {
        //printf("\nAt node = %d, %d", j, parm);
        // at the beginning, we create the terminal node,
        // so nodes[npos].subnodes.parcorr = s2,
        // nodes[npos].subnodes.parcorr_0 = s2^1, and
        // nodes[npos].subnodes.parcorr_1 = s2^2.
        // hit "continue" if there are no quadruples to be created
        // from the Gamma(s_1) network for the node (j-1, parm)
        if (nodes[npos].subnodes == NULL) continue;
        int ref_1;
        // Grab the Successor Node:
        // from formula ( (k-1,u) in P(k, s_1k) ): u
        // (the minimum value u can take on for the k-1 stage)
        llm = imax(0, parm - colm[j]);
        // from formula ( (k-1,u) in P(k, s_1k) ): u
```

```
// (the maximum value u can take on for the k-1 stage)
hlm = imin(parm, cumcol[j - 1]);
ill = stpos[j - 1] + llm - minvl[j - 1];
ihh = ill + hlm - llm;
count = parm - llm;  // from Formula: s_1k - u
up = imin(nodes[npos].upper, obscorr_0 + obscorr_1);

for (nodind = nodes[npos].lower; nodind <= up; nodind++) {
cursnode = &(nodes[npos].subnodes[nodind]);
// for terminal node, this will always be equivalent to
// s2^1 > 0 or s2^2 > 0
if ((cursnode->parcorr_1) < 0) continue;
num = count;  // s_1k - u
// Create the Predecessor Nodes:
// k represents node position
for (k = ill; k <= ihh; k++, num--) {
// last term of v in the Formula (part (a)):
// (s_1k - u)(nk - s_1k + u)
ipl = num * (colm[j] - num);
remcorr_0 = obscorr_0;  // v in the Formula (part (a))
remcorr_1 = cursnode->parcorr_1 - ipl;
if (j == m_trt_switch + 2 && remcorr_1 != 0) continue;
if (j != m_trt_switch + 2 &&
(remcorr_1 > obscorr_1 || remcorr_1 < 0)) continue;
// number of predecessor nodes created
(cursnode->numpred)++;
// initialize subnode
if (nodes[k].subnodes == NULL) {
high = imin(nodes[k].upper, obscorr_0 + obscorr_1);
nodes[k].subnodes = SNvector(nodes[k].lower, high);
init(nodes[k].lower, high, nodes[k].subnodes);
}
newarc = (ARC *)malloc(sizeof(ARC));
// from Formula: x_k(s_1k - u) = RANK/ARC LENGTH
ipl1 = wt[j] * num;
// colm[j] = column total (n_k),
// num = (s_1k - u),
// from Formula: choose(n_k,(s_1k - u)) PROBABILITY LENGTH
// OF CONNECTING ARC
```

```
prarc = prbar0(colm[j], num, fact);
newarc->child = cursnode;
newarc->arc = ipl1;  // RANK/ARC LENGTH
// PROBABILITY LENGTH OF CONNECTING ARC
newarc->pr = prarc;
newarc->nextarc = NULL;
ref_1 = (obscorr_0 + obscorr_1) - (remcorr_0 + remcorr_1);
// If (k-1,u,v) has not yet been stored, then:
if ((nodes[k].subnodes[ref_1].lpl) < 0) {
nodes[k].subnodes[ref_1].parcorr_0 = remcorr_0;
nodes[k].subnodes[ref_1].parcorr_1 = remcorr_1;
// from Formula: LP_1(k-1,u,v)=r_1k + LP_1(k,s_1k,s_2k)
nodes[k].subnodes[ref_1].lpl = (cursnode->lpl) + ipl1;
// from Formala: SP_1(k-1,u,v)=r_1k + SP_1(k,s_1k,s_2k)
nodes[k].subnodes[ref_1].spl = (cursnode->spl) + ipl1;
// from Formula: TP(k-1,u,v)=c_ok * TP(k,s_1k,s_2k)
// (we add instead of multiply because we are working
// with the LOG probabilities)
nodes[k].subnodes[ref_1].tp = (cursnode->tp) + prarc;
nodes[k].subnodes[ref_1].arc = newarc;
}
// Else, (k-1,u,v) already exists (i.e., was found
// previously to be the predecessor of another node at the
// kth stage) and then:
else {
// from Formula (order switched):
// max{LP_1(k-1,u,v), r_1k + LP_1(k,s_1k,s_2k)}
if ((cursnode->lpl) + ipl1 >
nodes[k].subnodes[ref_1].lpl) {
// from Formula: LP_1(k-1,u,v)=max{LP_1(k-1,u,v),
// r_1k + LP_1(k,s_1k,s_2k)}
nodes[k].subnodes[ref_1].lpl =
cursnode->lpl + ipl1;
}
// from Formula (order switched):
// min{SP_1(k-1,u,v), r_1k + SP_1(k,s_1k,s_2k)}
if ((cursnode->spl) + ipl1 <
nodes[k].subnodes[ref_1].spl) {
// from Formula: SP_1(k-1,u,v)=min{SP_1(k-1,u,v),
```

```
        // r_1k + SP_1(k,s_1k,s_2k)}
        nodes[k].subnodes[ref_1].spl =
        cursnode->spl + ipl1;
        }
        // from Formula: c_ok * TP(k,s_1k,s_2k)
        newprob = prarc + cursnode->tp;
        // from Formula : TP(k-1,u,v)=TP(k-1,u,v) +
        // c_ok * TP(k,s_1k,s_2k)
        nodes[k].subnodes[ref_1].tp =
        addlog(nodes[k].subnodes[ref_1].tp, newprob);
        if (nodes[k].subnodes[ref_1].arc == NULL)
        nodes[k].subnodes[ref_1].arc = newarc;
        else {
        carc = nodes[k].subnodes[ref_1].arc;
        narc = carc->nextarc;
        while (narc != NULL) {
        carc = carc->nextarc;
        narc = carc->nextarc;
        }
        carc->nextarc = newarc;
        }
        }
        }
        // if node has no predeccesors then drop the node
        if ((cursnode->numpred) == 0)
        dropnd(cursnode);
        //printnd(nnodes, obscorr, remcorr_0, remcorr_1, nodes);
        }
        }
}
// same for loop as previous for loop, except now we are working with
// the other treatment group
// j is decreasing --> going backwards through the network
for (j = m_trt_switch + 1; j >= 2; j--) {
        /* printf("\nAt the beginning of stage = %d with wt = %d",
                j, wt[j]); */
        llim = minvl[j];
        hlim = llim + stpos[j - 1] - stpos[j] - 1;
        for (parm = llim; parm <= hlim; parm++, npos++) {
```

```
if (nodes[npos].subnodes == NULL) continue;
int ref_1;
llm = imax(0, parm - colm[j]);
hlm = imin(parm, cumcol[j - 1]);
ill = stpos[j - 1] + llm - minvl[j - 1];
ihh = ill + hlm - llm;
count = parm - llm;
up = imin(nodes[npos].upper, obscorr_0 + obscorr_1);
for (nodind = nodes[npos].lower; nodind <= up; nodind++) {
cursnode = &(nodes[npos].subnodes[nodind]);
if ((cursnode->parcorr_0) < 0) continue;
num = count;
for (k = ill; k <= ihh; k++, num--) {
ipl = num * (colm[j] - num);
remcorr_0 = cursnode->parcorr_0 - ipl;
remcorr_1 = 0;
if (j == 2 && remcorr_0 != 0) continue;
if (j != 2 && (remcorr_0 > obscorr_0 || remcorr_0 < 0)) {
continue;
}

(cursnode->numpred)++;
if (nodes[k].subnodes == NULL) {
high = imin(nodes[k].upper, obscorr_0 + obscorr_1);
nodes[k].subnodes = SNvector(nodes[k].lower, high);
init(nodes[k].lower, high, nodes[k].subnodes);
}
newarc = (ARC *)malloc(sizeof(ARC));
prarc = prbar0(colm[j], num, fact);
newarc->child = cursnode;
newarc->arc = ipl1;
newarc->pr = prarc;
newarc->nextarc = NULL;
ref_1 = (obscorr_0 + obscorr_1) - (remcorr_0 + remcorr_1);
if ((nodes[k].subnodes[ref_1].lpl) < 0) {
nodes[k].subnodes[ref_1].parcorr_0 = remcorr_0;
nodes[k].subnodes[ref_1].parcorr_1 = remcorr_1;
nodes[k].subnodes[ref_1].lpl = (cursnode->lpl) + ipl1;
nodes[k].subnodes[ref_1].spl = (cursnode->spl) + ipl1;
```

```
                nodes[k].subnodes[ref_1].tp = (cursnode->tp) + prarc;
                nodes[k].subnodes[ref_1].arc = newarc;
                }
                else {
                if ((cursnode->lpl) + ipl1 >
                nodes[k].subnodes[ref_1].lpl) {
                nodes[k].subnodes[ref_1].lpl =
                cursnode->lpl + ipl1;
                }
                if ((cursnode->spl) + ipl1 <
                nodes[k].subnodes[ref_1].spl) {
                nodes[k].subnodes[ref_1].spl =
                cursnode->spl + ipl1;
                }
                newprob = prarc + cursnode->tp;
                nodes[k].subnodes[ref_1].tp =
                addlog(nodes[k].subnodes[ref_1].tp, newprob);
                if (nodes[k].subnodes[ref_1].arc == NULL)
                nodes[k].subnodes[ref_1].arc = newarc;
                else {
                carc = nodes[k].subnodes[ref_1].arc;
                narc = carc->nextarc;
                while (narc != NULL) {
                carc = carc->nextarc;
                narc = carc->nextarc;
                }
                carc->nextarc = newarc;
                }
                }
                }
                if ((cursnode->numpred) == 0)
                dropnd(cursnode);
                //printnd(nnodes, obscorr, remcorr_0, remcorr_1, nodes);
                }
                }
        }
}

void dropnd(SUBND *cursnode) {
```

```
        SUBND *cnode;
        ARC *carc, *narc;

        cursnode->lpl = -1;
        cursnode->spl = -1;
        cursnode->tp = -1;
        cursnode->parcorr_0 = -1;
        cursnode->parcorr_1 = -1;
        carc = cursnode->arc;
        while (carc != NULL) {
                cnode = carc->child;
                (cnode->numpred)--;
                if ((cnode->numpred) == 0)
                        dropnd(cnode);
                narc = carc->nextarc;
                free(carc);
                carc = narc;
        }
        cursnode->arc = NULL;
        return;
}

void init(int lower, int upper, SUBND *subnodes) {

        int k;

        for (k = lower; k <= upper; k++) {
                subnodes[k].parcorr_0 = -1;
                subnodes[k].parcorr_1 = -1;
                subnodes[k].lpl = -1;
                subnodes[k].spl = -1;
                subnodes[k].tp = -1;
                subnodes[k].numpred = 0;
                subnodes[k].arc = NULL;
                subnodes[k].rec = NULL;
        }
        return;
}
```

```
SUBND *crenode(int remcorr_0, int remcorr_1, int lp, int sp,
        int ipl, double prarc) {

        SUBND *newnode;

        newnode = (SUBND *)malloc(sizeof(SUBND));
        newnode->parcorr_0 = remcorr_0;
        newnode->parcorr_1 = remcorr_1;
        newnode->lpl = lp + ipl;
        newnode->spl = sp + ipl;
        newnode->tp = prarc;
        newnode->numpred = 0;
        return(newnode);
}

void forind(int ncol, int rowm, int nnodes, int sampsz, int *colm,
        int *cumcol, double *fact, int *stpos, int *minvl, NODE *nodes) {

        int llim, hlim, k, j, npos, spl, lpl;
        // shortest path and longest path for the correlation
        // sufficient statistic
        void corrlpsp(int sampsz, int parm, int stage, int ncol, int rowm,
                int nnodes, int *colm, int *cumcol, int *stpos, int *minvl,
                NODE *nodes, int *spl, int *lpl);

        npos = stpos[1];  // last node
        nodes[npos].splcorr = 0;  // last node's splcorr = 0
        nodes[npos].lplcorr = 0;  // last node's lplcorr = 0
        nodes[npos].subnodes = NULL;    // last node's subnodes = NULL
        for (j = 2; j <= ncol + 1; j++) {  // go through each stage
                npos = stpos[j];
                llim = minvl[j];  // min value of s1_k at stage j
                // max value of s1_k at stage j:
                hlim = llim + stpos[j - 1] - stpos[j] - 1;
                // go through each value of s1_k at each stage j
                for (k = llim; k <= hlim; k++) {
                        corrlpsp(sampsz, k, j, ncol, rowm, nnodes, colm,
                                cumcol, stpos, minvl, nodes, &spl, &lpl);
```

```
                        nodes[npos].splcorr = spl;
                        nodes[npos].lplcorr = lpl;
                        nodes[npos].numsucc = 0;
                        nodes[npos].subnodes = NULL;
                        npos++;
                }
        }
}

void corrlpsp(int sampsz, int parm, int stage, int ncol, int rowm,
        int nnodes, int *colm, int *cumcol, int *stpos, int *minvl,
        NODE *nodes, int *spl, int *lpl) {

        // isp = ith shortest path
        // ilp = ith longest path
        int llm, hlm, ill, ihh, isp, ilp, ipl, isp2, ilp2, count, pos;

        llm = imax(0, parm - colm[stage]);
        hlm = imin(parm, cumcol[stage - 1]);
        // use arc lengths from nodes at this position to nodes
        // at ihh position
        ill = stpos[stage - 1] + llm - minvl[stage - 1];
        ihh = ill + hlm - llm;

        // parm = value of k (s1_k at stage j);
        // this part of r_2k formula: s_(1,k+1) - s_1k
        count = parm - llm;
        // r_2k = (s_(1,k+1)-s_1k )(n_(k+1)-s_(1,k+1)+s_1k )
        ipl = count * (colm[stage] - count);
        // ith shortest path - starting with arc from ill then
        // will go to arc ihh in the below for loop
        isp = nodes[ill].splcorr + ipl;
        ilp = nodes[ill].lplcorr + ipl;
        if (ill < ihh) {
                for (pos = ill + 1; pos <= ihh; pos++) {
                        count--;
                        ipl = count * (colm[stage] - count);
                        isp2 = nodes[pos].splcorr + ipl;
                        ilp2 = nodes[pos].lplcorr + ipl;
```

```
                        if (isp2 < isp) isp = isp2;
                        if (ilp2 > ilp) ilp = ilp2;
                }
        }
        *spl = isp;
        *lpl = ilp;
        return;
}

// calculates nodes - builds network before the first pass
void calnds(int ncol, int *table, int rowm, int *nnodes, int *colm,
        int *cumcol, int *stpos, int *minvl) {

        // Formula: R(k-1, s_(1,k-1)) = {(k,u):max(s_(1,k-1),
        // s_1 - sum^N_(l=k)nl) <= u <= min(s_1, s_(1,k-1) + n_k)}
        // where rowm = s_1, cumcol[ncol + 1] = sum^N_(l=k)nl
        // colm = sample size for each cluster, stored backwards

        int i, stage, iconst, llim, hlim, npos;
        colm[1] = 0;
        cumcol[1] = 0;

        // build colm and cumcol (cummulative sum of colm) vectors
        for (i = 2; i <= ncol + 1; i++) {
                // column total - sample size in each treatment/study
                colm[i] = table[i - 2];
                // cumulative column totals - 7 is the total sample size
                cumcol[i] = cumcol[i - 1] + colm[i];
        }
        // this part of Formula: s_1 - sum^N_(l=k)nl
        // FOR THE INITIAL NODE - at the initial node, this number
        // will be the most negative it can be, then it decreases.
        // That's why we add a positive number to this negative
        // number down in the for loop:
        iconst = rowm - cumcol[ncol + 1];
        npos = 2;
        minvl[ncol + 1] = rowm;  // this part of Formula: s_1
        stpos[ncol + 1] = 1;
        // goes through stages 0 to ncol - 1 and finds the minimum partial
```

```
        // sum (llim) and maximum parital sum (hlim) for that stage
        // if llim = 0 and hlim = 2, then at that stage (say, stage k),
        // there are 3 successor nodes ([k,0], [k,1], and [k,2])
        // npos is a cumulative sum of the number of nodes at each stage
        for (stage = ncol - 1; stage >= 0; stage--) {
                // lower limit for u -> this part of Formula: max(s_(1,k-1),
                // s_1 - sum^N_(l=k)nl) - see comment above iconst declaration
                // essentially taking iconst + cummulative sample size
                // at each stage
                llim = imax(0, iconst + cumcol[stage + 1]);
                // upper limit for u
                hlim = imin(rowm, cumcol[stage + 1]);
                stpos[stage + 1] = npos;
                minvl[stage + 1] = llim;
                npos = npos + (hlim - llim + 1);
        }
        *nnodes = npos - 1;
        return;
}

double prbar0(int coltot, int numsucc, double *fact) {

        double logprob;

        if (coltot == numsucc || numsucc == 0)
                logprob = 0.0;
        else
                logprob = fact[coltot + 1] - fact[numsucc + 1] -
                        fact[coltot - numsucc + 1];
        return(logprob);
}


double testmax(double x, double y);

double testmax(double x, double y) {
        if (x > y) return(x);
        else return(y);
}
```

```
double addlog(double num1, double num2) {

        double total2, t12, t22, tmax2;
        double q, x, s;

        tmax2 = testmax(num1, num2);
        t12 = testmax(num1 - tmax2, -80.0);
        t22 = testmax(num2 - tmax2, -80.0);
        q = exp(t12);
        x = exp(t22);
        s = q + x;
        total2 = tmax2 + log(exp(t12) + exp(t22));
        return(total2);
}

void faclog(int sampsz, double *fact) {

        int i;

        fact[1] = 0.0;
        for (i = 1; i <= sampsz; i++) {
                fact[i + 1] = fact[i] + log((double)i);
        }
        return;
}

void printnd(int nnodes, int obscorr_0, int obscorr_1, NODE *nodes) {

        int i, k, high;
        SUBND *cursnode;
        ARC *carc;
        FILE *fout;

        fout = fopen("clust.out", "a");
        fprintf(fout, "\n\n nnodes=%d\n\nNode SPL LPL LOW UPP\n", nnodes,
                "--------------------");
        for (i = 1; i <= nnodes; i++) {
                fprintf(fout, "\n%d   %d   %d   %d   %d", i, nodes[i].splcorr,
```

```
                nodes[i].lplcorr, nodes[i].lower, nodes[i].upper);
                if (nodes[i].subnodes != NULL) {
                high = imin(nodes[i].upper, obscorr_0 + obscorr_1);
                for (k = nodes[i].lower; k <= high; k++) {
                if (nodes[i].subnodes[k].parcorr_0 < 0 ||
                nodes[i].subnodes[k].parcorr_1 < 0) continue;
                cursnode = &(nodes[i].subnodes[k]);
                fprintf(fout, "\n       %d   %d   %d   %d   %5.3f",
                cursnode->parcorr_0, cursnode->parcorr_1, cursnode->spl,
                cursnode->lpl, cursnode->tp);
                carc = cursnode->arc;
                while (carc != NULL) {
                fprintf(fout, "\n               arc:  %d   %5.3f",
                carc->arc, carc->pr);
                fprintf(fout, "\n               child:  %d %d %d %d %5.3f",
                carc->child->parcorr_0, carc->child->parcorr_1,
                carc->child->spl, carc->child->lpl, carc->child->tp);
                carc = carc->nextarc;
                }
                }
                }
        }
        fprintf(fout, "\n\n\n");
        fclose(fout);
}

#define NR_END 1
#define FREE_ARG char*

// allocate a NODE vector with subscript range v[nl..nh]
NODE *NDvector(long nl, long nh) {
        NODE *v;

        v = (NODE *)malloc((size_t)((nh - nl + 1 + NR_END) * sizeof(NODE)));
        if (!v) nrerror("allocation failure in NDvector()");
        return v - nl + NR_END;
}

// allocate a NODE vector with subscript range v[nl..nh]
```

```
SUBND *SNvector(long nl, long nh) {
        SUBND *v;

        v = (SUBND *)malloc((size_t)((nh - nl + 1 + NR_END) *
                sizeof(SUBND)));
        if (!v) nrerror("allocation failure in SNvector()");
        return v - nl + NR_END;
}

// free a float vector allocated with vector()
void free_NDvector(NODE *v, long nl, long nh) {
        free((FREE_ARG)(v + nl - NR_END));
}

// free a float vector allocated with vector()
void free_SNvector(SUBND *v, long nl, long nh) {
        free((FREE_ARG)(v + nl - NR_END));
}
```

Exact Distribution C Code - numrout.c:

```
#include "nrutil.h"
#include <stdio.h>

int imax(int x, int y);
int imin(int x, int y);
double dmax(double x, double y);
float fmax(float x, float y);

/* CAUTION: This is the ANSI C (only) version of the Numerical Recipes
utility file nrutil.c.  Do not confuse this file with the same-named
file nrutil.c that is supplied in the same subdirectory or archive
as the header file nrutil.h.  *That* file contains both ANSI and
traditional K&R versions, along with #ifdef macros to select the
correct version.  *This* file contains only ANSI C.               */

#define NR_END 1
#define FREE_ARG char*

// Numerical Recipes standard error handler
void nrerror(char error_text[]) {
        fprintf(stderr, "Numerical Recipes run-time error...\n");
        fprintf(stderr, "%s\n", error_text);
        fprintf(stderr, "...now exiting to system...\n");
        exit(1);
}

// allocate a float vector with subscript range v[nl..nh]
float *vector(long nl, long nh) {
        float *v;

        v = (float *)malloc((size_t)((nh - nl + 1 + NR_END) *
                sizeof(float)));
        if (!v) nrerror("allocation failure in vector()");
        return v - nl + NR_END;
}

// allocate an int vector with subscript range v[nl..nh]
int *ivector(long nl, long nh) {
```

```
        int *v;

        v = (int *)malloc((size_t)((nh - nl + 1 + NR_END) * sizeof(int)));
        if (!v) nrerror("allocation failure in ivector()");
        return v - nl + NR_END;
}

// allocate an unsigned char vector with subscript range v[nl..nh]
unsigned char *cvector(long nl, long nh) {
        unsigned char *v;

        v = (unsigned char *)malloc((size_t)((nh - nl + 1 + NR_END) *
                sizeof(unsigned char)));
        if (!v) nrerror("allocation failure in cvector()");
        return v - nl + NR_END;
}

// allocate an unsigned long vector with subscript range v[nl..nh]
unsigned long *lvector(long nl, long nh) {
        unsigned long *v;

        v = (unsigned long *)malloc((size_t)((nh - nl + 1 + NR_END) *
                sizeof(long)));
        if (!v) nrerror("allocation failure in lvector()");
        return v - nl + NR_END;
}

// allocate a double vector with subscript range v[nl..nh]
double *dvector(long nl, long nh) {
        double *v;

        v = (double *)malloc((size_t)((nh - nl + 1 + NR_END) *
                sizeof(double)));
        if (!v) nrerror("allocation failure in dvector()");
        return v - nl + NR_END;
}

// allocate a float matrix with subscript range m[nrl..nrh][ncl..nch]
float **matrix(long nrl, long nrh, long ncl, long nch) {
```

```
        long i, nrow = nrh - nrl + 1, ncol = nch - ncl + 1;
        float **m;

        // allocate pointers to rows
        m = (float **)malloc((size_t)((nrow + NR_END) * sizeof(float*)));
        if (!m) nrerror("allocation failure 1 in matrix()");
        m += NR_END;
        m -= nrl;

        // allocate rows and set pointers to them
        m[nrl] = (float *)malloc((size_t)((nrow*ncol + NR_END) *
                sizeof(float)));
        if (!m[nrl]) nrerror("allocation failure 2 in matrix()");
        m[nrl] += NR_END;
        m[nrl] -= ncl;

        for (i = nrl + 1; i <= nrh; i++) m[i] = m[i - 1] + ncol;

        // return pointer to array of pointers to rows
        return m;
}

// allocate a double matrix with subscript range m[nrl..nrh][ncl..nch]
double **dmatrix(long nrl, long nrh, long ncl, long nch) {
        long i, nrow = nrh - nrl + 1, ncol = nch - ncl + 1;
        double **m;

        /* allocate pointers to rows */
        m = (double **)malloc((size_t)((nrow + NR_END) * sizeof(double*)));
        if (!m) nrerror("allocation failure 1 in matrix()");
        m += NR_END;
        m -= nrl;

        /* allocate rows and set pointers to them */
        m[nrl] = (double *)malloc((size_t)((nrow*ncol + NR_END) *
                sizeof(double)));
        if (!m[nrl]) nrerror("allocation failure 2 in matrix()");
        m[nrl] += NR_END;
        m[nrl] -= ncl;
```

```
        for (i = nrl + 1; i <= nrh; i++) m[i] = m[i - 1] + ncol;

        /* return pointer to array of pointers to rows */
        return m;
}

// allocate a int matrix with subscript range m[nrl..nrh][ncl..nch]
int **imatrix(long nrl, long nrh, long ncl, long nch) {
        long i, nrow = nrh - nrl + 1, ncol = nch - ncl + 1;
        int **m;

        /* allocate pointers to rows */
        m = (int **)malloc((size_t)((nrow + NR_END) * sizeof(int*)));
        if (!m) nrerror("allocation failure 1 in matrix()");
        m += NR_END;
        m -= nrl;

        /* allocate rows and set pointers to them */
        m[nrl] = (int *)malloc((size_t)((nrow*ncol + NR_END) * sizeof(int)));
        if (!m[nrl]) nrerror("allocation failure 2 in matrix()");
        m[nrl] += NR_END;
        m[nrl] -= ncl;

        for (i = nrl + 1; i <= nrh; i++) m[i] = m[i - 1] + ncol;

        /* return pointer to array of pointers to rows */
        return m;
}

// point a submatrix [newrl..][newcl..] to a[oldrl..oldrh][oldcl..oldch]
float **submatrix(float **a, long oldrl, long oldrh, long oldcl, long oldch,
        long newrl, long newcl) {
        long i, j, nrow = oldrh - oldrl + 1, ncol = oldcl - newcl;
        float **m;

        /* allocate array of pointers to rows */
        m = (float **)malloc((size_t)((nrow + NR_END) * sizeof(float*)));
        if (!m) nrerror("allocation failure in submatrix()");
```

```
        m += NR_END;
        m -= newrl;

        /* set pointers to rows */
        for (i = oldrl, j = newrl; i <= oldrh; i++, j++) m[j] = a[i] + ncol;

        /* return pointer to array of pointers to rows */
        return m;
}

/* allocate a float matrix m[nrl..nrh][ncl..nch] that points to the matrix
declared in the standard C manner as a[nrow][ncol], where nrow=nrh-nrl+1
and ncol=nch-ncl+1. The routine should be called with the address
&a[0][0] as the first argument. */
float **convert_matrix(float *a, long nrl, long nrh, long ncl, long nch) {
        long i, j, nrow = nrh - nrl + 1, ncol = nch - ncl + 1;
        float **m;

        /* allocate pointers to rows */
        m = (float **)malloc((size_t)((nrow + NR_END) * sizeof(float*)));
        if (!m) nrerror("allocation failure in convert_matrix()");
        m += NR_END;
        m -= nrl;

        /* set pointers to rows */
        m[nrl] = a - ncl;
        for (i = 1, j = nrl + 1; i<nrow; i++, j++) m[j] = m[j - 1] + ncol;
        /* return pointer to array of pointers to rows */
        return m;
}

// allocate a float 3tensor with range t[nrl..nrh][ncl..nch][ndl..ndh]
float ***f3tensor(long nrl, long nrh, long ncl, long nch, long ndl, long ndh) {
        long i, j, nrow = nrh - nrl + 1, ncol = nch - ncl + 1;
        long ndep = ndh - ndl + 1;
        float ***t;

        /* allocate pointers to pointers to rows */
        t = (float ***)malloc((size_t)((nrow + NR_END) * sizeof(float**)));
```

```
        if (!t) nrerror("allocation failure 1 in f3tensor()");
        t += NR_END;
        t -= nrl;

        /* allocate pointers to rows and set pointers to them */
        t[nrl] = (float **)malloc((size_t)((nrow*ncol + NR_END) *
                sizeof(float*)));
        if (!t[nrl]) nrerror("allocation failure 2 in f3tensor()");
        t[nrl] += NR_END;
        t[nrl] -= ncl;

        /* allocate rows and set pointers to them */
        t[nrl][ncl] = (float *)malloc((size_t)((nrow*ncol*ndep + NR_END) *
                sizeof(float)));
        if (!t[nrl][ncl]) nrerror("allocation failure 3 in f3tensor()");
        t[nrl][ncl] += NR_END;
        t[nrl][ncl] -= ndl;

        for (j = ncl + 1; j <= nch; j++) t[nrl][j] = t[nrl][j - 1] + ndep;
        for (i = nrl + 1; i <= nrh; i++) {
                t[i] = t[i - 1] + ncol;
                t[i][ncl] = t[i - 1][ncl] + ncol*ndep;
                for (j = ncl + 1; j <= nch; j++) t[i][j] = t[i][j - 1] + ndep;
        }

        /* return pointer to array of pointers to rows */
        return t;
}

// free a float vector allocated with vector()
void free_vector(float *v, long nl, long nh) {
        free((FREE_ARG)(v + nl - NR_END));
}

// free an int vector allocated with ivector()
void free_ivector(int *v, long nl, long nh) {
        free((FREE_ARG)(v + nl - NR_END));
}
```

```
// free an unsigned char vector allocated with cvector()
void free_cvector(unsigned char *v, long nl, long nh) {
        free((FREE_ARG)(v + nl - NR_END));
}

// free an unsigned long vector allocated with lvector()
void free_lvector(unsigned long *v, long nl, long nh) {
        free((FREE_ARG)(v + nl - NR_END));
}

// free a double vector allocated with dvector()
void free_dvector(double *v, long nl, long nh) {
        free((FREE_ARG)(v + nl - NR_END));
}

// free a float matrix allocated by matrix()
void free_matrix(float **m, long nrl, long nrh, long ncl, long nch) {
        free((FREE_ARG)(m[nrl] + ncl - NR_END));
        free((FREE_ARG)(m + nrl - NR_END));
}

// free a double matrix allocated by dmatrix()
void free_dmatrix(double **m, long nrl, long nrh, long ncl, long nch) {
        free((FREE_ARG)(m[nrl] + ncl - NR_END));
        free((FREE_ARG)(m + nrl - NR_END));
}

// free an int matrix allocated by imatrix()
void free_imatrix(int **m, long nrl, long nrh, long ncl, long nch) {
        free((FREE_ARG)(m[nrl] + ncl - NR_END));
        free((FREE_ARG)(m + nrl - NR_END));
}

// free a submatrix allocated by submatrix()
void free_submatrix(float **b, long nrl, long nrh, long ncl, long nch) {
        free((FREE_ARG)(b + nrl - NR_END));
}

// free a matrix allocated by convert_matrix()
```

```
void free_convert_matrix(float **b, long nrl, long nrh, long ncl, long nch) {
        free((FREE_ARG)(b + nrl - NR_END));
}

// free a float f3tensor allocated by f3tensor()
void free_f3tensor(float ***t, long nrl, long nrh, long ncl, long nch,
        long ndl, long ndh) {
        free((FREE_ARG)(t[nrl][ncl] + ndl - NR_END));
        free((FREE_ARG)(t[nrl] + ncl - NR_END));
        free((FREE_ARG)(t + nrl - NR_END));
}
/* (C) Copr. 1986-92 Numerical Recipes Software #(k3#12#1i.. */

int imax(int x, int y) {

        if (x>y) return(x);
        else return(y);
}

int imin(int x, int y) {

        if (x<y) return(x);
        else return(y);
}

double dmax(double x, double y) {

        if (x > y) return(x);
        else return(y);
}

float fmax(float x, float y) {

        if (x>y) return(x);
        else return(y);
}
```

Exact Distribution C Code - nrutil.h:

```
#ifndef _NR_UTILS_H_
#define _NR_UTILS_H_

static float sqrarg;
#define SQR(a) ((sqrarg=(a)) == 0.0 ? 0.0 : sqrarg*sqrarg)

static double dsqrarg;
#define DSQR(a) ((dsqrarg=(a)) == 0.0 ? 0.0 : dsqrarg*dsqrarg)

static double dmaxarg1, dmaxarg2;
#define DMAX(a,b) (dmaxarg1=(a),dmaxarg2=(b),(dmaxarg1) > (dmaxarg2) ?\
        (dmaxarg1) : (dmaxarg2))

static double dminarg1, dminarg2;
#define DMIN(a,b) (dminarg1=(a),dminarg2=(b),(dminarg1) < (dminarg2) ?\
        (dminarg1) : (dminarg2))

static float maxarg1, maxarg2;
#define FMAX(a,b) (maxarg1=(a),maxarg2=(b),(maxarg1) > (maxarg2) ?\
        (maxarg1) : (maxarg2))

static float minarg1, minarg2;
#define FMIN(a,b) (minarg1=(a),minarg2=(b),(minarg1) < (minarg2) ?\
        (minarg1) : (minarg2))

static long lmaxarg1, lmaxarg2;
#define LMAX(a,b) (lmaxarg1=(a),lmaxarg2=(b),(lmaxarg1) > (lmaxarg2) ?\
        (lmaxarg1) : (lmaxarg2))

static long lminarg1, lminarg2;
#define LMIN(a,b) (lminarg1=(a),lminarg2=(b),(lminarg1) < (lminarg2) ?\
        (lminarg1) : (lminarg2))

static int imaxarg1, imaxarg2;
#define IMAX(a,b) (imaxarg1=(a),imaxarg2=(b),(imaxarg1) > (imaxarg2) ?\
        (imaxarg1) : (imaxarg2))

static int iminarg1, iminarg2;
```

```
#define IMIN(a,b) (iminarg1=(a),iminarg2=(b),(iminarg1) < (iminarg2) ?\
        (iminarg1) : (iminarg2))

#define SIGN(a,b) ((b) >= 0.0 ? fabs(a) : -fabs(a))

void nrerror(char error_text[]);
float *vector(long nl, long nh);
int *ivector(long nl, long nh);
unsigned char *cvector(long nl, long nh);
unsigned long *lvector(long nl, long nh);
double *dvector(long nl, long nh);
float **matrix(long nrl, long nrh, long ncl, long nch);
double **dmatrix(long nrl, long nrh, long ncl, long nch);
int **imatrix(long nrl, long nrh, long ncl, long nch);
float **submatrix(float **a, long oldrl, long oldrh, long oldcl, long oldch,
        long newrl, long newcl);
float **convert_matrix(float *a, long nrl, long nrh, long ncl, long nch);
float ***f3tensor(long nrl, long nrh, long ncl, long nch, long ndl, long ndh);
void free_vector(float *v, long nl, long nh);
void free_ivector(int *v, long nl, long nh);
void free_cvector(unsigned char *v, long nl, long nh);
void free_lvector(unsigned long *v, long nl, long nh);
void free_dvector(double *v, long nl, long nh);
void free_matrix(float **m, long nrl, long nrh, long ncl, long nch);
void free_dmatrix(double **m, long nrl, long nrh, long ncl, long nch);
void free_imatrix(int **m, long nrl, long nrh, long ncl, long nch);
void free_submatrix(float **b, long nrl, long nrh, long ncl, long nch);
void free_convert_matrix(float **b, long nrl, long nrh, long ncl, long nch);
void free_f3tensor(float ***t, long nrl, long nrh, long ncl, long nch,
        long ndl, long ndh);

#endif /* _NR_UTILS_H_ */
```

Exact Estimation Method R Code - ExactEstimation.R:

```
# Load the Conditional Maximum Likelihood Estimate Function
source("CMLE.R")

# Example Meta-Analysis Data Set with 10 Studies
x <- c(rep(0, 10), rep(1, 10))  # 0 = Control Group, 1 = Treatment Group
y <- c(2, 1, 3, 4, 2, 1, 1, 3, 3, 4, 1, 1, 3, 2, 2, 0, 2, 1, 0)  # Events
n <- c(42, 36, 26, 18, 39, 41, 45, 46, 47, 34, 29, 25, 17, 26,
       36, 31, 16, 31, 44, 29)  # Total
t_obs <- as.numeric(x %*% y)  # Observed Test Statistic

# Exact Distribution (Found Using C Code)
test_statistics <- 14:19
normalized_probabilities <- c(0.00076, 0.05204, 0.38865,
                              0.47705, 0.08142, 0.00009)
normalized_probabilities <- ifelse(normalized_probabilities == 0,
                                   1.0e-16,
                                   normalized_probabilities)

# Throw out Data Sets where t_obs is at Either Extreme of the Distribution
if (t_obs == max(test_statistics) ||
    t_obs == min(test_statistics) ||
    sum(normalized_probabilities[test_statistics > t_obs]) < 1.0e-15 ||
    sum(normalized_probabilities[test_statistics < t_obs]) < 1.0e-15) {

  logOR_hat <- NA
  CI_lower <- NA
  CI_upper <- NA

} else {

  # Scale Down Test Statistics to Start at 0
  t_obs_scaled <- t_obs - min(test_statistics)
  test_statistics_scaled <- test_statistics - min(test_statistics)

  # use CMLE since t_obs not at extremes of distribution (instead of MUE)
  logOR_hat <- CMLE(prob = normalized_probabilities,
                    u = test_statistics_scaled,
                    t_obs = t_obs_scaled,
```

```
                    t_or_alpha = t_obs_scaled,
                    ci = FALSE)
  CI_lower <- CMLE(prob = normalized_probabilities,
                   u = test_statistics_scaled,
                   t_obs = t_obs_scaled,
                   t_or_alpha = 0.025,
                   ci = TRUE,
                   dir = "lower")
  CI_upper <- CMLE(prob = normalized_probabilities,
                   u = test_statistics_scaled,
                   t_obs = t_obs_scaled,
                   t_or_alpha = 0.025,
                   ci = TRUE,
                   dir = "upper")
}
```

Exact Estimation Method R Code - CMLE.R:

```
# Conditional Maximum Likelihood Estimation

CMLE <- function(prob, u, t_obs, t_or_alpha, ci = FALSE,
                 dir = "upper", lower_bound = -15, upper_bound = 15) {

  prob <- log(prob)  # Normalized Probabilities on the Log Scale

  # Objective Function for Log Odds Ratio Estimate and CI Estimate
  beta_objective_fun <- function(beta0, prob, u, t_obs, t_or_alpha,
                                 ci = FALSE, dir = "upper") {

    add_log_probs <- prob[1] + u[1] * beta0

    for (q in 2:length(u)) {  # add log probabilities
      log_p <- max(add_log_probs, prob[q] + u[q] * beta0)
      add_log_probs <- log_p + log(exp(add_log_probs - log_p) +
                                     exp(prob[q] + u[q] * beta0 - log_p))
    }

    num <- prob + (u * beta0)
    num2 <- exp(num - add_log_probs)

    if (ci) {
      final <- ifelse(dir == "lower",
                      sum(num2[u >= t_obs]),
                      sum(num2[u <= t_obs]))
    } else {
      final <- sum(u * num2) / sum(num2)
    }
    abs(t_or_alpha - final)
  }

  est_res <- optimize(beta_objective_fun, c(lower_bound, upper_bound),
                      prob, u, t_obs, t_or_alpha, ci, dir,
                      tol = 0.00000001, maximum = FALSE)
  result <- est_res$minimum
  result
}
```

Simulations R Code - ma.sim.dat.R:

```
# Code to simulate meta-analysis data

ma.sim.data <- function(theta, k = 20, mu, var_mu = 0.5, tau2,
                        min_unif = 50, max_unif = 100) {

  # Generate sample size for trt group
  n_it <- round(runif(k, min = min_unif, max = max_unif))
  # Generate sample size for ctrl group
  n_ic <- round(runif(k, min = min_unif, max = max_unif))

  # Generate error terms for ctrl and trt groups
  e1 <- rnorm(length(mu), 0, sqrt(var_mu))
  e2 <- rnorm(length(mu), 0, sqrt(tau2))

  # Generate response/event rate for ctrl group
  p_ic <- exp(mu + e1) / (1 + exp(mu + e1))
  # Generate number of responses/events for ctrl group
  x_ic <- sapply(1:k, function(x) {rbinom(1, n_ic[x], p_ic)})

  # Generate response/event rate for trt group
  p_it <- exp(mu + e1 + theta + e2) / (1 + exp(mu + e1 + theta + e2))
  # Generage number of responses/events for trt group
  x_it <- sapply(1:k, function(x) {rbinom(1, n_it[x], p_it)})

  data_set <- list("CTRL_n" = n_ic, "TRT_n" = n_it,
                   "CTRL_event" = x_ic, "TRT_event" = x_it)
  return(data_set)
}
```

Simulations R Code - Generate_Data.R:

```
source("ma.sim.data.R")

num_reps <- 10000
theta_sim <- 0
k_sim <- 10
mu_sim <- -4
var_mu_sim <- 0.5
tau2_sim <- 0.4
min_unif_sim <- 10
max_unif_sim <- 50

generate.dat <- function(theta, k, mu, var_mu, tau2, min_unif, max_unif) {

  sdata <- ma.sim.data(theta, k, mu, var_mu, tau2, min_unif, max_unif)
  # Convert sdata to matrix form for exact test
  colNamesTRTGrp <- c(rep(0, length(sdata$CTRL_n)),
                      rep(1, length(sdata$CTRL_n)))
  # (4 * num studies) for the desired number of cells in the matrix
  sdata_formatted <- rep(0, 4 * length(sdata$CTRL_n))
  # put ctrl event in every other spot in the first half of sdata_formatted
  # (starting with the first element)
  sdata_formatted[seq(1, length(sdata_formatted) / 2, 2)] <- sdata$CTRL_event
  # put ctrl nonevent in every other spot in the first half of sdata_formatted
  # (starting with the second element)
  sdata_formatted[seq(2, length(sdata_formatted) / 2,
                      2)] <- sdata$CTRL_n - sdata$CTRL_event
  # put trt event in every other spot in the second half of sdata_formatted
  # (starting with the first element of the second half of sdata_formatted)
  sdata_formatted[seq((length(sdata_formatted) / 2) + 1,
                      length(sdata_formatted), 2)] <- sdata$TRT_event
  # put trt nonevent in every other spot in the second half of sdata_formatted
  # (starting with the second element of the second half of sdata_formatted)
  sdata_formatted[seq((length(sdata_formatted) / 2) + 2,
                      length(sdata_formatted),
                      2)] <- sdata$TRT_n - sdata$TRT_event
  finaldat <- matrix(sdata_formatted, nrow = 2, byrow = FALSE,
                     dimnames = list(Out = c(1, 0), TRT = colNamesTRTGrp))
  return(list(finaldat))
```

```
}

set.seed(1993)
all_dat_full <- replicate(num_reps, generate.dat(theta = 0,
                                                 k = k_sim,
                                                 mu = mu_sim,
                                                 var_mu = var_mu_sim,
                                                 tau2 = tau2_sim,
                                                 min_unif = min_unif_sim,
                                                 max_unif = max_unif_sim))

# If all studies have no events in the treatment group, flag with a "1"
zero_trt_events <- sapply(all_dat_full,
                          function(x) ifelse(all(x[1, colnames(x) == 1] == 0),
                                             1, 0))
sum(zero_trt_events)
all_dat <- all_dat_full[which(zero_trt_events != 1)]  # remove such studies

# If all studies have no events in the control group, flag with a "1"
zero_ctrl_events <- sapply(all_dat,
                           function(x) ifelse(all(x[1, colnames(x) == 0] == 0),
                                              1, 0))
sum(zero_ctrl_events)
all_dat <- all_dat[which(zero_ctrl_events != 1)]  # remove such studies

rm(list=ls()[! ls() %in% c("all_dat", "tau2_sim", "mu_sim")])
save.image(paste0("Theta=0,Tau2=", tau2_sim, ",Mu=", mu_sim, ".RData"))
```

Appendix D: Code for Chapter 3

This appendix contains selected code for Chapter 3, and includes the following:

- nma.sim.dat.R
- Generate_NMA_Data.R

For the simulations in Chapter 3, nma.sim.dat.R and Generate_NMA_Data.R were used to simulate the network meta–analysis data sets. These data sets were then run through the exact distribution C code and the exact estimation R code (both in Appendix C) to obtain exact estimates.

Simulations R Code - nma.sim.dat.R:

```
# Function for generating binary network meta-analysis data
# Two treatments A and B have been compared head-to-head against another
# treatment C, but not against each other. The goal is to obtain an indirect
# estimate of the BC comparison.
# The simulation structure is based off:
# "Estimating the Power of Indirect Comparisons: A Simulation Study"
# by E. J. Mills, I. Ghement, C. O'Regan, and K. Thorlund

# k_bc is the number of trials for the BC comparison
# k_ac is the number of trials for the CA comparison
# min_n is the minimum number of participants in a study
# max_n is the maximum number of participants in a study
# prop_n_in_ctrl is the proportion of participants in the control group (versus
#                        the treatment group)
# pi_c is the true average event rate in the common comparator group C
# OR_bc is the true relative effect of BC, quantified as an odds ratio
# OR_ac is the true relative effect of CA, quantified as an odds ratio
# tau2 is the between-study variance, assumed constant across the BC and AC
#              comparisons

# Note: knowing OR_bc and OR_ac allows the determination of OR_ab, via the
# formula: OR_ab = exp(log(OR_ac) - log(OR_bc))

set.seed(1993)

nma.sim.data <- function(k_bc = 5, k_ac = 5, min_n = 10, max_n = 50,
                         prop_n_in_ctrl = 1 / 2, pi_c = 0.10,
                         OR_bc = 1.4, OR_ac = 1.4, tau2 = 0.2) {

  # ------------------------ BC comparison ----------------------------------#
  # Generate sample size (round to nearest even integer)
  n_i <- 2 * round(runif(n = k_bc, min = min_n, max = max_n) / 2)
  # sample size in arm C for BC comparison
  n_ic <- floor(n_i * prop_n_in_ctrl)
  # sample size in arm B for BC comparison
  n_ib <- n_i - n_ic

  # Generate number of events for arm C
```

```
  pi_ic <- runif(n = k_bc, min = pi_c - (pi_c / 2), max = pi_c + (pi_c / 2))
  e_ic <- sapply(1:k_bc, function(x) {rbinom(1, n_ic[x], pi_ic[x])})

  # Generate number of events for arm B
  ln_OR_ibc <- rnorm(n = k_bc, mean = OR_bc, sd = sqrt(tau2))
  pi_ib <- (pi_ic * exp(ln_OR_ibc)) / (1 - pi_ic + pi_ic * exp(ln_OR_ibc))
  e_ib <- sapply(1:k_bc, function(x) {rbinom(1, n_ib[x], pi_ib[x])})

  # ------------------------ AC comparison ----------------------------------#
  # Generate sample size
  n_i2 <- 2 * round(runif(n = k_ac, min = min_n, max = max_n) / 2)
  # sample size in arm C for BC comparison
  n_ic2 <- n_i2 / 2
  # sample size in arm B for AC comparison
  n_ia <- n_i2 / 2

  # Generate number of events for arm A
  pi_ic2 <- runif(n = k_ac, min = pi_c - (pi_c / 2), max = pi_c + (pi_c / 2))
  e_ic2 <- sapply(1:k_ac, function(x) {rbinom(1, n_ic2[x], pi_ic2[x])})

  # Generate number of events for arm C
  ln_OR_iac <- rnorm(n = k_ac, mean = OR_ac, sd = sqrt(tau2))
  pi_ia <- (pi_ic2 * exp(ln_OR_iac)) / (1 - pi_ic2 + pi_ic2 * exp(ln_OR_iac))
  e_ia <- sapply(1:k_ac, function(x) {rbinom(1, n_ia[x], pi_ia[x])})

  # ---------------------------- Data ---------------------------------------

  bc_dat <- cbind("c_event" = e_ic, "c_n" = n_ic,
                  "other_event" = e_ib, "other_n" = n_ib)
  ac_dat <- cbind("c_event" = e_ic2, "c_n" = n_ic2,
                  "other_event" = e_ia, "other_n" = n_ia)
  comp <- c(rep("bc", k_bc), rep("ac", k_ac))
  final_dat <- as.data.frame(cbind(comp, rbind(bc_dat, ac_dat)))
  return(final_dat)
}
```

Simulations R Code - Generate_NMA_Data.R:

```
source("nma.sim.data.R")
```

```
generate.dat <- function(k_bc, k_ac, min_n, max_n, prop_n_in_ctrl, pi_c,
                         OR_bc, OR_ac, tau2) {

  sdata <- nma.sim.data(k_bc, k_ac, min_n, max_n, prop_n_in_ctrl, pi_c,
                        OR_bc, OR_ac, tau2)
  bc_dat <- sdata[sdata$comp == "bc", ]
  ac_dat <- sdata[sdata$comp == "ac", ]
  bc_dat$c_event <- as.numeric(as.character(bc_dat$c_event))
  bc_dat$c_n <- as.numeric(as.character(bc_dat$c_n))
  bc_dat$other_event <- as.numeric(as.character(bc_dat$other_event))
  bc_dat$other_n <- as.numeric(as.character(bc_dat$other_n))
  ac_dat$c_event <- as.numeric(as.character(ac_dat$c_event))
  ac_dat$c_n <- as.numeric(as.character(ac_dat$c_n))
  ac_dat$other_event <- as.numeric(as.character(ac_dat$other_event))
  ac_dat$other_n <- as.numeric(as.character(ac_dat$other_n))
  # create column names for control group and treatment group (2 * num_studies)
  bc_colNamesTRTGrp <- c(rep(0, length(bc_dat$comp)),
                         rep(1, length(bc_dat$comp)))
  ac_colNamesTRTGrp <- c(rep(0, length(ac_dat$comp)),
                         rep(1, length(ac_dat$comp)))
  # 4 * num studies for how many cells in matrix we want
  bc_cells <- rep(0, 4 * length(bc_dat$comp))
  ac_cells <- rep(0, 4 * length(ac_dat$comp))
  # put ctrl event in every other spot in the first half of test
  # (starting with the first element)
  bc_cells[seq(1, length(bc_cells) / 2, 2)] <- bc_dat$c_event
  ac_cells[seq(1, length(ac_cells) / 2, 2)] <- ac_dat$c_event
  # put ctrl nonevent in every other spot in the first half of test
  # (starting with the second element)
  bc_cells[seq(2, length(bc_cells) / 2, 2)] <- bc_dat$c_n - bc_dat$c_event
  ac_cells[seq(2, length(ac_cells) / 2, 2)] <- ac_dat$c_n - ac_dat$c_event
  # put trt event in every other spot in the second half of test
  # (starting with the first element of the second half of test)
  bc_cells[seq((length(bc_cells) / 2) + 1,
               length(bc_cells), 2)] <- bc_dat$other_event
  ac_cells[seq((length(ac_cells) / 2) + 1,
               length(ac_cells), 2)] <- ac_dat$other_event
  # put trt nonevent in every other spot in the second half of test
```

```
  # (starting with the second element of the second half of test)
  bc_cells[seq((length(bc_cells) / 2) + 2,
               length(bc_cells), 2)] <- bc_dat$other_n - bc_dat$other_event
  ac_cells[seq((length(ac_cells) / 2) + 2,
               length(ac_cells), 2)] <- ac_dat$other_n - ac_dat$other_event

  bc_matrix <- matrix(bc_cells, nrow = 2, byrow = FALSE,
                      dimnames = list(Out = c(1, 0), TRT = bc_colNamesTRTGrp))
  ac_matrix <- matrix(ac_cells, nrow = 2, byrow = FALSE,
                      dimnames = list(Out = c(1, 0), TRT = ac_colNamesTRTGrp))
  return(list("bc" = bc_matrix, "ac" = ac_matrix))
}

scenario_num <- 1
num_reps <- 10000
k_bc_sim <- 5
min_n_sim <- 10
max_n_sim <- 50
pi_c_sim <- 0.10
OR_bc_sim <- 1
OR_ac_sim <- 1

if (scenario_num == 1) {
  k_ac_sim <- 5
  tau2_sim <- 0
  prop_n_in_ctrl_sim <- 1 / 2
} else if (scenario_num == 2) {
  k_ac_sim <- 5
  tau2_sim <- 0.4
  prop_n_in_ctrl_sim <- 1 / 2
} else if (scenario_num == 3) {
  k_ac_sim <- 5
  tau2_sim <- 0
  prop_n_in_ctrl_sim <- 1 / 4
} else if (scenario_num == 4) {
  k_ac_sim <- 5
  tau2_sim <- 0.4
  prop_n_in_ctrl_sim <- 1 / 4
} else if (scenario_num == 5) {
```

```
  k_ac_sim <- 10
  tau2_sim <- 0
  prop_n_in_ctrl_sim <- 1 / 2
} else if (scenario_num == 6) {
  k_ac_sim <- 10
  tau2_sim <- 0.4
  prop_n_in_ctrl_sim <- 1 / 2
} else if (scenario_num == 7) {
  k_ac_sim <- 10
  tau2_sim <- 0
  prop_n_in_ctrl_sim <- 1 / 4
} else if (scenario_num == 8) {
  k_ac_sim <- 10
  tau2_sim <- 0.4
  prop_n_in_ctrl_sim <- 1 / 4
}

set.seed(1993)
all_dat_full <- replicate(num_reps,
                          generate.dat(k_bc = k_bc_sim,
                                       k_ac = k_ac_sim,
                                       min_n = min_n_sim,
                                       max_n = max_n_sim,
                                       prop_n_in_ctrl = prop_n_in_ctrl_sim,
                                       pi_c = pi_c_sim,
                                       OR_bc = OR_bc_sim,
                                       OR_ac = OR_ac_sim,
                                       tau2 = tau2_sim))

# For BC comparison: if all studies have no events in treatment group,
# flag with a "1"
bc_zero_trt_events <- sapply(all_dat_full[1, ],
                             function(x) ifelse(all(x[1,
                                                      colnames(x) == 1] == 0),
                                                1, 0))
sum(bc_zero_trt_events)
# remove such studies
all_dat <- all_dat_full[, which(bc_zero_trt_events != 1)]
# For BC comparison: if all studies have no events in control group,
```

```
# flag with a "1"
bc_zero_ctrl_events <- sapply(all_dat[1, ],
                              function(x) ifelse(all(x[1,
                                                       colnames(x) == 0] == 0),
                                                 1, 0))
sum(bc_zero_ctrl_events)
# remove such studies
all_dat <- all_dat[, which(bc_zero_ctrl_events != 1)]

# For AC comparison: if all studies have no events in treatment group,
# flag with a "1"
ac_zero_trt_events <- sapply(all_dat[2, ],
                             function(x) ifelse(all(x[1,
                                                      colnames(x) == 1] == 0),
                                                1, 0))
sum(ac_zero_trt_events)
# remove such studies
all_dat <- all_dat[, which(ac_zero_trt_events != 1)]
# For AC comparison: if all studies have no events in control group,
# flag with a "1"
ac_zero_ctrl_events <- sapply(all_dat[2, ],
                              function(x) ifelse(all(x[1,
                                                       colnames(x) == 0] == 0),
                                                 1, 0))
sum(ac_zero_ctrl_events)
# remove such studies
all_dat <- all_dat[, which(ac_zero_ctrl_events != 1)]

rm(list = ls()[! ls() %in% c("all_dat", "scenario_num")])
save.image(paste0("Scenario", scenario_num, ".RData"))
```

Appendix E: Code for Chapter 4

This appendix contains selected code for Chapter 4, and includes the following:

- gmeta_functions.R
- Modification1.R
- Modification2.R
- Modification3.R
- Simulation_Modification1.R
- Simulation_Modification2.R
- Simulation_Modification3.R

For the simulations in Chapter 4, the data was generated using the ma.sim.dat.R and Generate_Data.R code in Appendix C. The file gmeta_functions.R contains selected functions taken directly from the *gmeta* R package. The files Modification1.R, Modification2.R, and Modification3.R contain modified code taken from the *gmeta* R package, and they are used in the files Simulation_Modification1.R, Simulation_Modification2.R, and Simulation_Modification3.R.

Simulations R Code - gmeta_functions.R:

```
gmeta.exact.indiv <- function(data_matrix, gmo.xgrid, ci.level) {
  # number of study
  K = nrow(data_matrix)
  # input  - individual study mean and standard deviation
  gmeta.theta = log((data_matrix[, 1] *
                       (data_matrix[, 4] - data_matrix[, 2])) /
                      (data_matrix[, 2] *
                         (data_matrix[, 3] - data_matrix[, 1])))
  gmeta.sigma = sqrt(1 / data_matrix[, 1] + 1 / data_matrix[, 2] +
                       1 / (data_matrix[, 3] - data_matrix[, 1]) + 1 /
                       (data_matrix[, 4] - data_matrix[, 2]))
  index = is.finite(gmeta.theta) & is.finite(gmeta.sigma)
  # output - individual study CDs
  indiv.cds = matrix(NA, K, length(gmo.xgrid))
  indiv.cis = matrix(NA, K, 2)
  indiv.medians = rep(NA, K)
  indiv.means = rep(NA, K)
  indiv.stddevs = rep(NA, K)
  # construct individual CDs
  alpha <- 1 - ci.level
  pfunc <- function(theta) {
    return(1 - pFNCHypergeo.wrap(data_matrix[i, 1],
                                 data_matrix[i, 3],
                                 data_matrix[i, 4],
                                 data_matrix[i, 5],
                                 theta) + dFNCHypergeo(data_matrix[i, 1],
                                                       data_matrix[i, 3],
                                                       data_matrix[i, 4],
                                                       data_matrix[i, 5],
                                                       theta) / 2)
  }
  for (i in 1:K) {
    indiv.cds[i, ] <- sapply(exp(gmo.xgrid), pfunc)
  }
  # otherwise qnorm() will resolve Inf.
  indiv.cds <- ifelse(indiv.cds < 1 - 0.1 ^ 12, indiv.cds, 1 - 0.1 ^ 12)
  # individual study - CI median
  for (i in 1:K) {
```

```
    indiv.cis[i, ] <- log(c(.quantileCD(pfunc, alpha / 2),
                            .quantileCD(pfunc, 1 - alpha / 2)))
    indiv.medians[i] <- log(.quantileCD(pfunc, 0.5))
  }
  # individual study - mean and standard deviation
  na.moment <- is.na(gmeta.theta * 0)
  for (i in c(1:K)[!na.moment]) {
    indiv.means[i] <- gmeta.cd.mean(gmo.xgrid, indiv.cds[i,])
    indiv.stddevs[i] <- gmeta.cd.stddev(gmo.xgrid, indiv.cds[i,])
  }
  indiv.means[na.moment] <- gmeta.theta[na.moment]
  indiv.stddevs[na.moment] <- gmeta.sigma[na.moment]
  # return
  gmeta.data <- list(data_matrix   = data_matrix,
                     # individual CDs
                     indiv.cds     = indiv.cds,
                     indiv.cis     = indiv.cis,
                     indiv.medians = indiv.medians,
                     indiv.means   = indiv.means,
                     indiv.stddevs = indiv.stddevs,
                     # output gridding points
                     x.grids       = gmo.xgrid)
  return(gmeta.data)
}

##### overall test size error for exact1 method
test.size.error <- function(s,
                            n.vec,
                            p1.vec,
                            m.vec,
                            p0.vec,
                            weight.vec,
                            result.1minusS = FALSE,
                            mc.iteration = 1000000) {
  # number of study
  K = length(n.vec)
  psi = (p1.vec[1] / (1 - p1.vec[1])) / (p0.vec[1] / (1 - p0.vec[1]))
  # generate p.i(\psi), i=1,2,...K.
  # each column is for an i
```

```
mid.p.sample     = matrix(NA, nrow = mc.iteration, ncol = K)
# each column is for an i
mid.p.sample.adj = matrix(NA, nrow = mc.iteration, ncol = K)
for (i in 1:K) {
  x = rbinom(mc.iteration, n.vec[i], p1.vec[i])
  y = rbinom(mc.iteration, m.vec[i], p0.vec[i])
  mid.p.sample[, i] = midp.oddsratio(x, n.vec[i], m.vec[i], x + y, or = psi)
  mid.p.sample.adj[, i] = adjust.beta(mid.p.sample[, i],
                                      n = n.vec[i],
                                      pt = p1.vec[i],
                                      m = m.vec[i],
                                      pc = p0.vec[i])
}
# uniform sample
uniform.sample = matrix(runif(mc.iteration * K, 0, 1),
                        nrow = mc.iteration,
                        ncol = K)
# quantile
qnorm.mid.p.sample     = qnorm(mid.p.sample)
qnorm.mid.p.sample.adj = qnorm(mid.p.sample.adj)
qnorm.uniform.sample   = qnorm(uniform.sample)
#
bb1 = sapply(1:K, function(i) {
  ww <- weight.vec / weight.vec[i]
  ww[1:K <= i] <- 0
  ww
})
bb2 = sapply(1:K, function(i) {
  ww <- weight.vec / weight.vec[i]
  ww[1:K >= i] <- 0
  ww
})
#
xoffset = sqrt(sum(weight.vec ^ 2)) / weight.vec * qnorm(s)
#
main     = qnorm.mid.p.sample     %*% bb1 + qnorm.uniform.sample %*% bb2
main.adj = qnorm.mid.p.sample.adj %*% bb1 + qnorm.uniform.sample %*% bb2
#
aa     = t(xoffset - t(main))
```

```
aa.adj = t(xoffset - t(main.adj))
#
pnorm.aa     = pnorm(aa)
pnorm.aa.adj = pnorm(aa.adj)
#
dd = sapply(1:K, FUN = function(i) {
  findInterval(pnorm.aa[, i],
               sort(mid.p.sample[, i])) / mc.iteration - pnorm.aa[, i]
})
dd.adj = sapply(1:K, FUN = function(i) {
    findInterval(pnorm.aa.adj[, i],
                 sort(mid.p.sample.adj[, i])) /
    mc.iteration - pnorm.aa.adj[, i]
  })
# calculate test.size.error
test.size.error     = sum(colMeans(dd))
test.size.error.adj = sum(colMeans(dd.adj))
# calculate test.size.error.1minusS
test.size.error.1minusS     = 'Not requested'
test.size.error.adj.1minusS = 'Not requested'
if (result.1minusS) {
  # use 1 - s instead of s
  xoffset = sqrt(sum(weight.vec ^ 2)) / weight.vec * qnorm(1 - s)
  #
  aa      = t(xoffset - t(main))
  aa.adj  = t(xoffset - t(main.adj))
  #
  pnorm.aa     = pnorm(aa)
  pnorm.aa.adj = pnorm(aa.adj)
  #
  dd = sapply(1:K, FUN = function(i) {
      findInterval(pnorm.aa[, i],
                   sort(mid.p.sample[, i])) / mc.iteration - pnorm.aa[, i]
  })
  dd.adj = sapply(1:K, FUN = function(i) {
    findInterval(pnorm.aa.adj[, i],
                 sort(mid.p.sample.adj[, i])) /
      mc.iteration - pnorm.aa.adj[, i]
  })
```

```
    # calculate test.size.error.1minusS
    test.size.error.1minusS     = sum(colMeans(dd))
    test.size.error.adj.1minusS = sum(colMeans(dd.adj))
  }
  # return
  return(list(Test.Size.Error                  = test.size.error,
              Test.Size.Error.Adjusted         = test.size.error.adj,
              Test.Size.Error.1minusS          = test.size.error.1minusS,
              Test.Size.Error.Adjusted.1minusS = test.size.error.adj.1minusS))
}

## meta-analysis - combine evidence from 2x2 tables[done]

# unified meta-analysis[done]

# postprocessing

# ****************************************************************************
#    # postprocessing - print, summary, plot, etc..
# ****************************************************************************
# print, summary, plot, etc.

## print&summary [gmeta.e]
###print
print.gmeta.e <- function(x, ...) {
  # Title
  cat('\t\tExact Meta-Analysis Approach through CD-Framework\n')
  # Call
  cat('\nCall:\n')
  print(x$call) #[*$call]: type of language
  # Results
  cat('\nSummary of Combined CD:\n')
  cmbd.cd.summary = data.frame(
    mean   = format(round(x$combined.mean, 4), nsmall = 4),
    median  = format(round(x$combined.median, 4), nsmall = 4),
    standard.deviation = format(round(x$combined.sd, 4), nsmall = 4)
  )
  row.names(cmbd.cd.summary) <- 'Combined CD'
  print(cmbd.cd.summary)
```

```
  # Details
  cat('\nCombined Confidence Distribution:\n')
  # construct combined CD
  cmbd.cd <- data.frame(x = x$x.grids,
                        density = x$combined.density,
                        probability = x$combined.cd)
  # set lower/upper bound
  xl  <- min(x$x.grids)
  xu  <- max(x$x.grids)
  # count
  n.grids <- sum(cmbd.cd$x >= xl & cmbd.cd$x <= xu)
  # print head/tail
  if (n.grids <= 10) {
    print(cmbd.cd) # simple print all x-cd-points
  } else {
    cmbd.cdhead <- cmbd.cd[1:5, ]
    cmbd.cdtail <- cmbd.cd[(n.grids - 5):n.grids, ]
    names(cmbd.cdtail) <- c(' ', ' ', ' ') # avoid duplicate titles
    # print head/tail 5 x-cd-points
    # output
    print(cmbd.cdhead)
    cat('\t...\n\t...\n')
    print(cmbd.cdtail)
    #cat('\n           \n')
  }
}
###summary
summary.gmeta.e <- function(object, ...) {
  # set
  object.sry <- list()
  # set Call
  object.sry$call <- object$call
  # set combined CD
  object.sry$cmbd <- data.frame(mean     = object$combined.mean,
                                median   = object$combined.median,
                                stddev   = object$combined.sd,
                                ci.lower = object$combined.ci[1],
                                ci.upper = object$combined.ci[2])
  row.names(object.sry$cmbd) <- 'Combined CD'
```

```
  # set individual CDs
  object.sry$idiv <- data.frame(mean     = object$individual.means,
                                median   = object$individual.medians,
                                stddev   = object$individual.stddevs,
                                ci.lower = object$individual.cis[, 1],
                                ci.upper = object$individual.cis[, 2])
  # set individual study names
  row.names(object.sry$idiv) <- object$study.names
  # set ci.level
  object.sry$ci.level <- object$ci.level
  # set number of study
  object.sry$n.study <- dim(object.sry$idiv)[1]
  # set class
  class(object.sry) <- 'summary.gmeta.e'
  # return
  return(object.sry)
}
###print of summary
print.summary.gmeta.e <- function(x, ...) {
  # Title
  cat('\t\tExact Meta-Analysis Approach through CD-Framework\n')
  # Call
  cat('\nCall:\n')
  print(x$call) #[*$call]: type of language
  # Results
  cat('\nSummary of Combined CD:\n')
  print(x$cmbd)
  cat('\nConfidence level =', x$ci.level, '\n')
  # Details
  cat('\nSummary of Individual CDs:\n')
  print(x$idiv)
  cat('\nConfidence level =', x$ci.level, '\n')
}

## plot functions
plot.gmeta <- function(gmo,
                       studies = NULL,
                       plot.option = c('confidence-density',
                                       'confidence-curve',
```

```
                                        'cv',
                                        'confidence-distribution',
                                        'cdf'),
                        type = 'l',
                        xlab = 'x',
                        ylab = 'density',
                        xlim = NULL,
                        ylim = NULL,
                        ...) {
  UseMethod("plot", gmo)

}

###plot functions - combine evidence from 2x2 tables
plot.gmeta.e <- function(x,
                         studies = NULL,
                         plot.option = c('confidence-density',
                                         'confidence-curve',
                                         'cv',
                                         'confidence-distribution',
                                         'cdf'),
                         type = 'l',
                         xlab = 'x',
                         ylab = 'confidence density',
                         xlim = NULL,
                         ylim = NULL,
                         ...) {
  # match plot.option
  mfplot <- match.call()
  plot.option = match.arg(plot.option)

  # plot.individual.studies
  # if ( is.null(studies) ) {
  # # take all studies
  # gmi     <- x$input
  # nn1     <- ifelse(is.list(gmi), length(gmi), dim(gmi)[1])
  # studies <- c(1:nn1)
  # }
  # take only non-zero/zero-event studies
```

```
  idx     <- studies[is.na(x$individual.mean)[studies]]
  studies <- studies[!is.na(x$individual.mean)[studies]]
  if (length(idx) != 0) {
    for (i in idx) {
      cat('The confidence distribution of study', i,
        'cannot be plot because it contains zero-zero events.',
        '\n')
    }
  }
  # plot via plot.option
  if (plot.option == 'confidence-density') {
    gmeta.plot.cdd(x, studies, type, xlab, ylab, xlim, ylim, ...)
  } else if (plot.option == 'cv' ||
             plot.option == 'confidence-curve') {
    if (ylab == 'confidence density') {
      ylab = 'confidence curve'
    }
    gmeta.plot.cvs(x, studies, type, xlab, ylab, xlim, ylim, ...)
  } else if (plot.option == 'cdf' ||
             plot.option == 'confidence-distribution') {
    if (ylab == 'confidence density') {
      ylab = 'confidence distribution'
    }
    gmeta.plot.cdf(x, studies, type, xlab, ylab, xlim, ylim, ...)
  } else {
    stop('plot.option not recognize')
  }
}

###plot functions - shared by model based and exact methods on 2x2 tables
#####
gmeta.plot.cdd <-
  function(x, studies, type, xlab, ylab, xlim, ylim, ...) {
    #e <- studies
    # number of layers
    nn <- length(studies) + 1 # individual studies + combined CD
    # extract study names
    enames <- c(x$study.names[studies], 'combined.density')
    # x.grids
```

```
x.grids <- x$x.grids
# individual CDs
ecds <- x$individual.cds[studies, ]
# individual densities
edns <- NULL
for (i in studies) {
  edns <- rbind(edns, F2f(x.grids, x$individual.cds[i, ]))
}
# individual and combined density
edns <- rbind(edns, x$combined.density)
# unified y-range on all layers for visual comparison
if (!is.null(studies)) {
  if (max(edns, na.rm = T) > 1) {
    edns <- edns / max(edns, na.rm = T)
  }
}
# individual and combined medians
mdn <- c(x$individual.medians[studies], x$combined.median)
# individual and combined confidence intervals
emdncis <- rbind(x$individual.cis[studies,], x$combined.ci)
# set lower/upper bound
if (!is.null(xlim)) {
  xl <- min(xlim)
  xu <- max(xlim)
} else {
  xl <- min(x.grids)
  xu <- max(x.grids)
}
# set study.names position
xnames <- xl
# plot
if (!is.null(studies)) {
  plot(
    0,
    type = 'n',
    xlab = xlab,
    ylab = ylab,
    xlim = c(xl, xu),
    ylim = c(0, nn),
```

```
      yaxt = 'n'
    )
  } else {
    plot(
      0,
      type = 'n',
      xlab = xlab,
      ylab = ylab,
      xlim = c(xl, xu),
      ylim = c(0, ceiling(max(edns))),
      yaxt = 'n'
    )
  }
  for (i in 1:nn) {
    # supporting line
    abline(h = i - 1, lty = 2)
    # plot of density
    lines(x.grids, edns[i, ] + i - 1, lty = 1)
    # mark median, ci.lower, ci.upper
    points(mdn[i], i - 1, cex = 1, col = 'dark red')
    points(emdncis[i, 1],
           i - 1,
           pch = '[',
           cex = 1,
           col = 'dark red')
    points(emdncis[i, 2],
           i - 1,
           pch = ']',
           cex = 1,
           col = 'dark red')
    # legend study.names
    legend(xnames,
           i - 0.5,
           legend = paste('#', enames[i], sep = ''),
           bty = 'n')
  }
  # overall median
  abline(v = x$combined.median, lty = 2)
}
```

```
#####
gmeta.plot.cvs <-
  function(gmo,
           studies,
           type,
           xlab,
           ylab,
           xlim,
           ylim,
           ...) {
    # number of layers
    nn <- length(studies) + 1 # individual studies + combined CD
    # extract study names
    enames <- c(gmo$study.names[studies], 'combined.cv')
    # x.grids
    x.grids <- gmo$x.grids

    # construct individual and combined CVs
    combined.cv    <- 1 - 2 * abs(gmo$combined.cd - 0.5)
    individual.cvs <- 1 - 2 * abs(gmo$individual.cds - 0.5)
    # set cvs
    ecvs <- rbind(individual.cvs[studies,], combined.cv)

    # individual and combined medians
    mdn <- c(gmo$individual.medians[studies], gmo$combined.median)
    # individual and combined confidence intervals
    emdncis <- rbind(gmo$individual.cis[studies,], gmo$combined.ci)
    # set lower/upper bound
    if (!is.null(xlim)) {
      xl <- min(xlim)
      xu <- max(xlim)
    } else {
      xl <- min(x.grids)
      xu <- max(x.grids)
    }
    # set study.names position
    xnames <- xl
    # plot
    if (!is.null(studies)) {
```

```
  plot(
    0,
    type = 'n',
    xlab = xlab,
    ylab = ylab,
    xlim = c(xl, xu),
    ylim = c(0, nn),
    yaxt = 'n'
  )
} else {
  plot(
    0,
    type = 'n',
    xlab = xlab,
    ylab = ylab,
    xlim = c(xl, xu),
    ylim = c(0, ceiling(max(ecvs))),
    yaxt = 'n'
  )
}
for (i in 1:nn) {
  # supporting line
  abline(h = i - 1, lty = 2)
  # plot of density
  lines(x.grids, ecvs[i, ] + i - 1, lty = 1)
  # mark median, ci.lower, ci.upper
  points(mdn[i], i - 1, cex = 1, col = 'dark red')
  points(emdncis[i, 1],
         i - 1,
         pch = '[',
         cex = 1,
         col = 'dark red')
  points(emdncis[i, 2],
         i - 1,
         pch = ']',
         cex = 1,
         col = 'dark red')
  # legend study.names
  legend(xnames,
```

```
            i - 0.5,
            legend = paste('#', enames[i], sep = ''),
            bty = 'n')
    }
    # overall median
    abline(v = gmo$combined.median, lty = 2)
  }
#####
gmeta.plot.cdf <-
  function(gmo,
           studies,
           type,
           xlab,
           ylab,
           xlim,
           ylim,
           ...) {
    #e <- studies
    # number of layers
    nn <- length(studies) + 1 # individual studies + combined CD
    # extract study names
    enames <- c(gmo$study.names[studies], 'combined.cdf')
    # x.grids
    x.grids <- gmo$x.grids
    # individual CDs
    ecds <- gmo$individual.cds[studies, ]
    # individual and combined CDs
    ecds <- rbind(ecds, gmo$combined.cd)
    # unified y-range on all layers for visual comparison
    if (!is.null(studies)) {
      if (max(ecds, na.rm = T) > 1) {
        ecds <- ecds / max(ecds, na.rm = T)
      }
    }
    # individual and combined medians
    mdn <- c(gmo$individual.medians[studies], gmo$combined.median)
    # individual and combined confidence intervals
    emdncis <- rbind(gmo$individual.cis[studies,], gmo$combined.ci)
    # set lower/upper bound
```

```
if (!is.null(xlim)) {
  xl <- min(xlim)
  xu <- max(xlim)
} else {
  xl <- min(x.grids)
  xu <- max(x.grids)
}
# set study.names position
xnames <- xl
# plot
if (!is.null(studies)) {
  plot(
    0,
    type = 'n',
    xlab = xlab,
    ylab = ylab,
    xlim = c(xl, xu),
    ylim = c(0, nn),
    yaxt = 'n'
  )
} else {
  plot(
    0,
    type = 'n',
    xlab = xlab,
    ylab = ylab,
    xlim = c(xl, xu),
    ylim = c(0, ceiling(max(ecds))),
    yaxt = 'n'
  )
}
for (i in 1:nn) {
  # supporting line
  abline(h = i - 1, lty = 2)
  # plot of CDs
  lines(x.grids, ecds[i, ] + i - 1, lty = 1)
  # mark median, ci.lower, ci.upper
  points(mdn[i], i - 1, cex = 1, col = 'dark red')
  points(emdncis[i, 1],
```

```
            i - 1,
            pch = '[',
            cex = 1,
            col = 'dark red')
      points(emdncis[i, 2],
            i - 1,
            pch = ']',
            cex = 1,
            col = 'dark red')
      # legend study.names
      legend(xnames,
            i - 0.5,
            legend = paste('#', enames[i], sep = ''),
            bty = 'n')
    }
    # overall median
    abline(v = gmo$combined.median, lty = 2)
  }

# postprocessing [done]


# ****************************************************************************
#    other functions used in this package
# ****************************************************************************
# other functions used in this package

### using CD to make inference
### get mean, median, sd, ci, given cd, used everywhere
##### get mean from cd
gmeta.cd.mean <- function(x, cd) {
  x  =  x[!is.na(cd)]
  cd = cd[!is.na(cd)]
  if (length(unique(cd)) == 1) {
    mn <- NA
  } else if (length(x) == length(cd)) {
    dn <- F2f(x, cd) # density
    mn <- sum(x * dn, na.rm = T) / sum(dn, na.rm = T) # mean
  } else {
```

```
    stop('length of x is not the same as length of cd.')
  }
  return(mn)
}
##### get median from cd
gmeta.cd.median <- function(x, cd) {
  x  =  x[!is.na(cd)]
  cd = cd[!is.na(cd)]
  if (length(unique(cd)) == 1) {
    mdn <- NA
  } else {
    # same as approx in
    # gmeta.cd.mdncis()
    x1 <- rev(x[cd < 0.5])[1]
    x2 <- x[cd > 0.5][1]
    y1 <- rev(cd[cd < 0.5])[1]
    y2 <- cd[cd > 0.5][1]
    # linear interpolation
    mdn <- x1 + (x2 - x1) / (y2 - y1) * (0.5 - y1)
  }
  return(mdn)
}
##### get stddev from cd
gmeta.cd.stddev <- function(x, cd) {
  x  =  x[!is.na(cd)]
  cd = cd[!is.na(cd)]
  if (length(unique(cd)) == 1) {
    sdv <- NA
  } else {
    sdv <-
      diff(approx(
        x = cd,
        y = x,
        xout = c(0.25, 0.75),
        ties = 'mean'
      )$y) / (qnorm(0.75) - qnorm(0.25))
  }
  return(sdv)
}
```

```
##### get ci&median from cd
gmeta.cd.mdncis <- function(x, cd, alpha) {
  x  =  x[!is.na(cd)]
  cd = cd[!is.na(cd)]
  if (length(unique(cd)) == 1) {
    mdncis <- c(NA, NA, NA)
  } else {
    mdncis <-
      approx(
        x = cd,
        y = x,
        xout = c(alpha / 2, 0.5 , 1 - alpha / 2),
        ties = 'mean'
      )$y
  }
  return(mdncis)
}

### computing density(pdf) from distribution(cdf), used everywhere
### mainly used for getting CD density from CD distribution,
### NA on two end to avoid sudden jump
F2f <- function (x, Fx) {
  fx = diff(Fx, lag = 2) / diff(x, lag = 2)
  fx = c(NA, fx, NA) # match the length of x-fx-cdf
  # n.xgrids
  n = length(fx)
  # liner extrapolation
  fx[1] = max(0, (fx[2] - fx[3]) / (x[2] - x[3]) * (x[1] - x[2]) + fx[2])
  fx[n] = max(0,
              (fx[n - 1] - fx[n - 2]) /
                (x[n - 1] - x[n - 2]) * (x[n] -
                                           x[n - 1]) + fx[n - 1])
  # return
  return(fx)
  #[update in v2.0: previously #return(c(NA, fx, NA))
  # NA so that the edge of the plot of density is right.]
}
```

```
## [exact methods tools]

### compute quantile given cd function, used in exact method
.quantileCD <- function(CDF, prbblty) {
  ff <- function(theta) {
    CDF(theta) - prbblty
  }
  for (power.l in 0:15) {
    if (ff(10 ^ (-power.l)) < 0) {
      break
    }
  }
  for (power.u in 0:15) {
    if (ff(10 ^ (power.u)) > 0) {
      break
    }
  }
  if (power.l == 15) {
    qntl = 0
  } else if (power.u == 15) {
    qntl = Inf
  } else {
    qntl = uniroot(
      f = function(theta) {
        CDF(theta) - prbblty
      },
      interval = c(10 ^ (-power.l), 10 ^ (power.u)),
      tol = 0.001
    )$root
  }
  return(qntl)
}

### compute cd function (based on p-value functions) using exact method
midp.oddsratio <- function(x, N, M, t, or) {
  # x and t can be vectors
  mid.p = rep(NA, length(x))
  for (i in min(t):max(t)) {
    index = which(t == i)
```

```
    if (!identical(index, integer(0))) {
      for (idx in index) {
        mid.p[idx] = 1 - pFNCHypergeo.wrap(x[idx], N, M, i, or) +
          dFNCHypergeo(x[idx], N, M, i, or) / 2
      }
    } else {
      #cat('\nindex null, mid.p all NA\n')
    }
  }
  return(mid.p)
}

### conditional MLE, used in excat method
.conditionalMLE <- function(data_matrix) {
  # number of study
  K = dim(data_matrix)[1]
  # or.hat.CMLE
  if (sum(data_matrix[, 1]) == 0) {
    or.hat.CMLE = 0
  } else if (sum(data_matrix[, 2]) == 0) {
    or.hat.CMLE = Inf
  } else {
    .estimatingFunctionCMLE <- function(theta) {
      summation = 0
      for (j in 1:K) {
        expectation = 0
        for (x in max(0, data_matrix[j, 1] + data_matrix[j, 2] -
                      data_matrix[j, 4]):min(data_matrix[j, 3],
                                             data_matrix[j, 1] +
                                             data_matrix[j, 2])) {
          expectation = expectation + x * dFNCHypergeo(x,
                                                       data_matrix[j, 3],
                                                       data_matrix[j, 4],
                                                       data_matrix[j, 1] +
                                                         data_matrix[j, 2],
                                                       theta)
        }
        summation = summation + expectation - data_matrix[j, 1]
      }
```

```
      return(summation)
    }
    or.hat.CMLE = uniroot(.estimatingFunctionCMLE, c(0.0001, 10000),
                          tol = 0.001)$root
  }
  return(or.hat.CMLE)
}

### calculate estimates of event rates and weights using empirical
### Bayes approach please refer to Efron 1996 JASA for details,
### functions are used in exact method
.Integrate <-
  function(f) {
    # input a function with range between 0 and 1
    ss1 = seq(0.0001, 0.1, 0.0001)
    ss2 = seq(0.101, 0.999, 0.001)
    return(sum(f(ss1)) * 0.0001 + sum(f(ss2)) * 0.001)
  }
#####
.Estimates <- function(data_matrix) {
  K = dim(data_matrix)[1]

  x = data_matrix[, 1]
  y = data_matrix[, 2]
  n = data_matrix[, 3]
  m = data_matrix[, 4]

  if (sum(x) + sum(y) == 0) {
    # in the case of all zeros in two arms
    p0.hat  = NA
    p1.hat  = NA
    psi.hat = NA
    weight.hat = rep(1, K)
  } else if (sum(y) == 0) {
    # in the case of all zeros in the control arm
    p0.hat  = NA
    p1.hat  = rep(sum(x) / sum(n), K)
    psi.hat = Inf
    weight.hat = sqrt(m * p1.hat / (1 - p1.hat))
```

```
} else if (sum(x) == 0) {
  # in the case of all zeros in the treatment arm
  p0.hat = .Estimates.1arm(y, m)
  p1.hat = NA
  psi.hat = 0
  weight.hat = sqrt(n * p0.hat / (1 - p0.hat))
} else {
  # use moment estimates as initial values
  mu = mean(y / m)
  v.square = var(y / m)
  beta1.initial = mu * (mu * (1 - mu) / v.square - 1)
  beta2.initial = beta1.initial * (1 - mu) / mu
  psi.initial   = .conditionalMLE(data_matrix)
  # use optim() to obtain parameters estimates
  parameter.initials  = c(log(beta1.initial),
                          log(beta2.initial),
                          log(psi.initial))
  .marginalLikelihood <- function(parameters) {
    beta1 = exp(parameters[1])
    beta2 = exp(parameters[2])
    psi   = exp(parameters[3])
    ff = 0
    for (i in 1:K) {
      .individualLikelihood <- function(p0) {
        p1 = psi * p0 / (1 - p0 + psi * p0)
        return(dbinom(x[i], n[i], p1) * dbinom(y[i], m[i], p0) *
                 dbeta(p0, beta1, beta2))
      }
      ff = ff + log(.Integrate(.individualLikelihood))
    }
    return(-ff)
  }
  parameter.estimates = optim(
    parameter.initials,
    .marginalLikelihood,
    method = 'Nelder-Mead',
    lower = -Inf,
    upper = Inf
  )$par
```

```
    # resolve parameters estimates
    beta1.hat = exp(parameter.estimates[1])
    beta2.hat = exp(parameter.estimates[2])
    psi.hat   = exp(parameter.estimates[3])
    # update
    p0.hat = rep(NA, K)
    for (i in 1:K) {
      .individualLikelihoodEmpirical <- function(p0) {
        p1 = psi.hat * p0 / (1 - p0 + psi.hat * p0)
        return(dbinom(x[i], n[i], p1) * dbinom(y[i], m[i], p0) *
                  dbeta(p0, beta1.hat, beta2.hat))
      }
      denominator <- .Integrate(.individualLikelihoodEmpirical)
      nominator   <-
        .Integrate(
          f = function(p0) {
            p0 * .individualLikelihoodEmpirical(p0)
          }
        )
      p0.hat[i]   <- nominator / denominator
    }
    p1.hat = psi.hat * p0.hat / (1 - p0.hat + psi.hat * p0.hat)
    weight.hat = 1 / sqrt((n * p1.hat * (1 - p1.hat)) ^ (-1) +
                            (m * p0.hat *
                               (1 - p0.hat)) ^ (-1))
  }
  # return
  return(list(
    or.hat = psi.hat,
    pt.hat = p1.hat,
    pc.hat = p0.hat,
    weight.hat = weight.hat
  ))
}
#####
.Estimates.1arm <- function(y, m) {
  K = length(m)
  # initial
  mu = mean(y / m)
```

```
  v.square = var(y / m)
  beta1.initial = mu * (mu * (1 - mu) / v.square - 1)
  beta2.initial = beta1.initial * (1 - mu) / mu
  # optim()
  parameter.initials  = c(log(beta1.initial), log(beta2.initial))
  .marginalLikelihood <- function(parameters) {
    beta1 = exp(parameters[1])
    beta2 = exp(parameters[2])
    ff = 0
    for (i in 1:K) {
      ff = ff + lbeta(y[i] + beta1, m[i] - y[i] + beta2) - lbeta(beta1, beta2)
    }
    return(-ff)
  }
  parameter.estimates = optim(parameter.initials, .marginalLikelihood)$par
  # update
  beta1.hat = exp(parameter.estimates[1])
  beta2.hat = exp(parameter.estimates[2])
  p.hat = (y + beta1.hat) / (m + beta1.hat + beta2.hat)
  # return
  return(p.hat)
}

### mixed beta adjustment, used in exact method
adjust.beta <- function(x, n, pt, m, pc) {
  ifelse(x < 1 / 2, pbeta(x,
                          1 + 1 / (2 * m * pc * (1 - pc)),
                          1 + 1 / (2 * m * pc * (1 - pc))),
         pbeta(x, 1 + 1 / (2 * n * pt * (1 - pt)),
               1 + 1 / (2 * n * pt * (1 - pt))))
}

### update on 2012.11.26
### pFNCHypergeo gives NaN when odds extremely large which results a
### probability very close to 0 or 1 and pFNCHypergeo can not handle it.
### wrap pFNCHypergeo in pFNCHypergeo.wrap to recursively adjust odds
### when it results NaN
pFNCHypergeo.wrap <-
  function(x,
```

```
            m1,
            m2,
            n,
            odds,
            precision = 1E-7,
            lower.tail = TRUE) {
    prbblty = pFNCHypergeo(
      x = x,
      m1 = m1,
      m2 = m2,
      n = n,
      odds = odds,
      precision = precision,
      lower.tail = lower.tail
    )
    while (is.na(prbblty)) {
      odds = odds / 10
      prbblty = pFNCHypergeo(
        x = x,
        m1 = m1,
        m2 = m2,
        n = n,
        odds = odds,
        precision = precision,
        lower.tail = lower.tail
      )
    }
    return(prbblty)
  }
## [exact methods tools - done]
# other functions used in this package [done]
```

Simulations R Code - Modification1.R:

```
##############################################################################
#
#     Use Logit instead of Normal CDF in Exact 1 Method - without weights
#
##############################################################################
library(binom)
library(BiasedUrn)
source("gmeta_code_functions_to_load.R")

# input matrix of 2x2 tables in format c(rx-event_treatment_group,
# nx-number_observation_treatment_group, ry-event_control_group,
# ny-number_observation_treatment_group) for each row of the matrix
# edata = as.data.frame(matrix(c(3, 2, 4, 5, 1, 2, 4, 6), nrow = 2, ncol = 4))
# colnames(edata) = c("x", "n", "y", "m")


# ****************************************************************************
#    main
# ****************************************************************************

# main function
gmeta <- function(gmi,
                  gmi.type = c('pivot', 'cd', 'pvalue', '2x2'),
                  method = c(
                    'fixed-mle',
                    'fixed-robust1',
                    'fixed-robust2',
                    'fixed-robust2(sqrt12)',
                    'random-mm',
                    'random-reml',
                    'random-tau2',
                    'random-robust1',
                    'random-robust2',
                    'random-robust2(sqrt12)',
                    'fisher',
                    'normal',
                    'stouffer',
                    'min',
```

```
                    'tippett',
                    'max',
                    'sum',
                    'MH',
                    'Mantel-Haenszel',
                    'Peto',
                    'exact1',
                    'exact2'),
                  linkfunc = c('inverse-normal-cdf', 'inverse-laplace-cdf'),
                  weight = NULL,
                  study.names = NULL,
                  gmo.xgrid = NULL,
                  ci.level = 0.95,
                  tau2 = NULL,
                  mc.iteration = 10000,
                  eta = 'Inf',
                  verbose = FALSE,
                  report.error = FALSE) {
  UseMethod('gmeta')
}

# main function
gmeta.default <- function(gmi,
                          gmi.type = c('pivot',
                                       'cd',
                                       'pvalue',
                                       '2x2'),
                          method   = c(
                            'fixed-mle',
                            'fixed-robust1',
                            'fixed-robust2',
                            'fixed-robust2(sqrt12)',
                            'random-mm',
                            'random-reml',
                            'random-tau2',
                            'random-robust1',
                            'random-robust2',
                            'random-robust2(sqrt12)',
                            'fisher',
```

```
                          'normal',
                          'stouffer',
                          'min',
                          'tippett',
                          'max',
                          'sum',
                          'MH',
                          'Mantel-Haenszel',
                          'Peto',
                          'exact1',
                          'exact2'
                        ),
                        linkfunc = c('inverse-normal-cdf',
                                     'inverse-laplace-cdf'),
                        weight   = NULL,
                        study.names = NULL,
                        gmo.xgrid = NULL,
                        ci.level = 0.95,
                        tau2     = NULL,
                        mc.iteration = 10000,
                        eta      = 'Inf',
                        verbose  = FALSE,
                        report.error = FALSE) {
#check
mf           <- match.call()
ci.level  <-
  ifelse(is.numeric(ci.level) &&
           (ci.level >= 0) && (ci.level <= 1),
         ci.level,
         0.95)
mc.iteration <-
  round(ifelse(
    is.numeric(mc.iteration) && (mc.iteration > 1),
    mc.iteration,
    10000))
# meta-analysis
if (gmi.type == 'pvalue') {
  gmo <- gmeta.p(gmi, method)
} else if (gmi.type == 'cd' || gmi.type == 'pivot') {
```

```
  gmo <-
    gmeta.m(gmi,
            gmi.type,
            method,
            linkfunc,
            weight,
            gmo.xgrid,
            ci.level,
            tau2,
            verbose)
} else if (gmi.type == '2x2') {
  # combine 2x2 tables
  gmo <-
    gmeta.e(gmi,
            method,
            weight,
            gmo.xgrid,
            ci.level,
            mc.iteration,
            eta,
            verbose,
            report.error)
} else {
  stop("gmi.type must be 'cd', 'pivot', 'pvalue' or '2x2'.")
}
# post processing
gmo$call   <- match.call()
gmo$input  <- gmi
gmo$alpha  <- 1 - ci.level
gmo$study.names <- study.names
# registr S3 class
if (gmi.type == 'pvalue') {
  class(gmo) <- c('gmeta.p', 'gmeta')
} else if (gmi.type == 'cd' || gmi.type == 'pivot') {
  class(gmo) <- c('gmeta.m', 'gmeta')
} else if (gmi.type == '2x2') {
  class(gmo) <- c('gmeta.e', 'gmeta')
} else {
  stop("gmi.type must be 'cd', 'pivot', 'pvalue' or '2x2'.")
```

```
  }
  # return
  return(gmo)
}

## meta-analysis - combine evidence from 2x2 tables
# ****************************************************************************
#    gmeta.e() - combine evidence from 2x2 tables (exact methods)
# ****************************************************************************
## main[gmeta.e]
gmeta.e <-
  function(gmi,
           method,
           weight,
           gmo.xgrid,
           ci.level,
           mc.iteration,
           eta,
           verbose,
           report.error) {
    data_matrix = as.matrix(cbind(gmi[, 1], gmi[, 3], gmi[, 2], gmi[, 4],
                                  gmi[, 1] + gmi[, 3]))
    colnames(data_matrix) = c("x", "y", "N", "M", "x+y")
    if (method == 'exact1') {
      gmeta.data <- gmeta.exact.indiv(data_matrix, gmo.xgrid, ci.level)
      gmeta.cmbd <-
        gmeta.exact.combine(gmeta.data,
                            weight,
                            gmo.xgrid,
                            ci.level,
                            mc.iteration,
                            report.error)
    } else if (method == 'exact2') {
      gmeta.cmbd <-
        gmeta.exact.LT(
          data_matrix,
          weight,
          gmo.xgrid,
          ci.level,
```

```
          mc.iteration,
          eta,
          verbose,
          report.error)
    } else if (method == 'MH' || method == 'Mantel-Haenszel') {
      gmeta.cmbd <- gmeta.MH(data_matrix, weight, gmo.xgrid, ci.level)
    } else if (method == 'Peto') {
      gmeta.cmbd <- gmeta.peto(data_matrix, weight, gmo.xgrid, ci.level)
    } else {
      stop('gmi.type 2x2 only match methods MH, Mantel-Haenszel, Peto,
           exact1, exact2.')
    }
    return(gmeta.cmbd)
  }

### 2x2 with exact1(LLX)
##### data processing
##### combine: exact1 method
gmeta.exact.combine <-
  function(gmeta.data,
           weight,
           gmo.xgrid,
           ci.level,
           mc.iteration,
           report.error) {
    #gmeta.cmbd <- gmeta.data
    #x.grids <- gmeta.data$x.grids
    # data_matrix
    data_matrix = gmeta.data$data_matrix
    # number of study
    K = nrow(data_matrix)
    # combine individual CDs
    alpha = 1 - ci.level
    xstmts <- .Estimates(data_matrix[, 1:4])
    # obtain weight
    if (is.null(weight)) {
      weight <- xstmts$weight.hat
    } else {
      warning('combine 2x2 with method="exact1", default weight=NULL is
```

```
          strongly recommended.')
}
indiv.cds <- gmeta.data$indiv.cds
# ##### ******************** logit ******************** ###
degreesfree <- 5 * K + 4
scaler <-
  -1 / sqrt((K * pi ^ 2 * (5 * K + 2)) / (3 * (5 * K + 4)))
combined.cd <-
  colSums(log(indiv.cds / (1 - indiv.cds))) * scaler
combined.cd <-
  pt(combined.cd, df = degreesfree, lower.tail = FALSE)

# combined CD function
combinedCDF <-
  function(theta) {
    # combined CD function, used when searching for quantiles
    ff = sapply(
      1:K,
      FUN = function(j) {
        midp.oddsratio(data_matrix[j, 1],
                       data_matrix[j, 3],
                       data_matrix[j, 4],
                       data_matrix[j, 5],
                       theta)
      }
    )
    ff = ifelse(ff < 1 - 0.1 ^ 12, ff, 1 - 0.1 ^ 12)
    ff = pt(sum(log(ff / (1 - ff))) * scaler, df = degreesfree,
            lower.tail = FALSE)
    return(ff)
  }
# inference derived from combined CD function
mn.xstmt = log(.quantileCD(combinedCDF, 1 / 2))            # mean
lower.ci = log(.quantileCD(combinedCDF, alpha / 2))        # ci.lower
upper.ci = log(.quantileCD(combinedCDF, 1 - alpha / 2))    # ci.upper
# null hypothesis: odd-ratio==1
o.pvalue = 2 * min(combinedCDF(1), 1 - combinedCDF(1))
# adjust individual CDs and combined CD function to reduce test.size.error
if (is.na(xstmts$or.hat) == TRUE ||
```

```
    xstmts$or.hat == 0 || xstmts$or.hat == Inf) {
  mn.xstmt.adj = xstmts$or.hat
  lower.ci.adj = NA
  upper.ci.adj = NA
  o.pvalue.adj = NA
  coverage.prbblty.error     = NA
  coverage.prbblty.error.adj = NA
} else {
  indiv.cds.adj   <-
    t(sapply(
      1:K,
      FUN = function(j) {
        adjust.beta(
          gmeta.data$indiv.cds[j, ],
          data_matrix[j, 3],
          xstmts$pt.hat[j],
          data_matrix[j, 4],
          xstmts$pc.hat[j])
      }))

  # combined CD function - adjusted
  combined.cd.adj <-
    colSums(log(indiv.cds.adj / (1 - indiv.cds.adj))) * scaler
  combined.cd.adj <-
    pt(combined.cd.adj, df = degreesfree, lower.tail = FALSE)
  combinedCDF.adj <-
    function(theta) {
      # combined CD function, used when searching for quantiles
      ff = sapply(
        1:K,
        FUN = function(j) {
          midp.oddsratio(data_matrix[j, 1],
                         data_matrix[j, 3],
                         data_matrix[j, 4],
                         data_matrix[j, 5],
                         theta)
        })
      ff = sapply(
        1:K,
```

```
        FUN = function(j) {
          adjust.beta(ff[j],
                      data_matrix[j, 3],
                      xstmts$pt.hat[j],
                      data_matrix[j, 4],
                      xstmts$pc.hat[j])
        })
      ff = ifelse(ff < 1 - 0.1 ^ 12, ff, 1 - 0.1 ^ 12)
      ff = pt(sum(log(ff / (1 - ff))) * scaler, df = degreesfree,
              lower.tail = FALSE)
      return(ff)
    }
  # inference derived from combined CD function - adjusted
  mn.xstmt.adj = log(.quantileCD(combinedCDF.adj, 1 / 2))       # mean
  lower.ci.adj = log(.quantileCD(combinedCDF.adj, alpha / 2))  # ci.lower
  # ci.upper
  upper.ci.adj = log(.quantileCD(combinedCDF.adj, 1 - alpha / 2))
  # null hypothesis: odd-ratio==1
  o.pvalue.adj = 2 * min(combinedCDF.adj(1), 1 - combinedCDF.adj(1))
  # calculate coverage probability error
  coverage.prbblty.error     <- 'Not Requested'
  coverage.prbblty.error.adj <- 'Not Requested'
  if (report.error) {
    ee = test.size.error(
      alpha / 2,
      data_matrix[, 3],
      xstmts$pt.hat,
      data_matrix[, 4],
      xstmts$pc.hat,
      weight,
      result.1minusS = TRUE,
      mc.iteration)
    coverage.prbblty.error     = ee$Test.Size.Error.1minusS -
      ee$Test.Size.Error
    coverage.prbblty.error.adj = ee$Test.Size.Error.Adjusted.1minusS -
      ee$Test.Size.Error.Adjusted
  }
}
# return
```

```
gmeta.cmbd <- list(
  # input
  data_matrix = gmeta.data$data_matrix[, 1:4],
  # individual CDs
  individual.cds     = gmeta.data$indiv.cds,
  individual.cis     = gmeta.data$indiv.cis,
  individual.medians = gmeta.data$indiv.medians,
  individual.means   = gmeta.data$indiv.means,
  individual.stddevs = gmeta.data$indiv.stddevs,
  # combined CD function
  combined.cd        = combined.cd,
  combined.density   = F2f(gmo.xgrid, combined.cd),
  combined.mean      = mn.xstmt,
  combined.median    = gmeta.cd.median(gmo.xgrid, combined.cd),
  combined.sd        = gmeta.cd.stddev(gmo.xgrid, combined.cd),
  combined.ci        = c(lower.ci, upper.ci),
  # p-value for null hypothesis: odd-ratio==1
  pvalue = o.pvalue,
  # combined CD function - adjusted
  combined.cd.adjusted       = combined.cd.adj,
  combined.density.adjusted  = F2f(gmo.xgrid, combined.cd.adj),
  combined.mean.adjusted     = mn.xstmt.adj,
  combined.median.adjusted   = gmeta.cd.median(gmo.xgrid, combined.cd.adj),
  combined.sd.adjusted       = gmeta.cd.stddev(gmo.xgrid, combined.cd.adj),
  combined.ci.adjusted       = c(lower.ci.adj, upper.ci.adj),
  # p-value for null hypothesis: odd-ratio==1 - adjusted
  pvalue.adj = o.pvalue.adj,
  # coverage probability error - check liu2012exact
  coverage.prbblty.error     = coverage.prbblty.error,
  coverage.prbblty.error.adj = coverage.prbblty.error.adj,
  # other information
  method       = 'exact1',
  linkfunc     = 'inverse-fisher-exact-test-function',
  #[? adjusted]
  weight       = weight,
  tau2         = NULL,
  ci.level     = ci.level,
  verbose      = report.error,
  mc.iteration = mc.iteration,
```

```
    report.error = report.error,
    # output gridding points
    x.grids      = gmo.xgrid)
  return(gmeta.cmbd)
}
```

Simulations R Code - Modification2.R:

```
##############################################################################
#
#     Use Logit instead of Normal CDF in Exact 1 Method - with weights
#
##############################################################################
library(binom)
library(BiasedUrn)
source("gmeta_code_functions_to_load.R")

# input matrix of 2x2 tables in format c(rx-event_treatment_group,
# nx-number_observation_treatment_group, ry-event_control_group,
# ny-number_observation_treatment_group) for each row of the matrix
# edata = as.data.frame(matrix(c(3, 2, 4, 5, 1, 2, 4, 6), nrow = 2, ncol = 4))
# colnames(edata) = c("x", "n", "y", "m")

# ****************************************************************************
#    main
# ****************************************************************************

# main function
gmeta <- function(gmi,
                  gmi.type = c('pivot', 'cd', 'pvalue', '2x2'),
                  method = c(
                    'fixed-mle',
                    'fixed-robust1',
                    'fixed-robust2',
                    'fixed-robust2(sqrt12)',
                    'random-mm',
                    'random-reml',
                    'random-tau2',
                    'random-robust1',
                    'random-robust2',
                    'random-robust2(sqrt12)',
                    'fisher',
                    'normal',
                    'stouffer',
                    'min',
                    'tippett',
```

```
                    'max',
                    'sum',
                    'MH',
                    'Mantel-Haenszel',
                    'Peto',
                    'exact1',
                    'exact2'),
                  linkfunc = c('inverse-normal-cdf', 'inverse-laplace-cdf'),
                  weight = NULL,
                  study.names = NULL,
                  gmo.xgrid = NULL,
                  ci.level = 0.95,
                  tau2 = NULL,
                  mc.iteration = 10000,
                  eta = 'Inf',
                  verbose = FALSE,
                  report.error = FALSE) {
  UseMethod('gmeta')
}

# main function
gmeta.default <- function(gmi,
                          gmi.type = c('pivot',
                                       'cd',
                                       'pvalue',
                                       '2x2'),
                          method   = c(
                            'fixed-mle',
                            'fixed-robust1',
                            'fixed-robust2',
                            'fixed-robust2(sqrt12)',
                            'random-mm',
                            'random-reml',
                            'random-tau2',
                            'random-robust1',
                            'random-robust2',
                            'random-robust2(sqrt12)',
                            'fisher',
                            'normal',
```

```
                          'stouffer',
                          'min',
                          'tippett',
                          'max',
                          'sum',
                          'MH',
                          'Mantel-Haenszel',
                          'Peto',
                          'exact1',
                          'exact2'
                        ),
                        linkfunc = c('inverse-normal-cdf',
                                     'inverse-laplace-cdf'),
                        weight   = NULL,
                        study.names = NULL,
                        gmo.xgrid = NULL,
                        ci.level = 0.95,
                        tau2     = NULL,
                        mc.iteration = 10000,
                        eta      = 'Inf',
                        verbose  = FALSE,
                        report.error = FALSE) {
#check
mf           <- match.call()
ci.level  <-
  ifelse(is.numeric(ci.level) && (ci.level >= 0) && (ci.level <= 1),
         ci.level,
         0.95)
mc.iteration <-
  round(ifelse(
    is.numeric(mc.iteration) && (mc.iteration > 1),
    mc.iteration,
    10000))
# meta-analysis
if (gmi.type == 'pvalue') {
  gmo <- gmeta.p(gmi, method)
} else if (gmi.type == 'cd' || gmi.type == 'pivot') {
  gmo <- gmeta.m(gmi,
                 gmi.type,
```

```
                    method,
                    linkfunc,
                    weight,
                    gmo.xgrid,
                    ci.level,
                    tau2,
                    verbose)
  } else if (gmi.type == '2x2') {
    # combine 2x2 tables
    gmo <-
      gmeta.e(gmi,
              method,
              weight,
              gmo.xgrid,
              ci.level,
              mc.iteration,
              eta,
              verbose,
              report.error)
  } else {
    stop("gmi.type must be 'cd', 'pivot', 'pvalue' or '2x2'.")
  }
  # post processing
  gmo$call   <- match.call()
  gmo$input  <- gmi
  gmo$alpha  <- 1 - ci.level
  gmo$study.names <- study.names
  # registr S3 class
  if (gmi.type == 'pvalue') {
    class(gmo) <- c('gmeta.p', 'gmeta')
  } else if (gmi.type == 'cd' || gmi.type == 'pivot') {
    class(gmo) <- c('gmeta.m', 'gmeta')
  } else if (gmi.type == '2x2') {
    class(gmo) <- c('gmeta.e', 'gmeta')
  } else {
    stop("gmi.type must be 'cd', 'pivot', 'pvalue' or '2x2'.")
  }
  return(gmo)
}
```

```
## meta-analysis - combine evidence from 2x2 tables
# ****************************************************************************
#    gmeta.e() - combine evidence from 2x2 tables (exact methods)
# ****************************************************************************
## main[gmeta.e]
gmeta.e <-
  function(gmi,
           method,
           weight,
           gmo.xgrid,
           ci.level,
           mc.iteration,
           eta,
           verbose,
           report.error) {
    data_matrix = as.matrix(cbind(gmi[, 1], gmi[, 3], gmi[, 2], gmi[, 4],
                                  gmi[, 1] + gmi[, 3]))
    colnames(data_matrix) = c("x", "y", "N", "M", "x+y")
    if (method == 'exact1') {
      gmeta.data <- gmeta.exact.indiv(data_matrix, gmo.xgrid, ci.level)
      gmeta.cmbd <-
        gmeta.exact.combine(gmeta.data,
                            weight,
                            gmo.xgrid,
                            ci.level,
                            mc.iteration,
                            report.error)
    } else if (method == 'exact2') {
      gmeta.cmbd <-
        gmeta.exact.LT(
          data_matrix,
          weight,
          gmo.xgrid,
          ci.level,
          mc.iteration,
          eta,
          verbose,
          report.error
```

```
        )
    } else if (method == 'MH' || method == 'Mantel-Haenszel') {
      gmeta.cmbd <- gmeta.MH(data_matrix, weight, gmo.xgrid, ci.level)
    } else if (method == 'Peto') {
      gmeta.cmbd <- gmeta.peto(data_matrix, weight, gmo.xgrid, ci.level)
    } else {
      stop('gmi.type 2x2 only match methods MH, Mantel-Haenszel, Peto,
           exact1, exact2.')
    }
    return(gmeta.cmbd)
  }

### 2x2 with exact1(LLX)
##### data processing
##### combine: exact1 method
gmeta.exact.combine <-
  function(gmeta.data,
           weight,
           gmo.xgrid,
           ci.level,
           mc.iteration,
           report.error) {
    #gmeta.cmbd <- gmeta.data
    #x.grids <- gmeta.data$x.grids
    # data_matrix
    data_matrix = gmeta.data$data_matrix
    # number of study
    K = nrow(data_matrix)
    # combine individual CDs
    alpha = 1 - ci.level
    xstmts <- .Estimates(data_matrix[, 1:4])
    # obtain weight
    if (is.null(weight)) {
      weight <- xstmts$weight.hat
    } else {
      warning('combine 2x2 with method="exact1",
              default weight=NULL is strongly recommended.')
    }
    indiv.cds <- gmeta.data$indiv.cds
```

```
# ##### ******************** logit ******************** ###

degreesfree <- 4 + (5 / (sum(weight ^ 4 / sum(weight ^ 2) ^ 2)))
scaler <--1 / sqrt(((pi ^ 2 * (degreesfree - 2)) / degreesfree) *
                     sum(weight ^ 2 / 3))
combined.cd <-
  colSums(weight %*% log(indiv.cds / (1 - indiv.cds))) * scaler
combined.cd <-
  pt(combined.cd, df = degreesfree, lower.tail = FALSE)

# combined CD function
combinedCDF <-
  function(theta) {
    # combined CD function, used when searching for quantiles
    ff = sapply(
      1:K,
      FUN = function(j) {
        midp.oddsratio(data_matrix[j, 1],
                       data_matrix[j, 3],
                       data_matrix[j, 4],
                       data_matrix[j, 5],
                       theta)
      }
    )
    ff = ifelse(ff < 1 - 0.1 ^ 12, ff, 1 - 0.1 ^ 12)
    ff = pt(sum(weight %*% log(ff / (1 - ff))) * scaler,
            df = degreesfree,
            lower.tail = FALSE)
    return(ff)
  }
# inference derived from combined CD function
mn.xstmt = log(.quantileCD(combinedCDF, 1 / 2))            # mean
lower.ci = log(.quantileCD(combinedCDF, alpha / 2))        # ci.lower
upper.ci = log(.quantileCD(combinedCDF, 1 - alpha / 2))    # ci.upper
# null hypothesis: odd-ratio==1
o.pvalue = 2 * min(combinedCDF(1), 1 - combinedCDF(1))
# adjust individual CDs and combined CD function to reduce test.size.error
if (is.na(xstmts$or.hat) == TRUE ||
```

```
    xstmts$or.hat == 0 || xstmts$or.hat == Inf) {
  mn.xstmt.adj = xstmts$or.hat
  lower.ci.adj = NA
  upper.ci.adj = NA
  o.pvalue.adj = NA
  coverage.prbblty.error     = NA
  coverage.prbblty.error.adj = NA
} else {
  indiv.cds.adj   <-
    t(sapply(
      1:K,
      FUN = function(j) {
        adjust.beta(
          gmeta.data$indiv.cds[j, ],
          data_matrix[j, 3],
          xstmts$pt.hat[j],
          data_matrix[j, 4],
          xstmts$pc.hat[j]
        )
      }
    ))
  combined.cd.adj <-
    colSums(weight %*% log(indiv.cds.adj / (1 - indiv.cds.adj))) * scaler
  combined.cd.adj <-
    pt(combined.cd.adj, df = degreesfree, lower.tail = FALSE)
  combinedCDF.adj <-
    function(theta) {
      # combined CD function, used when searching for quantiles
      ff = sapply(
        1:K,
        FUN = function(j) {
          midp.oddsratio(data_matrix[j, 1],
                         data_matrix[j, 3],
                         data_matrix[j, 4],
                         data_matrix[j, 5],
                         theta)
        }
      )
      ff = sapply(
```

```
      1:K,
      FUN = function(j) {
        adjust.beta(ff[j],
                    data_matrix[j, 3],
                    xstmts$pt.hat[j],
                    data_matrix[j, 4],
                    xstmts$pc.hat[j])
      }
    )
    ff = ifelse(ff < 1 - 0.1 ^ 12, ff, 1 - 0.1 ^ 12)
    ff = pt(sum(weight %*% log(ff / (1 - ff))) * scaler,
            df = degreesfree,
            lower.tail = FALSE)
    return(ff)
  }
# inference derived from combined CD function - adjusted
mn.xstmt.adj = log(.quantileCD(combinedCDF.adj, 1 / 2))       # mean
lower.ci.adj = log(.quantileCD(combinedCDF.adj, alpha / 2))   # ci.lower
# ci.upper
upper.ci.adj = log(.quantileCD(combinedCDF.adj, 1 - alpha / 2))
# null hypothesis: odd-ratio==1
o.pvalue.adj = 2 * min(combinedCDF.adj(1), 1 - combinedCDF.adj(1))
# calculate coverage probability error
coverage.prbblty.error     <- 'Not Requested'
coverage.prbblty.error.adj <- 'Not Requested'
if (report.error) {
  ee = test.size.error(
    alpha / 2,
    data_matrix[, 3],
    xstmts$pt.hat,
    data_matrix[, 4],
    xstmts$pc.hat,
    weight,
    result.1minusS = TRUE,
    mc.iteration
  )
  coverage.prbblty.error     = ee$Test.Size.Error.1minusS -
    ee$Test.Size.Error
  coverage.prbblty.error.adj = ee$Test.Size.Error.Adjusted.1minusS -
```

```
      ee$Test.Size.Error.Adjusted
  }
}
# return
gmeta.cmbd <- list(
  # input
  data_matrix = gmeta.data$data_matrix[, 1:4],
  # individual CDs
  individual.cds     = gmeta.data$indiv.cds,
  individual.cis     = gmeta.data$indiv.cis,
  individual.medians = gmeta.data$indiv.medians,
  individual.means   = gmeta.data$indiv.means,
  individual.stddevs = gmeta.data$indiv.stddevs,
  # combined CD function
  combined.cd        = combined.cd,
  combined.density   = F2f(gmo.xgrid, combined.cd),
  combined.mean      = mn.xstmt,
  combined.median    = gmeta.cd.median(gmo.xgrid, combined.cd),
  combined.sd        = gmeta.cd.stddev(gmo.xgrid, combined.cd),
  combined.ci        = c(lower.ci, upper.ci),
  # p-value for null hypothesis: odd-ratio==1
  pvalue = o.pvalue,
  # combined CD function - adjusted
  combined.cd.adjusted       = combined.cd.adj,
  combined.density.adjusted  = F2f(gmo.xgrid, combined.cd.adj),
  combined.mean.adjusted     = mn.xstmt.adj,
  combined.median.adjusted   = gmeta.cd.median(gmo.xgrid, combined.cd.adj),
  combined.sd.adjusted       = gmeta.cd.stddev(gmo.xgrid, combined.cd.adj),
  combined.ci.adjusted       = c(lower.ci.adj, upper.ci.adj),
  # p-value for null hypothesis: odd-ratio==1 - adjusted
  pvalue.adj = o.pvalue.adj,
  # coverage probability error - check liu2012exact
  coverage.prbblty.error     = coverage.prbblty.error,
  coverage.prbblty.error.adj = coverage.prbblty.error.adj,
  # other information
  method       = 'exact1',
  linkfunc     = 'inverse-fisher-exact-test-function',
  #[? adjusted]
  weight       = weight,
```

```
    tau2          = NULL,
    ci.level      = ci.level,
    verbose       = report.error,
    mc.iteration = mc.iteration,
    report.error = report.error,
    # output gridding points
    x.grids       = gmo.xgrid
  )
  # return
  return(gmeta.cmbd)
}
```

Simulations R Code - Modification3.R:

```
#############################################################################
#
#                   Use Tian Exact 2 Method with Liu Weights
#
#############################################################################
library(binom)
library(BiasedUrn)
source("gmeta_code_functions_to_load.R")

# input matrix of 2x2 tables in format c(rx-event_treatment_group,
# nx-number_observation_treatment_group, ry-event_control_group,
# ny-number_observation_treatment_group) for each row of the matrix
# edata = as.data.frame(matrix(c(3, 2, 4, 5, 1, 2, 4, 6), nrow = 2, ncol = 4))
# colnames(edata) = c("x", "n", "y", "m")

# ***************************************************************************
#    main
# ***************************************************************************

# main function
gmeta <- function(gmi,
                  gmi.type = c('pivot', 'cd', 'pvalue', '2x2'),
                  method = c(
                    'fixed-mle',
                    'fixed-robust1',
                    'fixed-robust2',
                    'fixed-robust2(sqrt12)',
                    'random-mm',
                    'random-reml',
                    'random-tau2',
                    'random-robust1',
                    'random-robust2',
                    'random-robust2(sqrt12)',
                    'fisher',
                    'normal',
                    'stouffer',
                    'min',
                    'tippett',
```

```
                    'max',
                    'sum',
                    'MH',
                    'Mantel-Haenszel',
                    'Peto',
                    'exact1',
                    'exact2'),
                  linkfunc = c('inverse-normal-cdf', 'inverse-laplace-cdf'),
                  weight = NULL,
                  study.names = NULL,
                  gmo.xgrid = NULL,
                  ci.level = 0.95,
                  tau2 = NULL,
                  mc.iteration = 10000,
                  eta = 'Inf',
                  verbose = FALSE,
                  report.error = FALSE) {
  UseMethod('gmeta')
}

# main function
gmeta.default <- function(gmi,
                          gmi.type = c('pivot',
                                       'cd',
                                       'pvalue',
                                       '2x2'),
                          method   = c(
                            'fixed-mle',
                            'fixed-robust1',
                            'fixed-robust2',
                            'fixed-robust2(sqrt12)',
                            'random-mm',
                            'random-reml',
                            'random-tau2',
                            'random-robust1',
                            'random-robust2',
                            'random-robust2(sqrt12)',
                            'fisher',
                            'normal',
```

```
                          'stouffer',
                          'min',
                          'tippett',
                          'max',
                          'sum',
                          'MH',
                          'Mantel-Haenszel',
                          'Peto',
                          'exact1',
                          'exact2'
                        ),
                        linkfunc = c('inverse-normal-cdf',
                                     'inverse-laplace-cdf'),
                        weight   = NULL,
                        study.names = NULL,
                        gmo.xgrid = NULL,
                        ci.level = 0.95,
                        tau2     = NULL,
                        mc.iteration = 10000,
                        eta      = 'Inf',
                        verbose  = FALSE,
                        report.error = FALSE) {
#check
mf           <- match.call()
ci.level  <-
  ifelse(is.numeric(ci.level) &&
           (ci.level >= 0) && (ci.level <= 1),
         ci.level,
         0.95)
mc.iteration <-
  round(ifelse(
    is.numeric(mc.iteration) && (mc.iteration > 1),
    mc.iteration,
    10000))
# meta-analysis
if (gmi.type == 'pvalue') {
  gmo <- gmeta.p(gmi, method)
} else if (gmi.type == 'cd' || gmi.type == 'pivot') {
  gmo <-
```

```
    gmeta.m(gmi,
            gmi.type,
            method,
            linkfunc,
            weight,
            gmo.xgrid,
            ci.level,
            tau2,
            verbose)
} else if (gmi.type == '2x2') {
  # combine 2x2 tables
  gmo <-
    gmeta.e(gmi,
            method,
            weight,
            gmo.xgrid,
            ci.level,
            mc.iteration,
            eta,
            verbose,
            report.error)
} else {
  stop("gmi.type must be 'cd', 'pivot', 'pvalue' or '2x2'.")
}
# post processing
gmo$call   <- match.call()
gmo$input  <- gmi
gmo$alpha  <- 1 - ci.level
gmo$study.names <- study.names
# registr S3 class
if (gmi.type == 'pvalue') {
  class(gmo) <- c('gmeta.p', 'gmeta')
} else if (gmi.type == 'cd' || gmi.type == 'pivot') {
  class(gmo) <- c('gmeta.m', 'gmeta')
} else if (gmi.type == '2x2') {
  class(gmo) <- c('gmeta.e', 'gmeta')
} else {
  stop("gmi.type must be 'cd', 'pivot', 'pvalue' or '2x2'.")
}
```

```
  # return
  return(gmo)
}

## meta-analysis - combine evidence from 2x2 tables
# ****************************************************************************
#    gmeta.e() - combine evidence from 2x2 tables (exact methods)
# ****************************************************************************
## main[gmeta.e]
gmeta.e <-
  function(gmi,
           method,
           weight,
           gmo.xgrid,
           ci.level,
           mc.iteration,
           eta,
           verbose,
           report.error) {
    data_matrix = as.matrix(cbind(gmi[, 1], gmi[, 3], gmi[, 2], gmi[, 4],
                                  gmi[, 1] + gmi[, 3]))
    colnames(data_matrix) = c("x", "y", "N", "M", "x+y")
    if (method == 'exact1') {
      gmeta.data <- gmeta.exact.indiv(data_matrix, gmo.xgrid, ci.level)
      gmeta.cmbd <-
        gmeta.exact.combine(gmeta.data,
                            weight,
                            gmo.xgrid,
                            ci.level,
                            mc.iteration,
                            report.error)
    } else if (method == 'exact2') {
      gmeta.cmbd <-
        gmeta.exact.LT(
          data_matrix,
          weight,
          gmo.xgrid,
          ci.level,
          mc.iteration,
```

```
          eta,
          verbose,
          report.error)
    } else if (method == 'MH' || method == 'Mantel-Haenszel') {
      gmeta.cmbd <- gmeta.MH(data_matrix, weight, gmo.xgrid, ci.level)
    } else if (method == 'Peto') {
      gmeta.cmbd <- gmeta.peto(data_matrix, weight, gmo.xgrid, ci.level)
    } else {
      stop('gmi.type 2x2 only match methods MH, Mantel-Haenszel, Peto,
           exact1, exact2.')
    }
    return(gmeta.cmbd)
  }

### 2x2 with exact2(LT's method) - risk difference[delta = p1 - p2]
gmeta.exact.LT <-
  function(data_matrix,
           weight,
           gmo.xgrid,
           ci.level,
           mc.iteration,
           eta,
           verbose,
           report.error) {
    # [update in v2.0, target p1-p2, following program is on p2-p1,
    #  so change case-ctrl order]
    data.mi = data_matrix[, c(2, 1, 4, 3)]

    # number of trial
    nstudy = dim(data.mi)[1]

    # delta - individual studies
    n1 = data.mi[, 3]
    n2 = data.mi[, 4]
    p1 = data.mi[, 1] / data.mi[, 3]
    p2 = data.mi[, 2] / data.mi[, 4]
    deltap = p2 - p1

    id = (1:nstudy)[p1 * p2 == 0]
```

```
n1[id] = data.mi[id, 3] + 1
n2[id] = data.mi[id, 4] + 1
p1[id] = (data.mi[id, 1] + 0.5) / (data.mi[id, 3] + 1)
p2[id] = (data.mi[id, 2] + 0.5) / (data.mi[id, 4] + 1)
varp = p1 * (1 - p1) / n1 + p2 * (1 - p2) / n2
wght = (n1 * n2 / (n1 + n2)) / sum(n1 * n2 / (n1 + n2))

# Mantel-Haenszel's method
mu.MH = sum(deltap * wght)
sd.MH = sqrt(sum(wght ^ 2 * varp))
ci.MH = c(mu.MH - 1.96 * sd.MH, mu.MH + 1.96 * sd.MH)
p.MH  = 1 - pchisq(mu.MH ^ 2 / sd.MH ^ 2, 1)

# gridding detla
d0 = max(abs(ci.MH))
delta.grd = sort(c(0, seq(
  from = max(-1, -d0 * 15),
  to = min(1, d0 * 15),
  length = length(gmo.xgrid) - 1)))

# exact p-values [for observed data] given true delta [risk difference]
diff.exact = function(x1,
                      x2,
                      n1,
                      n2,
                      delta.grd,
                      n.grd = 15,
                      midp = TRUE) {
  # fit
  fit = binom.confint(x1, n1, 0.9995, methods = 'exact')
  # l,u
  l = fit$lower
  u = fit$upper
  # grd
  p1.grd = seq(l, u, length = n.grd)
  pnull1.tot = matrix(0, n1 + 1, n.grd)
  for (b in 1:n.grd) {
    p1 = p1.grd[b]
    pnull1.tot[, b] = dbinom(c(0:n1), n1, p1)
```

```
}
# df & sd
dfnull = matrix(0, n1 + 1, n2 + 1)
sdnull = matrix(0, n1 + 1, n2 + 1)
for (i in 0:n1) {
  p1 = (i + 0.5) / (n1 + 1)
  p2 = (c(0:n2) + 0.5) / (n2 + 1)
  dfnull[i + 1,] = c(0:n2) / n2 - i / n1
  sdnull[i + 1,] = sqrt(p1 * (1 - p1) / n1 + p2 * (1 - p2) / n2)
}
# pv1 & pv2
pv1 = numeric(0)
pv2 = numeric(0)
for (theta in delta.grd) {
  p1 = (x1 + 0.5) / (n1 + 1)
  p2 = (x2 + 0.5) / (n2 + 1)
  tt  = (x2 / n2 - x1 / n1 - theta) / sqrt(p1 * (1 - p1) / n1 + p2 *
                                           (1 - p2) / n2)
  # null hypothesis detla==theta
  tnull   = (dfnull - theta) / sdnull
  pvalue1 = rep(0, n.grd)
  pvalue2 = rep(0, n.grd)
  error = 1e-6
  for (b in 1:n.grd) {
    p1 = p1.grd[b]
    p2 = p1 + theta
    if (p2 >= 0 && p2 <= 1) {
      pnull1 = pnull1.tot[, b]
      pnull2 = dbinom(c(0:n2), n2, p2)
      n1.adj = n1 + 1 - max(c(1, (1:(n1 + 1))[cumsum(sort(pnull1)) <
                                                error]))
      n2.adj = n2 + 1 - max(c(1, (1:(n2 + 1))[cumsum(sort(pnull2)) <
                                                error]))
      id1 = order(pnull1)
      id2 = order(pnull2)
      id1 = (id1[(n1 + 1):1])[1:n1.adj]
      id2 = (id2[(n2 + 1):1])[1:n2.adj]
      pnull = pnull1[id1] %*% t(pnull2[id2])
      if (midp == TRUE) {
```

```
          pvalue1[b] = sum(pnull[tnull[id1, id2] > tt]) +
            sum(pnull[tnull[id1, id2] == tt]) * 0.5
          pvalue2[b] = sum(pnull[tnull[id1, id2] < tt]) +
            sum(pnull[tnull[id1, id2] == tt]) * 0.5
        } else {
          pvalue1[b] = sum(pnull[tnull[id1, id2] > tt]) +
            sum(pnull[tnull[id1, id2] == tt])
          pvalue2[b] = sum(pnull[tnull[id1, id2] < tt]) +
            sum(pnull[tnull[id1, id2] == tt])
        }
      }
    }
    pv1 = c(pv1, max(pvalue1) + (1 - 0.9995))
    pv2 = c(pv2, max(pvalue2) + (1 - 0.9995))
  }
  return(list(pv1 = pv1, pv2 = pv2))
}

# pv1.pool&pv2.pool are the CDs of individual studies.
pv1.pool = numeric(0)
pv2.pool = numeric(0)
for (kk in 1:nstudy) {
  # verbose
  if (verbose) {
    # use report.error as a surrogate as verbose [use parameter
    # verbose - update v2.0]
    cat('2x2 exact2 processing trial:', kk, '\n')
  } # update status - due to slow speed of diff.exact().
  # resolve
  x1 = data.mi[kk, 1]
  x2 = data.mi[kk, 2]
  n1 = data.mi[kk, 3]
  n2 = data.mi[kk, 4]
  # fit
  fit = diff.exact(x1, x2, n1, n2, delta.grd, n.grd = 15, midp = TRUE)
  pv1.pool = rbind(pv1.pool, fit$pv1)
  pv2.pool = rbind(pv2.pool, fit$pv2)
}
n = length(gmo.xgrid)
```

```
for (i in 1:nstudy) {
  for (j in 1:length(gmo.xgrid)) {
    pv1.pool[i, (n - j + 1)] = max(pv1.pool[i, 1:(n - j + 1)])
    pv2.pool[i, j]       = max(pv2.pool[i, j:n])
  }
}
# pv1.pool&pv2.pool are the CDs of individual studies.

# ************************************************************************
# weight for individual study
xstmts <- .Estimates(data.mi)
if (is.null(weight)) {
  weight <- xstmts$weight.hat
} else {
  if (verbose) {
    # use report.error as a surrogate as verbose [use parameter
    # verbose - update v2.0]
    cat(
      '\nwarning: use user specified weights, instead of the default
      weight determined by study size.\n')
  }
}
# ************************************************************************

# combine individual CDs
if (is.numeric(eta)) {
  # eta in [0,1]
  if (!(min(eta) >= 0 && max(eta) <= 1)) {
    eta = seq(0.05, 0.95, length = 20)
  }
  # F0^{-1}()
  F0inv <- function(ui) {
    sum(((ui > (1 - eta)) - eta) / (eta * (1 - eta)))
  }
} else {
  # eta = 'Inf'
  # for the case K\to\inf:
  # pv1.pool&pv2.pool - adjust to make sense ln(ui/(1-ui))
  pv1.pool[pv1.pool <= 0] = 1e-6
```

```
  pv1.pool[pv1.pool >= 1] = 1 - 1e-6
  pv2.pool[pv2.pool <= 0] = 1e-6
  pv2.pool[pv2.pool >= 1] = 1 - 1e-6
  # pv2.pool = 1+(1-0.9995)*2 - pv2.pool
  # F0^{-1}()
  F0inv <- function(ui) {
    log(ui / (1 - ui))
  }
}

# combine recipe g_c()
gc <- function(u) {
  sum(sapply(u, F0inv) * weight)
}
# simulation for G_c()
T = numeric(mc.iteration)
for (b in 1:mc.iteration) {
  u = runif(nstudy)
  T[b] = gc(u)
}
Gc = ecdf(T)

# combined CDs
Hc1 = Gc(apply(pv1.pool, 2, gc)) # HC1 non-decreasing
Hc2 = Gc(apply(pv2.pool, 2, gc)) # HC2 non-increasing
combined.ci = c(max(delta.grd[Hc1 < 0.025]), min(delta.grd[Hc2 < 0.025]))

# post processing
integrated2CDs <- function(cd1, cd2, delta.grd) {
  mdnpt = delta.grd[round(median(which(abs(cd1 - cd2) == min(abs(
    cd1 - cd2)))))]
  # 1+(1-0.9995)*2-cd2
  intgrtdcd = ifelse(delta.grd < mdnpt, cd1, 1 + (1 - 0.9995) * 2 - cd2)
  for (j in 1:n) {
    intgrtdcd[j] = max(intgrtdcd[1:j])
  }
  intgrtdcd[intgrtdcd <= 0] = 1e-6
  intgrtdcd[intgrtdcd >= 1] = 1 - 1e-6
```

```
  return(intgrtdcd)
}

# individual CDs
indiv.cds <- NULL
for (i in 1:nstudy) {
  indiv.cds = rbind(indiv.cds,
                    integrated2CDs(pv1.pool[i, ],
                                   pv2.pool[i, ], delta.grd))
}
indiv.means   = numeric(0)
indiv.stddevs = numeric(0)
indiv.ci      = numeric(0)
for (i in 1:nstudy) {
  indiv.means[i]   = gmeta.cd.mean(delta.grd, indiv.cds[i, ])
  indiv.stddevs[i] = gmeta.cd.stddev(delta.grd, indiv.cds[i, ])
  indiv.ci         = rbind(indiv.ci,
                           gmeta.cd.mdncis(delta.grd,
                                           indiv.cds[i, ], 1 - ci.level))
}

# combined CD
cmbdF   = integrated2CDs(Hc1, Hc2, delta.grd)
cmbdf   = F2f(delta.grd, cmbdF)
cmbdmn  = gmeta.cd.mean(delta.grd, cmbdF)
cmbdmdn = gmeta.cd.median(delta.grd, cmbdF)
cmbdsdv = gmeta.cd.stddev(delta.grd, cmbdF)
# report.error
if (min(gmo.xgrid) < -1 || max(gmo.xgrid) > 1) {
  #gmo.xgrid = seq(from=-1, to=1, length=length(gmo.xgrid))
  gmo.xgrid = delta.grd
  if (report.error) {
    warning('\nuse exact2method combine 2x2 tables, parameter is
            risk-difference, only evaluate gmo.xgrid within [-1,1]\n')
  }
}
# return
gmeta.cmbd <- list(
  # input
```

```
    data_matrix = data.mi,
    # individual CDs
    individual.cds     = indiv.cds,
    individual.cis     = indiv.ci[, c(1, 3)],
    individual.medians = indiv.ci[, 2],
    individual.means   = indiv.means,
    individual.stddevs = indiv.stddevs,
    # combined CD
    combined.cd        = cmbdF,
    combined.density   = cmbdf,
    combined.mean      = cmbdmn,
    combined.median    = cmbdmdn,
    combined.sd        = cmbdsdv,
    combined.ci        = combined.ci,
    # other information
    method       = 'exact2',
    linkfunc     = 'log(u/(1-u))',
    weight       = weight,
    tau2         = NULL,
    ci.level     = ci.level,
    mc.iteration = mc.iteration,
    eta          = eta,
    verbose      = verbose,
    report.error = report.error,
    # output gridding points
    x.grids      = delta.grd)
  return(gmeta.cmbd)
}
```

Simulations R Code - Simulation_Modification1.R:

```
source("Modification1.R")

inv_logit_liu_exact1 <- function(sdata) {
  mdata <- cbind(sdata[1, colnames(sdata) == 1],  # TRT_event
                 colSums(sdata[, colnames(sdata) == 1]),  # TRT_n
                 sdata[1, colnames(sdata) == 0],  # CTRL_event
                 colSums(sdata[, colnames(sdata) == 0]))  # CTRL_n
  m_inv_logit_liu <- gmeta(mdata,
                           gmi.type = "2x2",
                           method = "exact1",
                           gmo.xgrid = seq(-1, 1, by = 0.001),
                           report.error = TRUE)
  sig01 <- NA
  if (m_inv_logit_liu$combined.ci[1] < 0 &&
      m_inv_logit_liu$combined.ci[2] > 0) {
    sig01 <- 0
  } else if (m_inv_logit_liu$combined.ci[1] < 0 &&
             m_inv_logit_liu$combined.ci[2] < 0) {
    sig01 <- 1
  } else if (m_inv_logit_liu$combined.ci[1] > 0 &&
             m_inv_logit_liu$combined.ci[2] > 0) {
    sig01 <- 1
  } else if (m_inv_logit_liu$combined.ci[1] == 0 &&
             m_inv_logit_liu$combined.ci[2] > 0) {
    sig01 <- 0
  } else if (m_inv_logit_liu$combined.ci[1] < 0 &&
             m_inv_logit_liu$combined.ci[2] == 0) {
    sig01 <- 0
  }
  return(sig01)
}

tau2 <- 0
mu <- -5
reps <- 1000

# workspace generated from ma.sim.dat.R and Generate_Data.R code in Appendix C
load(paste0("Tau2=", tau2, ",Mu=", mu, ".RData"))
```

```
all_dat_small <- all_dat[1:reps]

# Inverse Fisher Exact Test Function - Exact 1 Method
set.seed(1993)

start_time <- Sys.time()
results_inv_logit <- sapply(all_dat_small, function(x)
  inv_logit_liu_exact1(x))
Sys.time() - start_time  # Time difference

typeIerror <- sum(results_inv_logit) / length(results_inv_logit)
typeIerror
```

Simulations R Code - Simulation_Modification2.R:

```
source("Modification2.R")

inv_logit_liu_exact1 <- function(sdata) {
  mdata <- cbind(sdata[1, colnames(sdata) == 1],  # TRT_event
                 colSums(sdata[, colnames(sdata) == 1]),  # TRT_n
                 sdata[1, colnames(sdata) == 0],  # CTRL_event
                 colSums(sdata[, colnames(sdata) == 0]))  # CTRL_n
  m_inv_logit_liu <- gmeta(mdata,
                           gmi.type = "2x2",
                           method = "exact1",
                           gmo.xgrid = seq(-1, 1, by = 0.001),
                           report.error = TRUE)
  sig01 <- NA
  if (m_inv_logit_liu$combined.ci[1] < 0 &&
      m_inv_logit_liu$combined.ci[2] > 0) {
    sig01 <- 0
  } else if (m_inv_logit_liu$combined.ci[1] < 0 &&
             m_inv_logit_liu$combined.ci[2] < 0) {
    sig01 <- 1
  } else if (m_inv_logit_liu$combined.ci[1] > 0 &&
             m_inv_logit_liu$combined.ci[2] > 0) {
    sig01 <- 1
  } else if (m_inv_logit_liu$combined.ci[1] == 0 &&
             m_inv_logit_liu$combined.ci[2] > 0) {
    sig01 <- 0
  } else if (m_inv_logit_liu$combined.ci[1] < 0 &&
             m_inv_logit_liu$combined.ci[2] == 0) {
    sig01 <- 0
  }
  return(sig01)
}

tau2 <- 0
mu <- -5
reps <- 1000

# workspace generated from ma.sim.dat.R and Generate_Data.R code in Appendix C
load(paste0("Tau2=", tau2, ",Mu=", mu, ".RData"))
```

```
all_dat_small <- all_dat[1:reps]

# Inverse Fisher Exact Test Function - Exact 1 Method
set.seed(1993)

start_time <- Sys.time()
results_inv_logit <- sapply(all_dat_small, function(x)
  inv_logit_liu_exact1(x))
Sys.time() - start_time  # Time difference

typeIerror <- sum(results_inv_logit) / length(results_inv_logit)
typeIerror
```

Simulations R Code - Simulation_Modification3.R:

```
source("Modification3.R")

tian_liu_weight_exact2 <- function(sdata) {
  mdata <- cbind(sdata[1, colnames(sdata) == 1],  # TRT_event
                 colSums(sdata[, colnames(sdata) == 1]),  # TRT_n
                 sdata[1, colnames(sdata) == 0],  # CTRL_event
                 colSums(sdata[, colnames(sdata) == 0]))  # CTRL_n
  m_tian_liu_weight_res <- gmeta(mdata,
                                 gmi.type = "2x2",
                                 method = "exact2",
                                 gmo.xgrid = seq(-1, 1, by = 0.001),
                                 report.error = TRUE)
  sig01 <- NA
  if (m_tian_liu_weight_res$combined.ci[1] < 0 &&
      m_tian_liu_weight_res$combined.ci[2] > 0) {
    sig01 <- 0
  } else if (m_tian_liu_weight_res$combined.ci[1] < 0 &&
             m_tian_liu_weight_res$combined.ci[2] < 0) {
    sig01 <- 1
  } else if (m_tian_liu_weight_res$combined.ci[1] > 0 &&
             m_tian_liu_weight_res$combined.ci[2] > 0) {
    sig01 <- 1
  } else if (m_tian_liu_weight_res$combined.ci[1] == 0 &&
             m_tian_liu_weight_res$combined.ci[2] > 0) {
    sig01 <- 0
  } else if (m_tian_liu_weight_res$combined.ci[1] < 0 &&
             m_tian_liu_weight_res$combined.ci[2] == 0) {
    sig01 <- 0
  }
  return(sig01)
}

tau2 <- 0
mu <- -5
reps <- 1000

# workspace generated from ma.sim.dat.R and Generate_Data.R code in Appendix C
load(paste0("Tau2=", tau2, ",Mu=", mu, ".RData"))
```

```
all_dat_small <- all_dat[1:reps]

# Tian Exact 2 Method with Liu Weights
set.seed(1993)

start_time <- Sys.time()
results_tian_liu_weights <- sapply(all_dat_small, function(x)
  tian_liu_weight_exact2(x))
Sys.time()- start_time  # Time difference

typeIerror <- sum(results_tian_liu_weights) / length(results_tian_liu_weights)
typeIerror
```

CURRICULUM VITAE

Brinley Zabriskie
brinley.zabriskie@gmail.com

EDUCATION

PhD in Statistics, Utah State University (USU), Logan, UT **05/2019**

GPA: 4.00
Presidential Doctoral Research Fellowship

Dissertation Title:	Meta–Analysis with Small or Sparse Samples
Committee:	Chris Corcoran (major professor), Tyler Brough, David Brown, Adele Cutler, John Stevens
Research Interests:	Techniques for small samples sizes, exact analysis, categorical data analysis, meta-analysis, network meta-analysis, biostatistics, and machine learning techniques for classification and prediction

BSc in Mathematics and Statistics, USU, Logan, UT **05/2014**

GPA: 3.93
magna cum laude, Presidential Scholarship, Dean's List Fall 2011 and Spring 2014

PUBLICATIONS

IN PREPARATION

- **Zabriskie, B.**, Corcoran, C., Senchaudhuri, P. "Exact Meta–Analysis Using a Permutation-Based Approach."
- **Zabriskie, B.**, Corcoran, C., Senchaudhuri, P. "Combining Confidence Distributions for Rare Event Meta–Analysis."

PRESENTATIONS

- **Zabriskie, B.**, Corcoran, C., Senchaudhuri, P. (2018), "Exact Meta–Analysis Using a Permutation-Based Approach," Oral and Poster Presentation at the Women in Statistics and Data Science (WSDS) Conference, Cincinnati, OH.
- **Zabriskie, B.**, Corcoran, C., Senchaudhuri, P. (2018), "Combining Confidence Distributions for Rare Event Meta–Analysis," Oral Presentation at the Joint Statistical Meetings (JSM), Vancouver, British Columbia, Canada.
- Corcoran, C., **Zabriskie, B.**, Senchaudhuri, P. (2018), "Exact Meta–Analysis Using a Permutation-Based Approach," Oral Presentation at the International Society of Non-Parametric Statistics (ISNPS) Conference, Salerno, Southern Italy.
- **Zabriskie, B.**, Corcoran, C., Senchaudhuri, P. (2018), "Combining Confidence Distributions for Rare Event Meta–Analysis," Oral Presentation at USU's Research Symposium, Logan, UT. (Received an award for this presentation)
- **Zabriskie, B.**, Corcoran, C., Senchaudhuri, P. (2016), "An Overview of Meta–Analysis Software Features," Oral Presentation to managers at Cytel, a large statistical software company, Cambridge, MA.

TEACHING EXPERIENCE

Business Statistics Course Instructor, USU, Logan, UT 08/2018–05/2019

- Energetically teach a 4–credit business statistics course of over 160 students
- Create new course material and continue to improve and update existing material
- Collaborate with two teaching assistants to create a cohesive, enriching environment for students

Business Statistics Teaching Assistant, USU, Logan, UT 01/2018–05/2018

- Designed and updated course material, which included creating online Excel assignments
- Taught two courses, each with more than 30 students

Business Statistics Teaching Assistant, USU, Logan, UT 08/2017–12/2017

- Collaborated with colleagues in the Jon M. Huntsman School of Business and the Mathematics and Statistics Department to create an updated version of this course to better prepare business students for their future courses and careers
- Effectively taught one course of 17 students, focusing on the students' individual needs

Statistics Course Instructor, USU, Logan, UT 06/2016–08/2016

- Dynamically taught a 5–credit introductory statistics class of 45 students, which included broadcast students from around the state of Utah
- Engaged students during two–hour class lectures for five days a week over a seven-week period by creating interactive classes and using a wide variety of teaching methods
- Provided ample help outside of class for both traditional and broadcast students

Statistics Teaching Assistant, USU, Logan, UT 01/2016–04/2016

- Taught introductory statistics to three classes of more than 30 science and engineering students
- Fostered a positive learning environment where students felt comfortable, asked questions, and were encouraged to meet one–on–one for additional help
- Collaborated with the other teaching assistant to create effective teaching strategies and materials, which helped students more fully understand topics covered in the main lecture

Business Statistics Teaching Assistant, USU, Logan, UT 01/2014–05/2014

- Enthusiastically taught business statistics to two classes of more than 30 students each
- Communicated complex subjects in clear, understandable ways to help struggling students achieve success

Math Refresher Course Teachering Assistant, USU, Logan, UT 01/2014

- Helped prepare students to succeed in their upcoming math classes

College Algebra Teaching Assistant, USU, Logan, UT 08/2013–12/2013

- Clearly taught college algebra in two classes of about 30 students each
- Responsibly proctored exams and quizzes

PROFESSIONAL EXPERIENCE

Graduate Research Assistant, USU, Logan, UT **05/2018–08/2018**

- Compared existing confidence distribution methods and created new confidence distribution methods for meta–analysis to determine which is best suited for meta–analyses with rare events and heterogeneity
- Presented these findings at USU's Together We Teach Conference

Statistical Consultant, USU, Logan, UT **01/2018–02/2018**

- Analyzed complex survey and web application data for a student finishing his PhD at Purdue University
- Provided the results and discussion sections of the student's dissertation in an approachable and clear manner

English Consultant, USU, Logan, UT **04/2016–11/2016**

- Edited professional papers and applications for a statistics faculty member who speaks English as her second language

Statistician, Intermountain Healthcare, Salt Lake City, UT **05/2015–08/2015**

- Efficiently extracted data from large databases to conduct complex analyses, which uncovered ways Intermountain Healthcare could be more efficient, more consistent, and more aware of patient needs
- Analyzed sensitive company data and presented results in a user–friendly form, facilitating decision making for managers and medical professionals

Business Analyst, Journal Technologies, Logan, UT **05/2014–12/2014**

- Extracted data from a database to produce complex, interactive reports per customer requirements that communicated data clearly and concisely
- Quickly learned to create personalized, automated documents with the company's proprietary software
- Communicated clearly and effectively with remote customers via verbal and written correspondences

PROFESSIONAL MEMBERSHIPS

- American Statistical Association 2014–present

PROFESSIONAL DEVELOPMENT

- Attend the Graduate Training Series by USU, 05/2014–present
- Attended a two–day teacher training workshop by USU's Mathematics and Statistics department, 08/2018
- Participated in weekly teacher training seminars by USU's Mathematics and Statistics department, 01/2018–04/2018

COMPUTER SKILLS

Programming Languages:	R, SAS, Python, C++, and C
Skills:	Data Cleaning, Web Data Extraction, Regular Expressions, and HPC Clusters
Database Tools:	MS SQL & SQL Server Reporting Services and Oracle SQL Developer & PL/SQL
Operating Systems:	Linux and Windows
Markup Languages:	LaTeX, HTML, and XML

AWARDS

- Graduate PhD Researcher of the Year Award, 2019
- Excellence in Teaching Award, 2019
- USU Student Research Symposium Outstanding Graduate Oral Presentation Award, 2018
- Academic Excellence Award, 2016
- USU Presidential Doctoral Research Fellowship for graduate degree, 2014
- Academic Excellence Award, 2014
- Data Ninja Award for being the top student in a management information systems course, 2014
- USU Presidential Scholarship for undergraduate degree, 2011
- Regents' Scholarship for Exemplary Academic Achievement, 2011

SERVICE AND VOLUNTEER WORK

- Volunteer as a mathematics and statistics tutor for undergraduates, 2011–present
- Served as a USU ambassador to potential graduate students, 2014–2018
- Reviewed USU Undergraduate Research and Creative Opportunity grants, 2017–2018
- Served as a judge for undergraduate research presentations during USU's Student Research Symposium, 2017–2018
- Freely consulted with a Biological Engineering PhD student to help her with the statistics in her dissertation, 2017