METHODOLOGY FOR ANALYZING PRECLINICAL EXPERIMENTS

By

TRAYMON EVERETT BEAVERS

A dissertation submitted to the

School of Graduate Studies

Rutgers, The State University of New Jersey

In partial fulfillment of the requirements

For the degree of

Doctor of Philosophy

Graduate Program in Statistics and Quantitative Biomedicine

Written under the direction of

Javier Cabrera

And approved by

_____

_____

_____

_____

New Brunswick, New Jersey

May 2019

ABSTRACT OF THE DISSERTATION

Methodology for Analyzing Preclinical Experiments

by TRAYMON EVERETT BEAVERS

Dissertation Director:

Javier Cabrera

The dissertation examines three distinct methodologies for analyzing data yielded from preclinical experiments:

1) Big data has created new challenges for data analysis, particularly in the realm of creating meaningful groups or clusters of data or classification. Clustering techniques, such as K-means or hierarchical clustering, based on pairwise distances of N objects, are popular methods for performing exploratory analysis on large datasets such as these. Unfortunately, these methods are not always possible to apply to big data due to memory or time constraints generated by calculations of order N^2. A work-around is to take a random sample of the large dataset and perform the clustering technique with the reduced dataset; however, this is not a foolproof solution since the structure of the dataset, particularly at the edges of the dataset, is not guaranteed to be maintained. In this chapter we will propose a new solution through the concept of "data nuggets". These data nuggets reduce a larger dataset into a small collection of nuggets of data, each containing a center, weight, and a scale parameter. Once the data is re-expressed as data nuggets, we may apply algorithms that compute standard statistical methods, such as principal

components analysis (PCA), clustering, classification, etc. We apply the methodology of data nuggets to the analysis of a dataset from flow cytometry in Biopharmaceutical research. This was conducted by performing weighted PCA and weighted K-means clustering on a dataset containing millions of observations (B-cells), and the objective was to find clusters that characterize cells according to which proteins are active on their surfaces. An R package was also developed to conduct these methods.

2) There are many cases in preclinical drug discovery when experiments are repeated but not precisely replicated regarding treatment arms. Further, full datasets are not always immediately accessible, leaving analysts to rely on summary measures such as sample mean and standard error. If one is only interested in comparing two treatment arms at a time, meta-analysis is a useful tool; however, when one applies this method they are limited to only comparing two of the potentially numerous treatment arms at a time. Further, information from experiments lacking these two treatment arms is not used. Mixed treatment comparisons meta-analysis, also known as network meta-analysis, can be used instead to compare all available treatment arms at once. This chapter will explain, explore, and compare two frequentist methods that exist to apply network meta-analysis. We focus on sets of experiments with designs typically found in preclinical experiments. We also use simulations to compare network meta-analysis results to those given by mixed-effect linear models for these types of experiments. An R package was also developed to perform both methods of network meta-analysis.

3) Power calculations for hypothesis tests play a critical role in conducting both clinical and non-clinical trials. Many programs exist to calculate the power for popular hypothesis tests, such as Student's t-test for hypothesis tests analyzing continuous data or

the log-rank test for hypothesis tests analyzing survival data. Calculating the power for hypothesis tests analyzing ordinal categorical data can be much more complicated. For data such as this, observations are given in the form of scores on a scale with a small range, typically between three and five points. The data is assumed to be distributed according to a multinomial distribution which can depend on many parameters. We propose a simple yet effective method for defining alternative multinomial distributions and performing power calculations by creating and shifting quantiles of the standard normal distribution. We offer simulation results and apply the method to a dataset. An R package was also developed to use this method.

ACKNOWLEDGEMENT

I would like to thank Dr. Javier Cabrera for his patience and guidance throughout my entire graduate career. I would like to thank Dr. John Kolassa for blessing me with the opportunity to earn a Ph.D. in statistics during his tenure as graduate director. I would like to thank Dr. David Tyler for presenting me a firm foundation of knowledge in statistics during my first two years as a graduate student throughout his many classes. I would like to thank Dr. Mariusz Lubomirski for helping to provide proper context for many of the ideas found in this dissertation. I would like to thank Dr. Gail Ferstandig Arnold for providing with me the opportunity to be a part of the joint Quantitative Biomedicine program. I would like to thank Davit Sargsyan for his assistance both professionally as a fellow statistician and in general as a trusted friend and mentor. Finally, I would like to thank Stephanie Beavers for always supplying me with the strength, motivation, and unconditional love and support I needed to complete this arduous task.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

INTRODUCTION

In this dissertation we will examine three different methodologies for analyzing data presented in preclinical drug discovery. The first chapter details a new algorithm created to form representative data from large datasets. In this new era of "Big Data", large datasets are often initially analyzed using clustering methods such as K-means clustering or hierarchical clustering after greatly reducing the dataset by drawing a random sample. These random samples are not always capable of maintaining the overall structure of the entire dataset, particularly at the edges of the data. We introduce a new algorithm which reduces the dataset into "data nuggets" which are nuggets of data used to describe the observations of the entire dataset. Then, weighted principal components can be used to examine the structure of the reduced data and a weighted K-means clustering algorithm can be used to form clusters of the data. We detail these algorithms, provide simulation results comparing the effects of forming clusters of data nuggets using weighted K-means clustering instead of K-means clustering, and apply the method to a preclinical dataset from a pharmaceutical company.

The second chapter examines and compares two different frequentist methods for conducting network meta-analysis. In the field of preclinical drug discovery many compounds are compared in multiple experiments and this method can be very useful since it combines the information from all experiments to produce effect size estimates. This method can also deliver results pertaining to compound comparisons which were never truly made using indirect evidence. Although these two methods are supposed to produce equivalent results, we produce a counterexample created from preclinical drug

discovery data which shows otherwise. We also offer simulation results to assess the effectiveness of the method.

The final chapter details a new method proposed for producing power calculations and finding the optimal sample sizes necessary for conducting future experiments for which all observations are given by ordinal categorical data where the ordinal scale contains only a small range of values. In experiments such as these the data is assumed to be distributed according to multinomial distribution which can depend on many parameters. As such, generating alternatives by manipulating these parameters haphazardly can prove to be an overwhelming task. The method proposed generates alternatives by shifting the parameters in a uniform, controlled manner. We detail this method, provide simulation results to assess its effectiveness, and apply the method to a preclinical dataset.

Chapter 1: Data Nuggets: A Method for Reducing Large Datasets While Preserving Data Structure

1.1 Introduction

Extremely large datasets, sometimes known as “Big Data”, are common in most areas of research and business including the pharmaceutical industry (Srinivasan, 2018). An example of how large datasets arise can be found in experiments where scientists are interested in measuring the abundance of proteins that are expressed on the surface of cells and they collect a sample of millions of cells to conduct the experiment. There are many ways to measure the abundance of these proteins (Amaratunga, Cabrera, & Shkedy, 2014).

One such method is flow cytometry (Jahan-Tigh, Ryan, Obermoser, & Schwarzenberger, 2012). Different antibodies which correspond to the proteins of interest are chosen. Each of them labeled according to a distinct fluorescence. Cells are then stained with these fluorescent antibodies and sent through a flow cytometer. In this flow cytometer, cells flow toward a laser and when cells pass by the laser, they absorb the laser’s energy and emit a wavelength of light specific to each antibody, and therefore each protein. The level of expression of the proteins of interest on the surface of the cells can then be measured. These experiments produce datasets which are very high in dimension. Clustering techniques can be used to attempt to find interesting groups of cells hidden within the data.

As a motivating example, suppose a drug is being developed to interact with T-cells in the liver. As a starting point, the scientists in charge of the experiment want to know the levels of expression of certain proteins. They perform the experiment with 10

million cells and use 10 different fluorescent antibodies to collect the levels of expression for each protein the fluorescent antibody corresponds to for each cell. The goal of the experiment is to find out if there are any groups of cells for which the level of expression of any protein is very high or very low.

The most typical method would be to apply a clustering technique to the dataset, such as K-means clustering or hierarchical clustering; however, a dataset as large as this would require far too many resources, such as computational memory and time. We propose a different method which instead reduces the 10 million data points into a smaller collection of "data nuggets". All the individual data points coalesce into many data nuggets, while still retaining the structure of the data. After this a weighted form of K-means clustering can be used to configure the data nuggets into various clusters.

Section 1.2 introduces notation and provides a brief overview of popular clustering methods and the issues that arise when attempting to use them to cluster large datasets. Section 1.3 describes the algorithm for creating data nuggets and the algorithm for creating clusters using weighted K-means clustering. This section also provides simulation results comparing the accuracy of K-means clustering to weighted K-means clustering for clustering data nuggets generated from a dataset with binary variables and compares data nuggets to the support points given by Mak and Joseph in (Mak & Joseph, 2018). Section 1.4 applies the algorithm to a dataset from a preclinical experiment. Section 1.5 briefly describes a package created to use the method, and some future work that could be done concerning this method.

1.2 Review of Popular Clustering Methods

We will introduce notation by generalizing our motivating example. Suppose the scientists perform their experiment with *N* observations (T-cells), where *N* is in the millions, and they are measuring the level of expression of *P* different proteins. Let **X** be the matrix containing the information pertaining to the levels of expression of each protein for each cell so that:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_{N-1} \\ \mathbf{x}_N \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1(P-1)} & x_{1P} \\ x_{12} & x_{22} & \cdots & x_{2(P-1)} & x_{2P} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{(N-1)1} & x_{(N-1)2} & \cdots & x_{(N-1)(P-1)} & x_{(N-1)P} \\ x_{N1} & x_{N2} & \cdots & x_{N(P-1)} & x_{NP} \end{bmatrix}$$

Where $\mathbf{x}_n$ is the row vector containing the cell surface marker levels for the n^{th} T-cell and x_{np} is the level of expression of protein p for the n^{th} T-cell, for $n = 1,2,\ldots,N$ and $p = 1,2,\ldots,P$.

Once again, the goal of the experiment is to find out if there are any groups of T-cells for which any proteins show a high or low level of expression. We can search for these groups by placing the T-cells into *K* different clusters and then finding out if any of the proteins have a particularly weak or strong level of expression in any of the clusters. The memory usage and computation needed is of the order of N^2 (where N is in the millions or greater) for typical clustering techniques. In general, clustering is performed in one of two ways: through K-means clustering or hierarchical clustering (Cabrera & McDougall, 2002).

1.2.1 K-means Clustering

Arguably the most popular clustering technique is K-means clustering. K-means clustering is applied by first initializing centers of the K different clusters and then minimizing the total within cluster sum of squares. Let $\boldsymbol{\mu}_{01}, \boldsymbol{\mu}_{02}, \ldots, \boldsymbol{\mu}_{0K}$ be the K $P \times 1$ vectors that are chosen to represent the initial centers of clusters $L_1, L_2, \ldots, L_K$, respectively. This can be done either by choosing K observations from the dataset (randomly or selected by the user) or by choosing K random points in $\mathbb{R}^P$.

Next, all observations are assigned to whichever initial center is closest to it according to some distance metric, typically Euclidean distance. Finally, the center of each cluster is updated to by calculating the mean of all observations within the cluster and replacing the current center of each cluster with these new means, $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K$. These steps are then repeated until the clusters reach a point where the total within cluster sum of squares, given by:

$$\sum_{i=1}^{K} \sum_{\mathbf{x} \in L_i} (\mathbf{x} - \boldsymbol{\mu}_i)'(\mathbf{x} - \boldsymbol{\mu}_i)$$

is minimized. The steps taken to reach this minimization of total within cluster sum of squares depends on the algorithm being used (Morissette & Chartier, 2013).

Lloyd's algorithm attempts to minimize the total within cluster sum of squares by reassigning observations to the closest cluster center and updating the cluster centers thereafter (Lloyd, 1982). Forgy's algorithm works in the same manner as Lloyd's algorithm but the total within cluster sum of squares is calculated assuming the

observations are distributed according to a continuous probability distribution instead of a discrete probability distribution (Forgy, 1965).

MacQueen's algorithm is also similar to LLoyd's algorithm, but instead of only recalculating the centers after all the observations have been reassigned to the cluster with the nearest center, this calculation is performed after every single reassignment (MacQueen, 1967). The Hartigan & Wong algorithm's shares the same objective of minimizing the total within cluster sum of squares; however, this algorithm will not always assign observations to the cluster with the nearest center (Hartigan & Wong, 1979). Instead, sometimes observations will be reassigned to a cluster with a farther center if doing so minimizes the total within cluster sum of squares. To accomplish this feat, the Hartigan & Wong algorithm must store both the cluster with the closest center and the cluster with the second closest center for each observation.

K-means clustering can be performed in R using the function *kmeans* (R Core Team, 2018). The Hartigan & Wong algorithm is the default method for this function, although the LLoyd, Forgy, and MacQueen algorithms are also available. An example of a dataset clustered with K-means clustering using the Hartigan & Wong algorithm is provided in Figure 1.

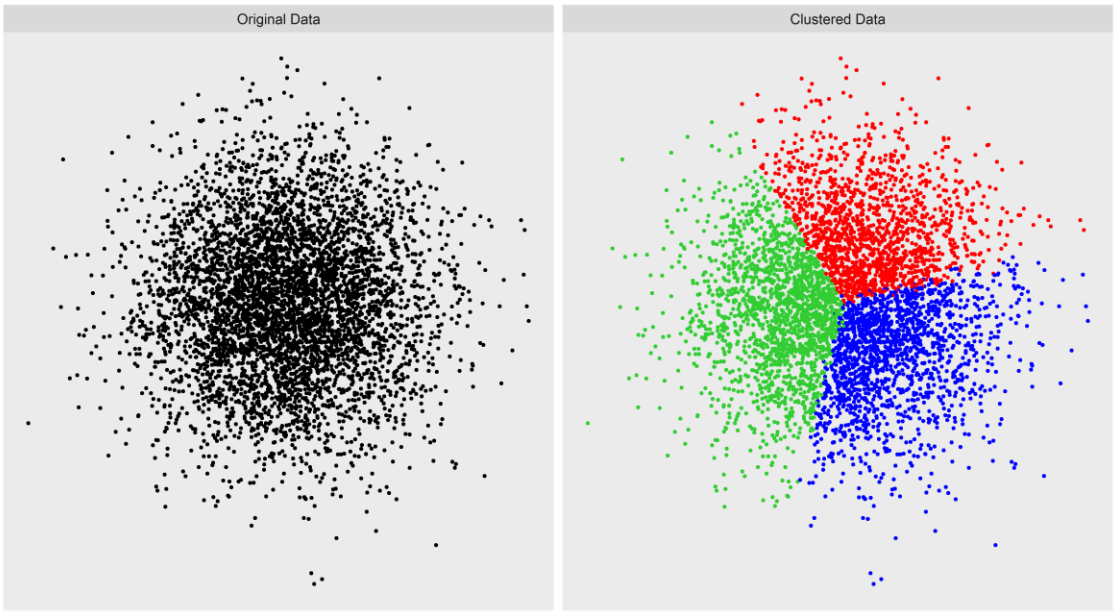

Figure 1: K-means Clustering Example

1.2.2 Hierarchical Clustering

Another popular method of clustering is hierarchical clustering. This method of clustering can be applied in one of two ways: either agglomerative, where all observations begin as their own cluster and then combine together to form the desired number of clusters; or divisive, where all observations are assigned to the same cluster and then split into the desired number of clusters (Rokach & Maimon, 2005).

The combining or dividing of clusters is done according to a measure of similarity or dissimilarity, respectively, calculated using a distance metric. Once again, this distance metric is usually Euclidean distance. There are three different styles of methods, each determined by the way this measure is calculated.

Single-link clustering methods regard the distance between clusters L_i and L_j to be equal to the shortest distance between any observation assigned to L_i and any

observation assigned to L_j. Complete-link clustering methods regard the distance between clusters L_i and L_j to be equal to the greatest distance between any observation assigned to L_i and any observation assigned to L_j. Finally, Average-link clustering methods regard the distance between clusters L_i and L_j to be equal to the average distance between any observation assigned to L_i and any observation assigned to L_j. Hierarchical clustering produces a dendrogram, which can then be cut with either consideration for how many clusters are desired or the desired level of similarity or dissimilarity according to the dendrogram.

Hierarchical clustering can be performed in R using the function *hclust*. Since these methods depend entirely on the distances between observations, the distance matrix for the data matrix serves as input instead of the data matrix itself. An example of a dataset clustered with hierarchical clustering is provided in Figure 2. The dendrogram used to form the clusters was cut at the third split in order to have the same number of clusters as the example with K-means clustering.

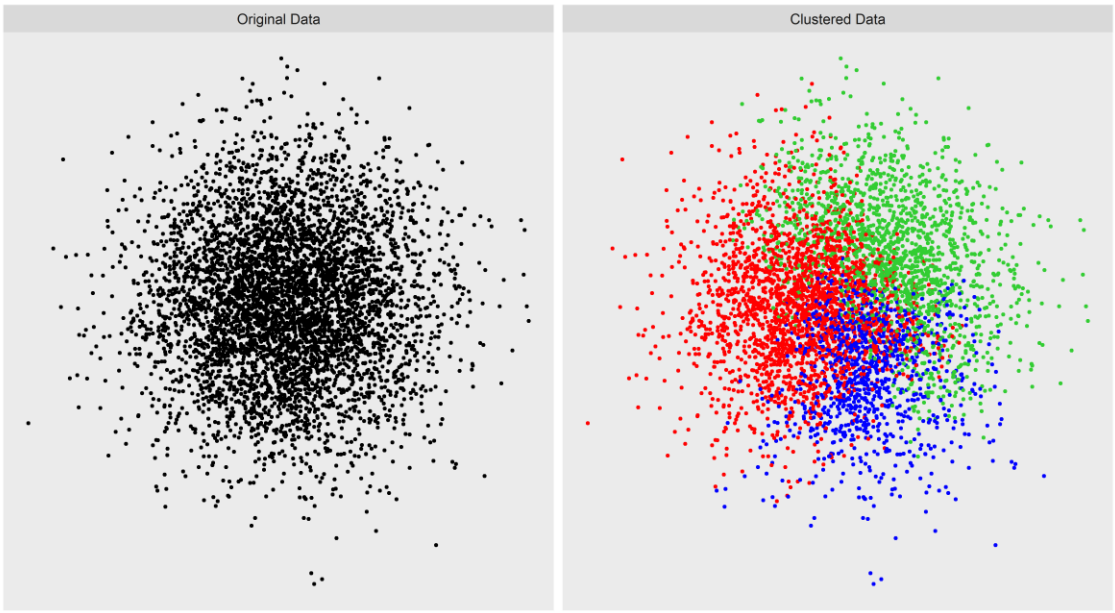

Figure 2: Hierarchical Clustering Example

1.2.3 Limitations for Large Datasets

Both methods of clustering have limitations for very large datasets. For K-means clustering, the final cluster assignments heavily depend on the initial choice of cluster centers (Ayramo & Karkkainen, 2006). A clear remedy for this is to choose multiple initial cluster centers, conduct the algorithm of choice for each, and choose the results which minimizes the total within cluster sum of squares. For datasets with a large number of observations many initial centers may need to be attempted. For the LLoyd, Forgy, and MacQueen algorithms the time cost is high in R, which may lead the user to sacrifice the number of initial cluster centers they choose to evaluate.

On the other hand, the Hartigan & Wong algorithm may fail to finish running for large datasets because the memory cost necessary to store the closest cluster assignment and the second closest cluster assignment for each observation is too high. This is also the

case for hierarchical clustering methods which may not even have a chance to begin because the distance matrix cannot be formed for datasets that are too large. More specifically, the largest distance matrix R would be able to store is one derived from a data matrix with 46,340 observations (since the most entries a matrix can have in R is 2,147,483,647), assuming the machine it is running on has enough memory space.

A common solution to this problem has been to retrieve a random sample of the data and use a clustering algorithm on this reduced dataset. The intuition is that if the sample is sufficiently large, the data structure of the sampled data should match the data structure of the entire data. Unfortunately, this intuition does not always hold. Further, since a distance matrix is needed for hierarchical clustering, the random sample may need to be reduced to a very small amount of observations when compared to the entire dataset.

Another possible solution is to reduce the large dataset to a set of only a few data points which are chosen to represent the dataset as a whole. Gosh, Cabrera, et al. introduce a notion of representative data with weights in (Ghosh, et al., 2016). Mak and Joseph introduce a method for producing “Support Points” which together form a representative dataset in (Mak & Joseph, 2018); however, support points are not suitable to find clusters on the edges of the distribution where the density of points is much lower. To avoid the time and memory constraints of using full datasets (or large random samples), the pitfalls associated with massive data reduction, and the lack of focus on the edges of the data structure, we propose using data nuggets.

1.3 Data Nuggets

In this section we will describe how to generate data nuggets from a large dataset. The process of creating data nuggets is inspired by the idea of using a p-dimensional grid that encompasses the entire dataset to partition the observations into M equally sized cells. The centers of these cells would form the data nugget centers, the amount of observations from the dataset which exist in these cells would form the data nugget weights, and the trace of the covariance matrices for the observations within each cell would from the data nugget scales.

When both p and M are low this is a relatively simple feat, but when either p or M is large the amount of computational resources required becomes unrealistic. A more feasible option would be to use observations already within the dataset as centers. This can be done by choosing observations in the dataset which are as equally spaced apart as possible. Then all the remaining observations are assigned to the data nugget they are closest too according to a distance metric. We detail the algorithm to create data nuggets below.

ALGORITHM 1: Create M data nuggets given: **X**, an $n \times p$ data matrix; N, the number of observations to randomly sample from **X**; M, the number of data nuggets to create; and D, a distance metric.

1. Randomly sample N observations from **X**.
2. Create a distance matrix for these observations using the given distance metric D.
3. Find the smallest non-diagonal distance in the matrix. Label the observations with this distance between them as y_1 and y_2.

4. Find the distance to the nearest neighbor of y_1 excluding y_2, d_{y_1}, and the distance of the nearest neighbor of y_2 excluding y_1, d_{y_2}.
5. If $d_{y_1} < d_{y_2}$, remove y_1 from the random sample. Otherwise remove y_2 from the random sample.
6. Create a new distance matrix with the remaining random sample.
7. Repeat steps 2 through 6 for $N - M$ iterations. The observations in the random sample of M observations that remain are the centers of the data nuggets. Let data nugget j have center $\mathbf{c}_j$ for $j = 1, 2, \dots, M$.
8. For each $\mathbf{x}_i$, $i = 1,2, \dots, n$, assign $\mathbf{x}_i$ to the data nugget such that $D(\mathbf{x}_i, c_j)$ is minimized over j. Let n_j be the number of observations assigned to data nugget j, and let $W_j = n_j$ be the weight of data nugget j for $j = 1,2, \dots, M$.
9. Re-center all the data nuggets by choosing $\mathbf{c}_j$ to be the mean of all the observations assigned to data nugget j.
10. Finally, let $S_j = tr\left(Cov(\mathbf{X}_j)\right)$ be the scale of data nugget j when $n_j > 1$, where $\mathbf{X}_j$ is the submatrix of observations from $\mathbf{X}$ which belong to data nugget j. When $n_j = 1$, $S_j = 0$.

Figure 3 compares the amount of data structure maintained after reducing a bivariate dataset of 15,601 to a simple random sample of 2,000 observations versus reducing that same bivariate dataset to 2,000 data nuggets. This comparison is carried out using density plots. The dataset is a mixture of data derived by sampling a large number of observations from two independent standard normal distributions and combining these observations with other observations which create a “smile” that is hidden inside the

random noise. This smile can only be observed by using a density plot. The data nuggets were created by reducing a random sample of 10,000 observations from this dataset to 2,000 data nuggets using **ALGORITHM 1** with Euclidean distance as the chosen distance metric.

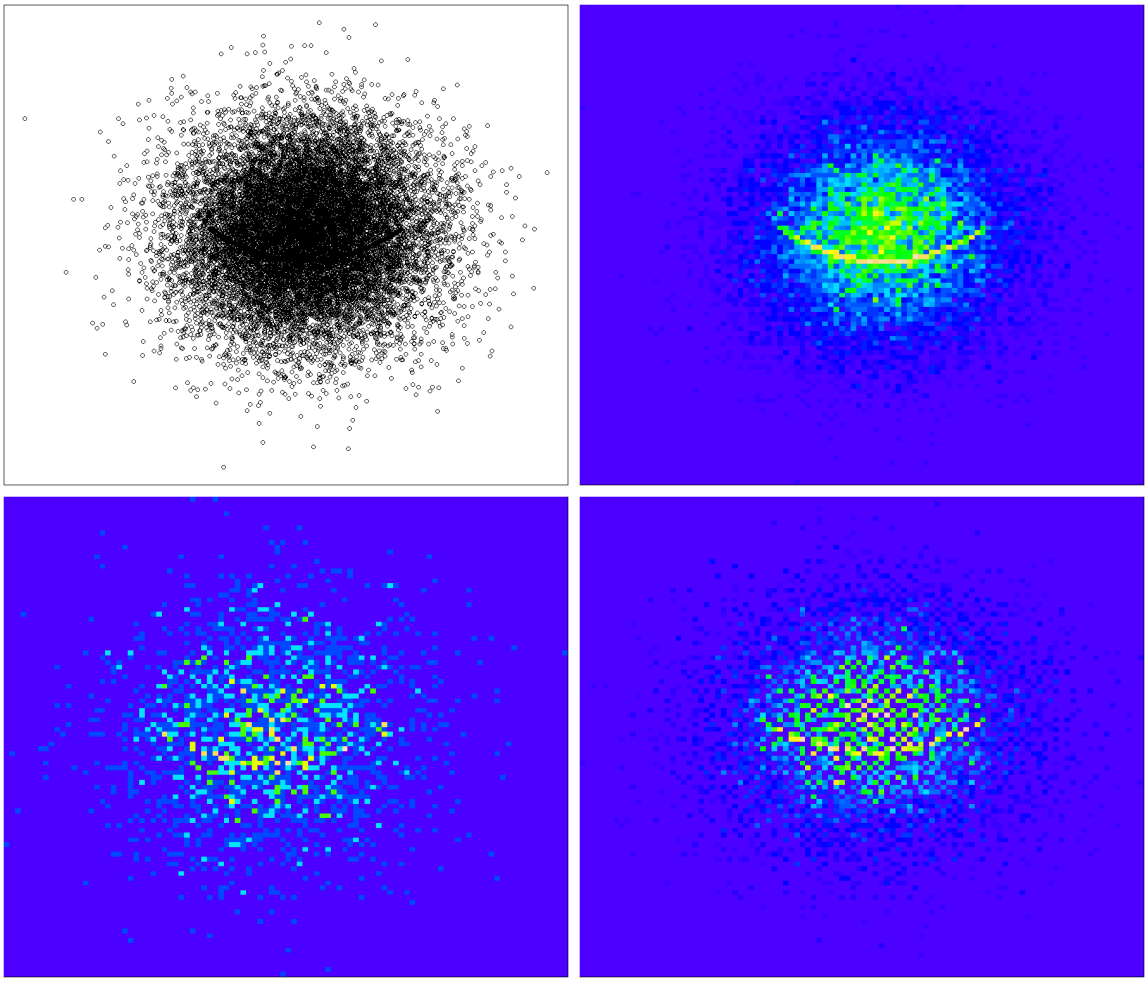

Figure 3: Comparing Density Plots for Random Sample and Data Nuggets

The density plots for the entire dataset and the random sample are created by dividing the area of the original scatterplot into a 100×100 grid and counting the number of observations in the dataset which fall inside each cell of the grid. The cells are then colored on a gradient according to how many observations are in the cell. Cells with

a low number of observations produce cool colors like blue or light green while cells with a higher number of observations produce hot colors such as yellow or orange.

The density plot for data nuggets is produced in a similar manner but with a slight modification. Once again, the area of the original scatterplot is divided into a 100×100 grid. However, instead of using the number of data nuggets that exist within the cell, the sum of the weights of the data nuggets that exist within the cell is used. Then the cells are colored accordingly. The first row of plots is a scatterplot of the entire dataset of 15,601 observations beside its corresponding density plot. The lower left plot is the density plot corresponding to a scatterplot of a random sample of 2,000 observations from the dataset. The lower right plot is the density plot corresponding to a scatterplot of the 2,000 data nuggets.

As shown in the figure, the density plot for the entire dataset clearly shows a thin smile inside the ball of random noise. The density plot for the random sample faintly produces the smile, but there are random gaps dispersed throughout the smile and the tails of the smile are not recognizable. This density plot also produces a large amount of random noise surrounding the smile. The density plot for the data nuggets shows a much more distinct and visible smile. While there are still a few gaps in the smile, they are more or less equally spaced. Further, the tails of the smile are clearly visible, and the amount of random noise is much more concentrated around the smile, matching what is seen in the density plot for the entire dataset.

Two natural questions arise regarding the generation of data nuggets: how large of a random sample should be initially drawn, and how many data nuggets must be chosen before a desirable set is retrieved? Regardless of the random sample size chosen, we

recommend creating enough data nuggets to reduce the random sample to 20% of its original sample size. Next, check if the minimum distance between any two points is large enough. What constitutes a large enough minimum distance is a subjective matter, but the goal should always be to make sure that there are enough data nuggets to maintain the shape of the structure of the data while also ensuring that there are not too many data nuggets crowded in the center of the structure. If these two goals are not realized for a selection of data nuggets, they can always be reduced further.

The data nuggets can also be refined according to the scale parameter and the minimum number of observations that must be assigned to a data nugget. The purpose of this method is to provide each data nugget with a more common level of variation. Data nuggets are refined as detailed in the algorithm below.

ALGORITHM 2: Refine M data nuggets given: S^{tol}, a scale tolerance value; and m, the minimum number of observations that a data nugget must contain as a result of this algorithm.

1. Obtain the median of the nonzero scale parameters for the M data nuggets, η.
2. Create B, a list of all data nuggets with scale parameters larger than $S^{tol}\eta$.
3. For every data nugget $j \in B$:
    3.1. If data nugget j contains greater than $2m$ observations, split data nugget j into two new data nuggets using K-means clustering.
    3.2. If either of the two new data nuggets created in step 3.1 contain less than m observations, delete these two data nuggets and retain data nugget j. Otherwise, delete data nugget j and remove data nugget j from B.

4. Repeat steps 2 and 3 until B is empty or step 3 is completed without any data nuggets being removed from B.

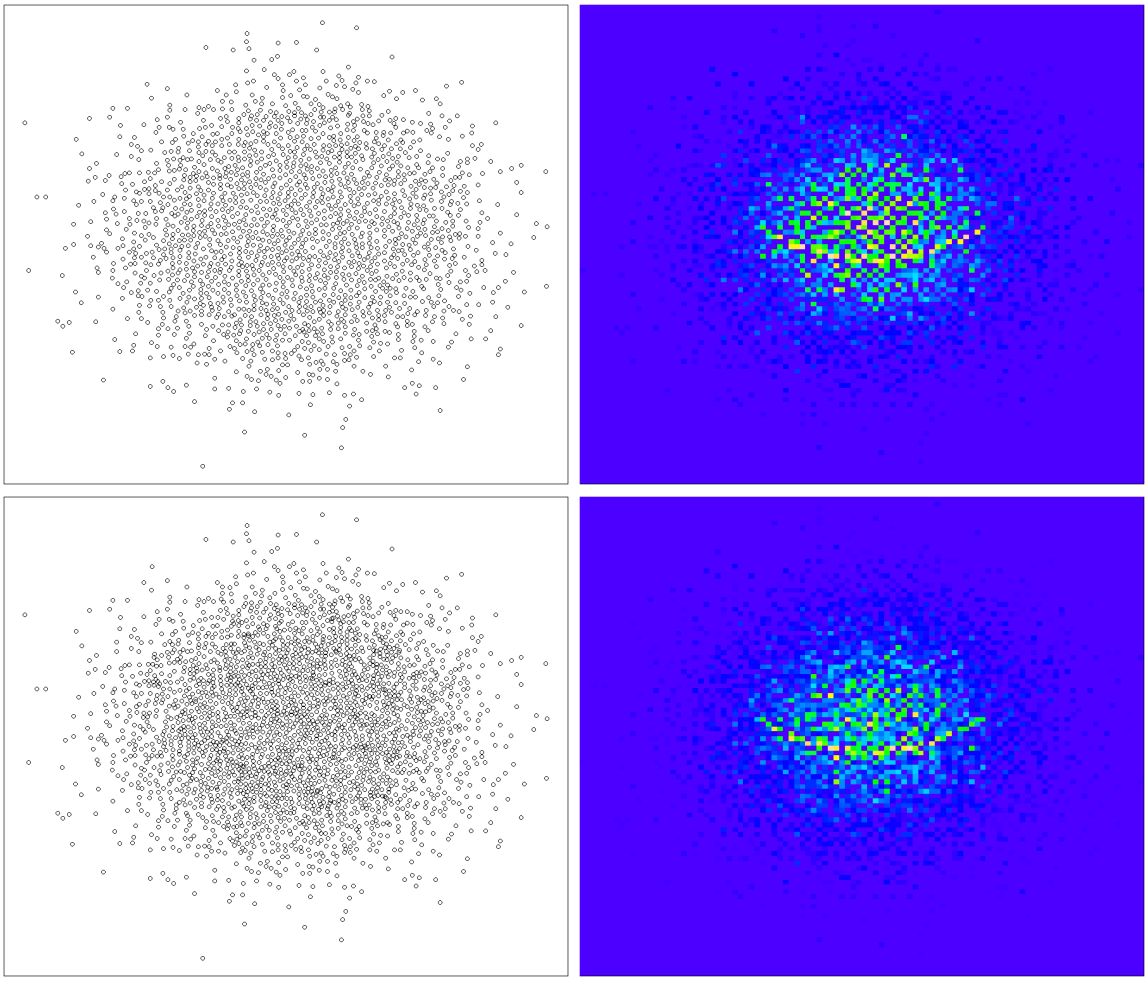

Figure 4: Comparing Density Plots for Original and Refined Data Nuggets

Figure 4 shows the same "smile" dataset described above, this time comparing the original 2,000 data nuggets refined to 2,562 data nuggets with scale tolerance value $S^{tol} = 1$ and minimum number of observations $m = 2$ after using ALGORITHM 2. The first row of plots is the scatterplot of the original 2,000 data nuggets beside its corresponding density plot, and the second row of plots is the scatterplot of the refined 2,562 data nuggets beside its corresponding density plot. The density plot for the 2,562

data nuggets has a much more consistent smile with fewer gaps, and the ball of random noise is slightly more concentrated around the smile.

After the data nuggets are created, modified versions of common statistical techniques can be applied to them. In this chapter we explore weighted K-means clustering and weighted principal component analysis. In both of these methods we ignore the scale parameter of the data nuggets, and instead focus on using the weight parameter. This is done because we believe that the internal variability of the data nuggets is miniscule. This belief is formalized in **PROPOSITION 1**.

PROPOSITION 1: Let Σ be the sample covariance matrix for $N \times P$ data matrix **X**. Let $\mathbf{X}_k$ be the collection of k data nuggets meant to form a representative dataset of **X**. Let $\mathbf{S}_k = \sum_{i=1}^{k} W_i \mathbf{c}_i \mathbf{c}_i'$. For large k, $S_k \approx \Sigma$.

If **PROPOSITION 1** is true, then given enough data nuggets the amount of variability within the dataset is preserved when the dataset is reduced without accounting for the individual amount of variability within each data nugget. As such, this variability can be ignored when applying statistical techniques to the data nuggets.

1.3.1 Weighted K-means Clustering for Data Nuggets

We now introduce a weighted K-means clustering algorithm that can be used to form clusters of these data nuggets. It is worth noting that other weighted K-means clustering methods have been developed. An example is an algorithm that can be used for analyzing social networks (Liu & Xu, 2014). This algorithm is designed for the purpose of finding clusters of nodes in a social network where weights are assigned to the edges

that connect the nodes. The weights of the edges are described as the "intimacy" level between the two nodes that the edge connects.

In our algorithm, the weights of each data nugget are a measure of how many observations from the original dataset are contained in the data nugget. We describe a method of weighted K-means clustering to form clusters of data nuggets with the algorithm below.

ALGORITHM 3: Conduct weighted K-means to form clusters of data nuggets given: M data nuggets; K, the number of clusters to be created; $\mathbf{w}$, the $M \times 1$ vector containing the weight of each data nugget; and D, a distance metric.

1. Choose K data nuggets (randomly or user-selected) to be the initial cluster centers, $\boldsymbol{\mu}_{01}, \boldsymbol{\mu}_{02}, \ldots, \boldsymbol{\mu}_{0K}$, of K clusters, $L_1, L_2, \ldots, L_K$, respectively.
2. Assign each of the M data nuggets to the cluster with the closest cluster center according to distance metric D.
3. Recalculate the cluster centers $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K$ as the mean of all the data nuggets within clusters $L_1, L_2, \ldots, L_K$, respectively.
4. For $i = 1, 2, \ldots, M$ data nuggets:
   4.1. Retrieve the cluster assignment for data nugget i, $L_{(i)}$, and the current total weighted within cluster sum of squares, $W_{(i)}$.
   4.2. Reassign data nugget i to every cluster in $\{L_1, L_2, \ldots, L_K\} \setminus L_{(i)}$ and calculate the total weighted within cluster sum of squares for each of the $K - 1$ possible reassignments, $\{W_1, W_2, \ldots, W_K\} \setminus W_{(i)}$, where W_k is the total weighted within

cluster sum of squares when data nugget i is assigned to cluster L_k for $k = 1,2,\dots,K$ and $k \neq (i)$.

4.3. If $W_k = min(\{W_1, W_2, \dots, W_K\}) < W_{(i)}$, reassign data nugget i to cluster L_k and recalculate the cluster centers $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_K$ as the mean of all the data nuggets within clusters $L_1, L_2, \dots, L_K$, respectively.

5. Repeat step 4 until step 4 is completed without executing step 4.3.

The outcome of **ALGORITHM 3** can be improved by repeating the algorithm with multiple choices for the initial centers chosen in step 1. The clustering assignments which minimize the total weighted within cluster sum of squares would then be chosen as the clustering configuration. Further, it may take an extremely long time for the algorithm to converge. As such, there could be a limit placed on the number of times step 4 is executed before the algorithm ends.

There could also be a threshold placed on the improvement of the total weighted within cluster sum of squares before ending the algorithm. For example, if the $W_{(i)}$ found in step 4.2 is only 10^{16} greater than the W_k found in step 4.3, this may be evidence that ending the algorithm at this point will provide almost identical results to those that would be yielded from convergence.

To illustrate the usefulness of **ALGORITHM 3** we conducted a simulation using binary data. This simulated dataset is meant to mimic a list of 300,000 patients and whether they suffer from a list of ten conditions. We separate the data into three clusters based on the set of conditions these patients suffer from. Let L_1, L_2, and L_3 represent the three clusters. Let p be the probability of having any condition. Let,

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \end{pmatrix} \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \\ y_9 \\ y_{10} \end{pmatrix} \mathbf{z} = \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \\ z_6 \\ z_7 \\ z_8 \\ z_9 \\ z_{10} \end{pmatrix}$$

where $x_i \sim Bin(1,1-p)$ for $i = 1,2,3,4,5$, $x_i \sim Bin(1,p)$ for $i = 6,7,8,9,10$, $y_i \sim Bin(1,p)$ for $i = 1,2,3,4,5$, $y_i \sim Bin(1,1-p)$ for $i = 6,7,8,9,10$, and $z_i \sim Bin(1,p)$ for $i = 1,2,\dots,10$. L_1 is formed by sampling 100,000 observations of form x, L_2 is formed by sampling 100,000 observations of form y, and L_3 is formed by sampling 100,000 observations of form z. These 300,000 observations are then placed together in a dataset.

Since these observations are binary, there are at most $2^{10} = 1024$ possible unique observations. Each of these observations will represent a data nugget, and the weight of the data nugget is the number of times this data nugget appears in the dataset divided by 300,000.

Using the Hartigan &Wong algorithm for K-means clustering and the weighted K-means clustering method given by **ALGORITHM 3**, we assign each data nugget to one of three clusters. Next, for every observation in the original dataset, we append the cluster assigned to its corresponding data nugget for each method. Finally, we find the proportion of correct cluster assignments for every possible permutation of cluster assignments and choose the cluster configuration which produces the best result for each method.

We repeat this process for 100 iterations and find the mean proportion of data nuggets correctly reassigned to their proper cluster for each method. We compare the two methods for various choices of p. 5 random sets of centers are used to initialize the algorithms for each iteration and the cluster configurations with the least within cluster sum of squares and weighted within cluster sum of squares for the K-means clustering method and the weighted K-means clustering method, respectively, are chosen. The simulation results are given in Table 1.

p	K-means (H-W)	Weighted K-means
0.80	0.5624	0.9063
0.82	0.5545	0.8925
0.84	0.5808	0.9163
0.86	0.6095	0.9116
0.88	0.6486	0.9104
0.90	0.6597	0.9454

Table 1: Correct Cluster Classification Simulation Results

It is clear to see that the weighted K-means algorithm outperforms the Hartigan-Wong algorithm in terms of the mean proportion of correct reclassification. This provides proper motivation to believe that using weighted K-means clustering to form clusters of data nuggets provides better results than ignoring the weights and simply using K-means clustering. As for choosing the best number of clusters, popular methods such as constructing silhouettes (Rousseeuw, 1987) or calculating the gap statistic (Tibshirani, Walther, & Hastie, 2001) are used to detect the optimal number of clusters. These methods and others, can be updated to include information concerning the data nugget weights.

1.3.2 Data Nuggets vs. Support Points

In section 1.2.3 we mentioned another method of producing representative data called “Support Points” given by Mak and Joseph. The goal of data nuggets and support points are the same on the surface: to create a small dataset that represents the large dataset it comes from. That being said, the resulting representative datasets differ greatly in terms of producing the correct quantiles corresponding to the highest and lowest percentiles of the probability distributions they are meant to represent.

This is by design in the case of support points, since they are defined as a set of M points in the dataset which has the best goodness of fit to the underlying distribution governing the dataset in terms of energy distance as defined in (Székely & Rizzo, 2013). This definition forces more observations to be chosen that exist near the center of the data, ultimately forsaking observations that exist at the edge of the data.

Data nuggets on the other hand are designed to avoid this problem. Since the algorithm that creates data nuggets chooses observations to delete based on how close they are to other observations, observations on the edges of the data are safe from elimination and are guaranteed to remain as a data nugget center. The information concerning the fact that this observation is at the edge of the data is not lost—this information is contained in the weight parameter of the data nugget.

We now produce the results of a simulation which examines this difference in a 1-Dimensional setting. The simulation was conducted by randomly sampling 100,000 observations from a standard normal distribution. Let this random sample of observations be $\hat{z}$. 100 support points and 100 data nuggets are then created from $\hat{z}$. The support points were generated using the *support* package created by Mak (Mak S., 2018). The data

nuggets were generated with **ALGORITHM 1** by choosing $\mathbf{X} = \hat{z}$, $N = 1{,}000$, $M = 100$, and D to be the Euclidean distance metric. The data nuggets are then ordered by their centers in an ascending fashion.

We then compute the quantiles corresponding to the 95^{th}, 96^{th}, 97^{th}, 98^{th} and 99^{th} percentiles for each representative dataset. The quantiles for the support points are calculated in the typical fashion; however, calculating the quantiles for the data nuggets requires a more thoughtful process. First, a linear regression model is fit with the cumulative sums of the data nugget weights (each divided by 100,000) as the predictor variable and the data nugget centers as the response variable. Then, .95, .96, .97, .98, and .99 are plugged into the resulting regression equation to produce the quantiles corresponding to those percentiles for the data nuggets. Finally, the true quantiles for a standard normal distribution are subtracted from the quantiles calculated for each method to calculate the bias of each quantiles for each method.

This simulation was repeated for 1000 sets of $\hat{z}$ and Figure 5 shows the results. For each method, the box plots represent the distribution of quantile estimate bias for each corresponding percentile. It is clear to see that support points perform poorly in terms of bias compared to data nuggets for calculating the quantiles at the upper tail of the normal distribution. Since the bias for the quantiles given by the data nuggets prove consistent across the percentiles, there may be a simple correction constant that can be applied to each quantile to eliminate the bias entirely.

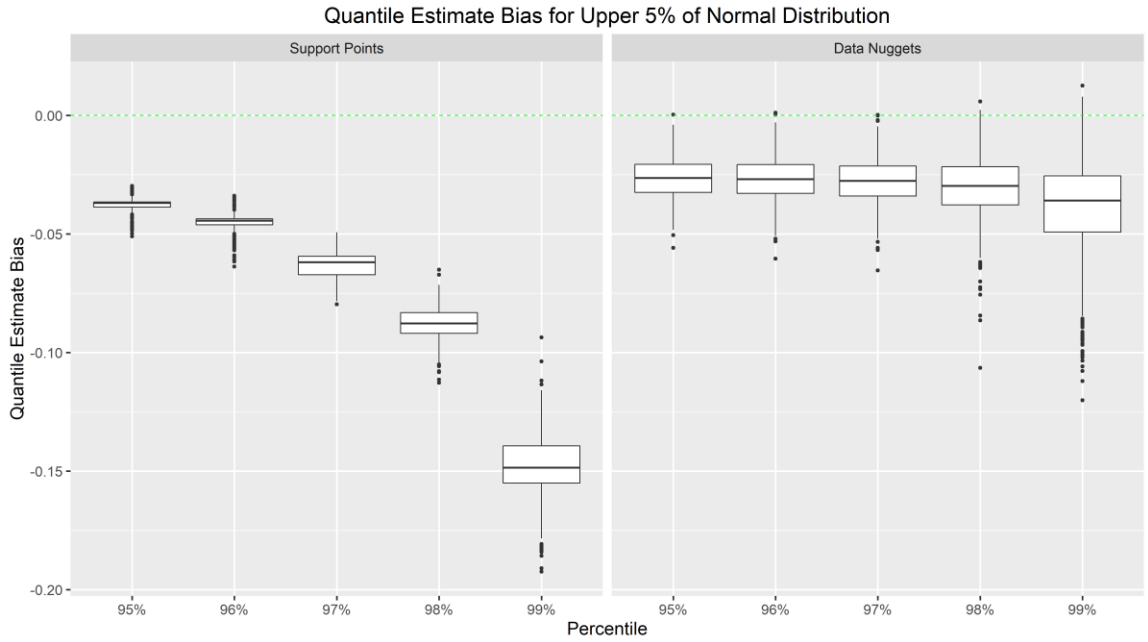

Figure 5: Quantile Bias Simulations Results

1.4 Application to Preclinical Dataset

In this section we apply the method of creating data nuggets and clustering them using weighted K-means clustering to a continuous dataset from a pharmaceutical company. The data set consists of over 1 million observations. Each observation corresponds to a B-cell and the columns correspond to the level of expression of nine different proteins on the surface of these B-cells. We will label these proteins A through I.

The scientists conducting the experiments are interested in knowing if there are clusters of cells which express extremely high or low levels of expression of certain proteins. High levels of expression of these proteins correspond to the activation of these cells to perform certain functions. These functions can play an extremely vital role in helping the immune system of the organism the cell belongs to.

We begin by taking a random sample of 10,000 observations from the dataset and reducing this sample to 2,000 data nuggets using **ALGORITHM 1**. We then refined the data nuggets using **ALGORITHM 2** with scale tolerance value $S^{tol} = 2$ and minimum number of observations $m = 2$, which resulted in 3,577 data nuggets.

Pairwise combinations of the first, second, and third principal components of the entire dataset are shown beside the same pairwise combinations of the first, second, and third weighted principal components of the initial 2,000 data nuggets and the refined 3,577 data nuggets are given in Figure 6 for a comparison of the resulting data structures. All principal components were found using the *wpca* function from the R package *aroma.light* (Neuvial, Bengtsson, & Speed, 2010). Note that the weights entered for the *wpca* function when creating the principal components for the entire dataset were all equal to 1.

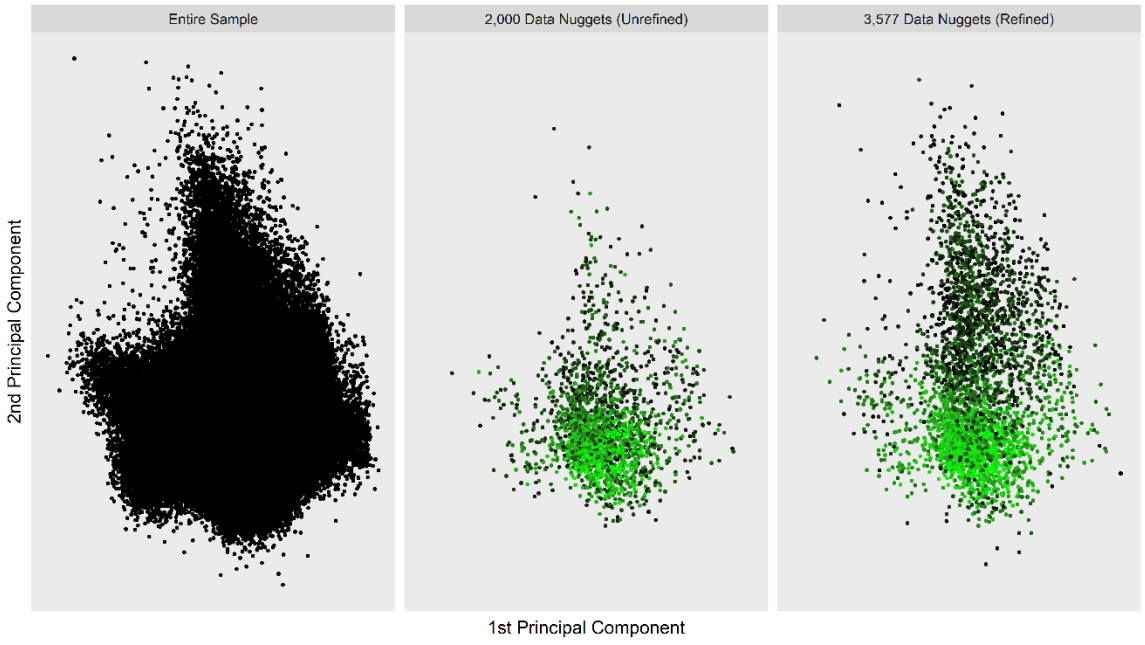

Figure 6: PCA Plots of Entire Dataset vs WPCA Plots of Data Nuggets

The principal components for the data nuggets were weighted according to the weights of the data nuggets. The color of each data nugget corresponds to the weight of the data nugget. Lighter green indicates a large weight while darker green indicates a low

weight. Observe how the structure of the data regarding the first three principal components is moderately recovered with the original 2,000 data nuggets and strongly recovered with the 3,577 refined data nuggets. Recall that the original dataset contains over 1 million observations, so the fact that less than 1% of these observations can be chosen and still produce a relatively strong representation of the structure of the data is noteworthy.

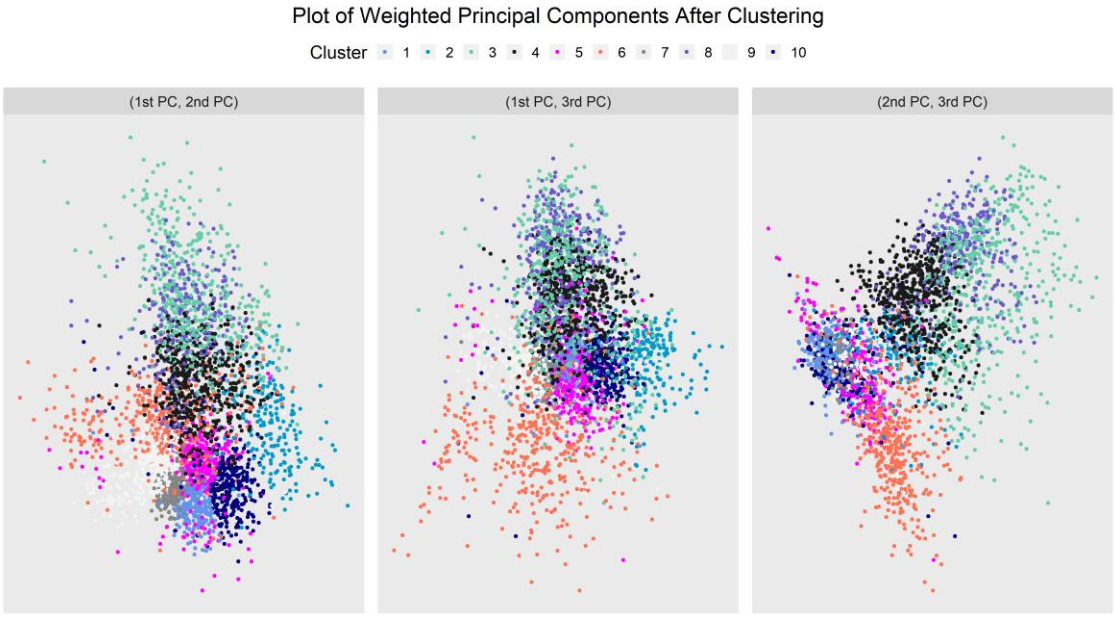

Figure 7: Weighted PCA Plots of Data Nuggets Separated Into 10 Clusters

Next, we configure the data nuggets into 10 clusters using **ALGORITHM 3** to perform weighted K-means clustering. We use 10 initial centers and choose the cluster configuration with the least weighted within cluster sum of squares. The pairwise combinations of the first, second, and third weighted principal components of the 3,577 refined data nuggets separated into 10 clusters is shown in Figure 7. Finally, we created box plots for each cluster which summarize the level of expression of the observations

within the cluster for each protein to search for whether any clusters show any visually significant levels of expression of any proteins. These box plots are given in Figure 8.

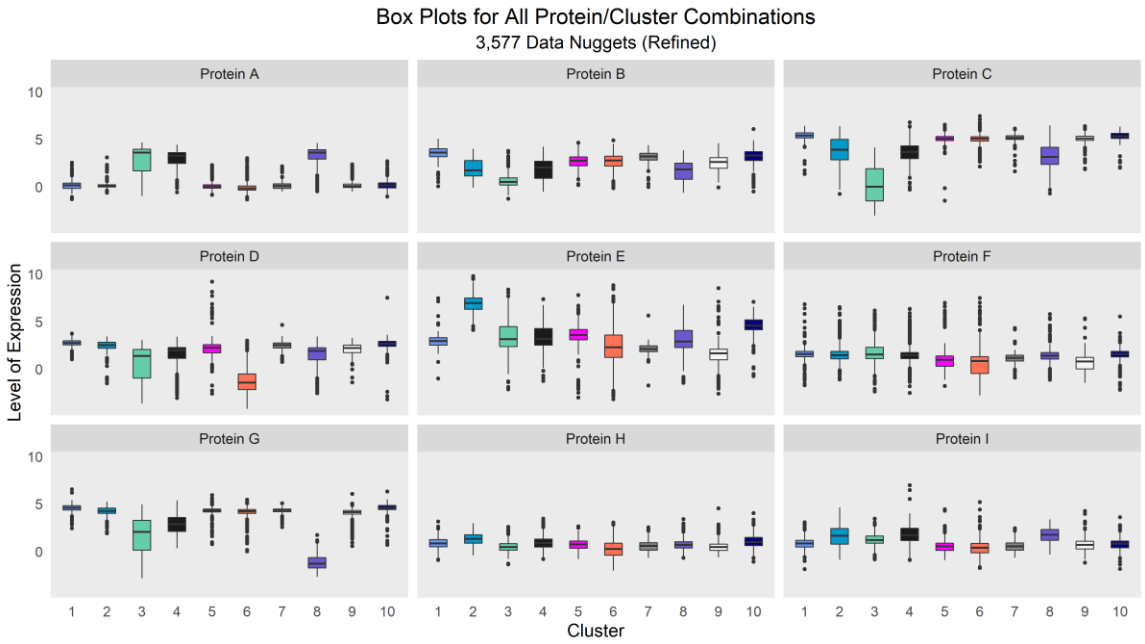

Figure 8: Levels of Expression for Each Protein and Cluster Combination

While for proteins B, F, H, and I there is not much difference between the clusters, there is a noticeable difference between clusters for the remaining proteins. The cells in clusters 3, 4, and 8 show a high level of expression of protein A. Clusters 6 and 8 show a low level of expression of protein D and G, respectively. Cluster 3 shows a low level of expression of protein C, although there is a high level of variability. Finally, cluster 2 shows a high level of expression of protein E.

1.5 Discussion

We have detailed a method for reducing “Big Data” using data nuggets. We also offer a weighted K-means algorithm to cluster these data nuggets and provide simulation results which show that this algorithm outperforms the K-means clustering algorithm for

data nuggets yielded from binary data. We also displayed the distinction between data nuggets and support points in the context of quantile bias at the tails of probability distributions using a simulation, showing that there is a greater level of bias when these quantiles are calculated with support points. Finally, we applied this method to a preclinical dataset and presented the results.

The R package *datanugget* has been developed to incorporate the methods described in this paper. It includes functions for generating, refining, and clustering data nuggets using weighted K-means clustering. While the runtime for these functions are slow in R, work could be done in the future to re-write the programs in the programming language C. If this were done the runtimes would improve a drastic amount.

Future work could be done to show how well the data nuggets work when other mainstream statistical techniques are applied. We have already shown how well data nuggets can work when unsupervised methods such as principal components and clustering are applied. Another unsupervised method of interest that could be applied is projection pursuit (Friedman & Tukey, 1974). The efficacy of data nuggets could also be observed in the context of supervised methods such as logistic regression and linear regression.

In the case of logistic regression, the response for each data nugget would be the number of “successful” and “unsuccessful” observations contained in the data nugget. In the case of linear regression, the response for each data nugget would be the mean of the responses of the observations contained in the data nugget and weighted least squares regression could be applied. The weight of each data nugget (potentially combined with

the variance of the response variable for each data nugget) would be used as the weight in the regression model.

An important area of improvement for this method would be to find the optimal number of data nuggets. As the method currently stands the amount of subjective calibration is undesirable. Simulations involving large classified continuous datasets could also be created to determine how much better weighted K-means clustering performs compared to K-means clustering of data nuggets in a continuous setting.

Another area of interest is showing that the results of the simulation in Section 1.3.2 hold for higher dimensions. Work could also be done to provide a correction for the constant bias for estimating the quantiles with data nuggets. Research into asymptotic results regarding how well the probability distribution can be returned through estimation of the mean and covariance of data nuggets generated from a random sample of this probability distribution as the number of data nuggets increases to infinity would be useful as well.

Chapter 2: A New Understanding of Network Meta-Analysis Regarding Experiments with Small Sample Sizes

2.1 Introduction

Often in the field of non-clinical discovery multiple experiments are conducted to answer the same question using various experiment designs. For example, suppose a pharmaceutical company is attempting to develop a drug to decrease the amount of white blood cells (WBC's) in the liver. When the scientists assigned to this problem begin, they develop compound B and compare it to the standard treatment, compound A. Not satisfied with their results, they conduct seven more experiments; some with the same design, some with a similar design, and others with an entirely different design. The evolution of these experiment designs is shown in Figure 9.

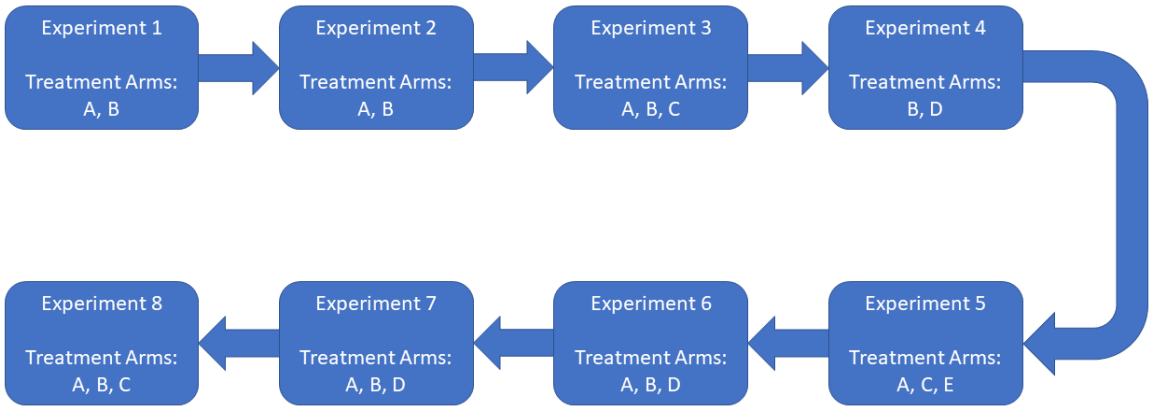

Figure 9: Example Experiment Designs

Cases such as this where only a few experiments are performed to compare many treatment arms using few observations per treatment arm are typical in preclinical drug discovery. Finally, unwilling to publish the entire dataset for each experiment in their reports, the scientists only publish a collection of summary measures: sample mean

amount of WBC's in the liver after treatment, the corresponding standard error, and the sample size for each treatment arm in each experiment.

The most basic approach would be to fit a fixed effect or random effect meta-analysis model to compare two treatment arms at a time to generate an effect size estimate (Borenstein, 2009). The effect size estimate chosen depends on the type of data collected (e.g. mean difference or sample mean difference for continuous data, log odds ratio for binary data, etc.). Using this approach, results from experiments with the same design or similar designs are combined to create an effect size estimate; however, experiments with completely different designs cannot be used in the meta-analysis since they do not have one (or perhaps either) of the treatment arms being compared.

Mixed treatment comparison meta-analysis, also known as network meta-analysis, is a method used to generate effect size estimates given individual summary measures or contrast summary measures, such as sample means or sample mean differences for continuous responses, respectively, along with the standard error of these estimates (Lumley, 2002). Using network meta-analysis we can generate results for all possible treatment arm comparisons using the summary measures from all eight different experiments.

Section 2.2 explains how to use two existing frequentist methods: generalized least squares (GLS) (Lu, Welton, Higgins, White, & Ades, 2011); or electrical network theory (ENT) (Rücker & Schwarzer, 2012), to conduct network meta-analysis. Section 2.3 provides two examples to demonstrate that the two methods are equivalent in some cases but not equivalent in others. Section 2.4 provides results of simulations created to assess how network meta-analysis using summary measures compares to mixed effects

linear models using full datasets. Section 2.5 discusses the implications of these simulations, the R package created to conduct network meta-analysis, and some future work that can be done in this area.

2.2 Review of Two Frequentist Methods to Perform Network Meta-Analysis

To illustrate how each method works we will continue with the fictional example described in Section 2.1. First, we will introduce general notation. Let N be the number of experiments conducted, J be the number of treatment arms, G be the number of unique experiment designs, and T_g, for $g \in \{g_1, g_2, \dots, g_G\}$, be the number of treatment arms in design g. In this example we are working with continuous data, but these methods can also be used to analyze summary measures yielded from experiments with binary data or survival data (Schwarzer, Carpenter, & Rücker, 2015).

Let the triplet $(\bar{x}_{ij}, s_{ij}, n_{ij})$ for $i = 1,2, \dots, N, j = 1,2, \dots, p_i$ be the sample mean, standard error, and sample size for treatment arm j in experiment i, where p_i is the total number of treatment arms in experiment i. Finally, let $\mathbf{I}_m$ be the identity matrix with dimension $m \times m$. Table 2 displays the data layout for individual summary measures.

For continuous data, every treatment arm has a true mean response associated with it. For example, compound A has the true mean response, θ_A. In other words, θ_A is the true mean amount of WBC's produced in the liver as a result of using compound A. For any combination of two treatment arms, there exists an effect size parameter which represents the difference between the true mean responses of these treatment arms. For example, we say $d_{AB} \equiv \theta_A - \theta_B$ is the effect size parameter for comparing compound A to compound B.

Experiment	Treatment Arm	Sample Size	Sample Mean	Standard Error
1	A	n_{11}	$\bar{x}_{11}$	s_{11}
1	B	n_{12}	$\bar{x}_{12}$	s_{12}
2	A	n_{21}	$\bar{x}_{21}$	s_{21}
2	B	n_{22}	$\bar{x}_{22}$	s_{22}
3	A	n_{31}	$\bar{x}_{31}$	s_{31}
3	B	n_{32}	$\bar{x}_{32}$	s_{32}
3	C	n_{33}	$\bar{x}_{33}$	s_{33}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
8	A	n_{81}	$\bar{x}_{81}$	s_{81}
8	B	n_{82}	$\bar{x}_{82}$	s_{82}
8	C	n_{83}	$\bar{x}_{83}$	s_{83}

Table 2: Individual Summary Measures Dataset

The goal of network meta-analysis is to estimate all $\binom{J}{2}$ effect size parameters. Note that these effect size parameters can be divided into two mutually exclusive sets: direct comparisons, which are treatment arm comparisons that were observed in at least one of the N experiments (e.g. A vs. B) and indirect comparisons, which are treatment arm comparisons that were not observed in any of the N experiments (e.g. C vs. D).

There are two assumptions necessary for conducting network meta-analysis. First, all the experiments used in the analysis are conducted independently of one another. Second, any indirect comparison of treatment arms can be formed by using direct comparisons. For example, $d_{BC} \equiv \theta_B - \theta_C = (\theta_A - \theta_C) - (\theta_A - \theta_B) \equiv d_{AC} - d_{AB}$. This is known as the transitivity assumption (Schwarzer, Carpenter, & Rücker, 2015). This is also known as the assumption of consistency (Snedecor, Patel, & C. Cappeller, 2014).

For the transitivity assumption to be used, of the set of treatment arms being used in the experiments must be "connected" in a specific way. This specification can be best understood in the context of graph theory. If the collection of treatment arms used in the

N experiments is viewed as a set of vertices, V, and the collection of direct comparisons is viewed as a set of edges, E, then a network graph, $G = (V, E)$, can be created. All edges correspond to an effect size parameter for the two vertices they connect. See Figure 10 for the network graph that would be created for this example. Further, a subgraph of G is any graph G_0 such that $G_0 = (V_0, E_0)$ where $V_0 \subset V$ and $E_0 \subset E$.

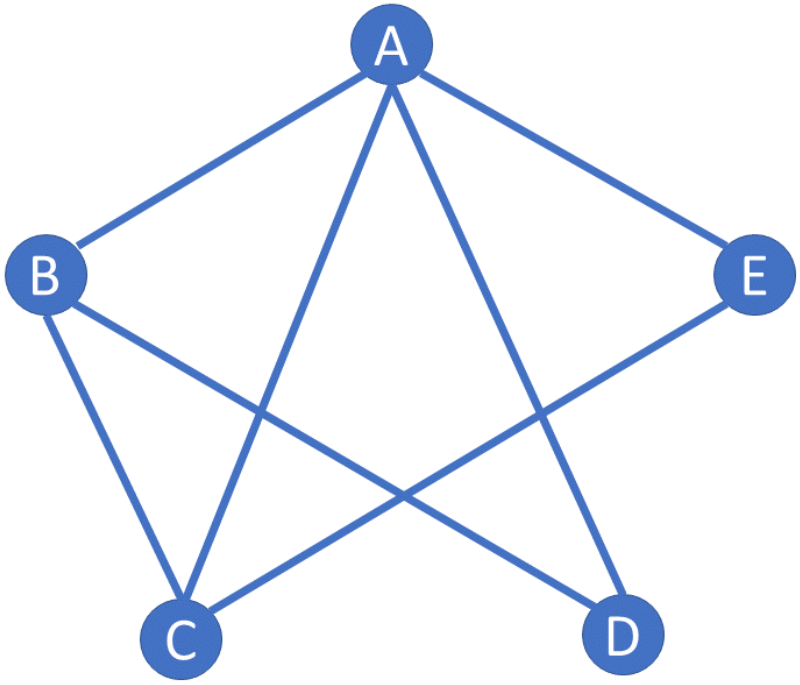

Figure 10: Example Network Graph

Note that a graph $G = (V, E)$ has $|V|$ vertices and $|E|$ edges. A spanning tree is a subgraph formed by a collection of all $|V|$ vertices and $|V| - 1$ edges such that all $|V|$ vertices in the graph are connected by edges. It is a necessary condition that a spanning tree can be formed in the network graph in order to employ network meta-analysis (Valkenhoef, et al., 2012). In our example, a spanning tree is given by the subgraph formed with all five vertices and the four edges attached to the A vertex.

There are methods to check whether the transitivity assumption holds for all the effect size estimates (Lu & Ades, 2006; Dias, Welton, Caldwell, Ades, & Hougaard, 2010; Jansen & Cope, 2012; White, Barrett, Jackson, & Higgins, 2012; Katsanos, 2014; Lu & Ades, 2006). Additionally, there are methods to view how information and evidence "flows" through the network of treatment arms to arrive at the final estimates (König, Krahn, & Binder, 2013). It is worth noting that these models can be expanded to use a random effect approach. This section of the paper will focus on defining the fixed effect approach.

2.2.1 Generalized Least Squares

In our example, $G = 5$ since there are five unique experiment designs: $\{AB, ABC, BD, ACE, and\ ABD\}$. For designs where $T_g = 2$, meta-analysis is performed in the typical fashion. We will show how experiments with design AB would be fit according to a fixed effect meta-analysis model using individual summary measures (i.e. the dataset type given in Table 2) to generate the initial effect size estimates and their respective standard errors. Let d_i^g and $\left(s_i^g\right)^2$ denote the sample mean difference and pooled variance, respectively, of experiment i with design g.

First the sample effect sizes, along with their pooled variances for all comparisons within this design must be calculated:

$$d_1^{AB} = \bar{x}_{11} - \bar{x}_{12}, \qquad (s_1^{AB})^2 = \frac{n_{11}s_{11}^2 + n_{12}s_{12}^2}{n_{11} + n_{12}}$$

$$d_2^{AB} = \bar{x}_{21} - \bar{x}_{22}, \qquad (s_2^{AB})^2 = \frac{n_{21}s_{21}^2 + n_{22}s_{22}^2}{n_{21} + n_{22}}$$

Next the sample effect sizes are used in conjunction with their pooled variances to provide a weighted effect size estimate and its respective variance. Let W_i^g denote the weight of experiment i with design g.

$$W_i^{AB} = \frac{1}{(s_i^{AB})^2} \text{ for } i = 1,2$$

$$\hat{d}_{AB} = \frac{W_1^{AB} d_1^{AB} + W_2^{AB} d_2^{AB}}{W_1^{AB} + W_2^{AB}}, \qquad s^2_{\hat{d}_{AB}} = \frac{1}{W_1^{AB} + W_2^{AB}}$$

In general, for a collection of M two arm experiments, all with the same design g, the weighted effect size estimate and its variance is given by:

$$\hat{d}_g = \frac{\sum_{i=1}^{M} W_i^g d_i^g}{\sum_{i=1}^{M} W_i^g}, \qquad s^2_{\hat{d}_g} = \frac{1}{\sum_{i=1}^{M} W_i^g}$$

For designs where $T_g > 2$, meta-analysis is performed in an analogous fashion with matrix multiplication to generate the weighted effect size estimates. We will now show how experiments with design ABC would be fit according to a fixed effect meta-analysis model using the dataset type given in Table 2 to generate the effect size estimates and their respective standard errors.

First a vector of size $T_g - 1$ sample effect sizes must be generated with respect to a treatment arm which we will denote as the baseline choice. This is done to ensure that the design matrix in the linear model will be full rank (Lu, Welton, Higgins, White, & Ades, 2011). This vector is meant to be analogous to the scalars produced in two arm meta-analysis. This baseline choice can be any treatment arm in design g, but it will be

discussed in Section 2.3 how certain baseline choices can optimize the results with respect to sum of squared errors (SSE).

Let d_i^g be the vector of sample effect sizes for experiment i with design g with respect to a baseline choice. Further, let $\underline{d}_i^{c_1c_2}$ and $\left(\underline{s}_i^{c_1c_2}\right)^2$ be the sample mean difference and pooled variance, respectively, comparing treatment arm c_1 to c_2 in experiment i.

For our example, we will choose compound A to be the baseline choice, so our vectors of sample effect sizes will all involve compound A:

$$\mathbf{d}_3^{ABC} = \begin{pmatrix} \underline{d}_3^{AB} \\ \underline{d}_3^{AC} \end{pmatrix} = \begin{pmatrix} \bar{x}_{31} - \bar{x}_{32} \\ \bar{x}_{31} - \bar{x}_{33} \end{pmatrix}$$

$$\mathbf{d}_8^{ABC} = \begin{pmatrix} \underline{d}_8^{AB} \\ \underline{d}_8^{AC} \end{pmatrix} = \begin{pmatrix} \bar{x}_{81} - \bar{x}_{82} \\ \bar{x}_{81} - \bar{x}_{83} \end{pmatrix}$$

Next, we create a covariance matrix meant to be analogous to the pooled variance in two arm meta-analysis. Let $\mathbf{V}_i^g$ denote the covariance matrix of experiment i with design g. This matrix is constructed by using the pooled variances of the sample mean differences contained in $\mathbf{d}_i^g$ along with the sample variance of the sample mean of the treatment arm chosen as the baseline choice:

$$\mathbf{V}_3^{ABC} = \begin{bmatrix} \left(\underline{s}_3^{AB}\right)^2 & s_{31}^2 \\ s_{31}^2 & \left(\underline{s}_3^{AC}\right)^2 \end{bmatrix}, \mathbf{V}_8^{ABC} = \begin{bmatrix} \left(\underline{s}_8^{AB}\right)^2 & s_{81}^2 \\ s_{81}^2 & \left(\underline{s}_8^{AC}\right)^2 \end{bmatrix}$$

where

$$\left(\underline{s}_i^{AB}\right)^2 = \frac{n_{i1}s_{i1}^2 + n_{i2}s_{i2}^2}{n_{i1} + n_{i2}}, \left(\underline{s}_i^{AC}\right)^2 = \frac{n_{i1}s_{i1}^2 + n_{i3}s_{i3}^2}{n_{i1} + n_{i3}} \quad \text{for } i = 3,8$$

This formulation for the covariance matrix is based on asymptotic results given in (Higgins & Whitehead, 1996). For small sample sizes we instead must assume that $\{\theta_A, \theta_B, \theta_C, \theta_D, \theta_E\}$ are mutually independent, which is a reasonable assumption if the treatment arms have no obvious relationship with each other.

To create the weight matrices, we simply invert the variance matrices. Let W_i^g denote the weight matrix of experiment i with design g.

$$\mathbf{W}_i^{ABC} = \left(\mathbf{V}_i^{ABC}\right)^{-1}$$

The weighted effect size estimate vector and its covariance matrix are calculated as follows:

$$\hat{\mathbf{d}}_{ABC} = (\mathbf{W}^{ABC})^{-1}(\mathbf{W}_3^{ABC}\mathbf{d}_3^{ABC} + \mathbf{W}_8^{ABC}\mathbf{d}_8^{ABC}) = \begin{pmatrix} \hat{d}_{AB} \\ \hat{d}_{\mathrm{AC}} \end{pmatrix}$$

$$\mathbf{V}^{ABC} = (\mathbf{W}^{ABC})^{-1} = \begin{bmatrix} s^2_{\hat{d}_{AB}} & \widehat{Cov}(\hat{d}_{AB}, \hat{d}_{\mathrm{AC}}) \\ \widehat{Cov}(\hat{d}_{AB}, \hat{d}_{\mathrm{AC}}) & s^2_{\hat{d}_{AC}} \end{bmatrix}$$

where

$$\mathbf{W}^{ABC} = \mathbf{W}_3^{ABC} + \mathbf{W}_8^{ABC}$$

In general, for a collection of M experiments with more than two arms, all with the same design g, the weighted effect size estimate vector and its respective covariance matrix is given by:

$$\hat{\mathbf{d}}_g = (\mathbf{W}^g)^{-1}\sum_{i=1}^{M}\mathbf{W}_i^g\,\mathbf{d}_i^g\,, \qquad \mathbf{V}^g = (\mathbf{W}^g)^{-1}$$

where

$$\mathbf{W}^g = \sum_{i=1}^{M} \mathbf{W}_i^g$$

Once the above steps are completed for all G designs, all of the effect size estimates must be combined into, $\mathbf{y}^{\text{gls}}$, a $T \times 1$ vector, and all of their respective variances must be combined into, $\mathbf{V}^{\text{gls}}$, a $T \times T$ block diagonal matrix where $T = \sum_{g=1}^{G}(T_g - 1)$.

Continuing our example, we form $\mathbf{y}^{\text{gls}}$ and $\mathbf{V}^{\text{gls}}$. Note that the effect size estimates with centered dots in the subscript come from designs with more than two treatment arms.

$$\mathbf{y}^{\text{gls}} = (\hat{d}_{AB}\hat{\mathbf{d}}_{ABC}, \hat{d}_{BD}, \hat{\mathbf{d}}_{ACE}, \hat{\mathbf{d}}_{ABD})$$

$$\mathbf{y}^{\text{gls}} = \left(\hat{d}_{AB}, \hat{d}_{AB\cdot}, \hat{d}_{AC\cdot}, \hat{d}_{BD}, \hat{d}_{AC\cdot\cdot}, \hat{d}_{AE\cdot\cdot}, \hat{d}_{AB\cdots}, \hat{d}_{AD\cdots}\right)'$$

$$\mathbf{V}^{\text{gls}} = (\mathbf{W}^{\text{gls}})^{-1} = \text{diag}\left(s^2_{\hat{d}_{AB}}, \mathbf{V}^{ABC}, s^2_{\hat{d}_{BD}}, \mathbf{V}^{ACE}, \mathbf{V}^{ABD}\right)$$

Next, we will begin forming the linear model that will be solved using the generalized least squares solution. Let $\mathbf{d}$ be the $M \times 1$ $(M \leq T)$ vector containing the unique effect size parameters estimated in $\mathbf{y}^{\text{gls}}$. Continuing our example:

$$\mathbf{d} = (d_{AB}, d_{AC}, d_{AD}, d_{AE}, d_{BD})'$$

$\mathbf{d}$ is then decomposed into two sub-vectors: $\mathbf{d}_{\text{b}}$, a $(J-1) \times 1$ basic parameter vector formed with the effect size parameters corresponding to a spanning tree in the network graph, and $\mathbf{d}_{\text{f}}$, an $(M - J + 1) \times 1$ functional parameter vector formed with the remaining effect size parameters.

$$\mathbf{d} = (\mathbf{d}_{\mathrm{b}}, \mathbf{d}_{\mathrm{f}})'$$

$$\mathbf{d}_{\mathrm{b}} = (d_{AB}, d_{AC}, d_{AD}, d_{AE})'$$

$$\mathbf{d}_{\mathrm{f}} = (d_{BD})'$$

The effect size parameters in the functional parameter vector are all linear combinations of the effect size parameters in the basic parameter vector, as shown below for this example:

$$\mathbf{F}\mathbf{d}_{\mathrm{b}} = [-1 \quad 0 \quad 1 \quad 0]\begin{pmatrix} d_{AB} \\ d_{AC} \\ d_{AD} \\ d_{AE} \end{pmatrix} = (d_{BD}) = \mathbf{d}_{\mathrm{f}}$$

Using F in conjunction with $\mathbf{I}_{j-1}$, $\mathbf{d}$ can then be expressed as a linear combination of the basic parameters:

$$\mathbf{d} = \begin{pmatrix} \mathbf{d}_{\mathrm{b}} \\ \mathbf{d}_{\mathrm{f}} \end{pmatrix} = \begin{pmatrix} \mathbf{I}_{j-1} \\ \mathbf{F} \end{pmatrix} \mathbf{d}_{\mathrm{b}} = \mathbf{H}\mathbf{d}_{\mathrm{b}}$$

The purpose of generating $\mathbf{H}$ is to create the linear constraint that will be used in conjunction with $\mathbf{y}^{\mathrm{gls}}$ and $\mathbf{V}^{\mathrm{gls}}$ to provide an estimate, $\hat{\mathbf{d}}^{\mathrm{gls}}$, which:

1. Provides the smallest sum of squared errors with respect to the weights by minimizing $\sum_{k=1}^{G}(\hat{\mathbf{d}}_{g_k} - \mathbf{y}_{g_k}^{\mathrm{gls}})'\boldsymbol{W}_{g_k}^{\mathrm{gls}}(\hat{\mathbf{d}}_{g_k} - \mathbf{y}_{g_k}^{\mathrm{gls}})$ when $\hat{\mathbf{d}}_{g_k} = \hat{\mathbf{d}}_{g_k}^{\mathrm{gls}(*)}$, where $\mathbf{y}_{g_k}^{\mathrm{gls}}$ and $\mathbf{W}_{g_k}^{\mathrm{gls}}$ are the portions of $\mathbf{y}^{\mathrm{gls}}$ and $\mathbf{W}^{\mathrm{gls}}$ corresponding to design g_k, respectively, and $\hat{\mathbf{d}}_{g_k}^{\mathrm{gls}(*)}$ is the portion of $\hat{\mathbf{d}}^{\mathrm{gls}}$ containing the estimates of the effect size parameters provided for design g_k.
2. Satisfies the linear constraint $\mathbf{d} = \mathbf{H}\mathbf{d}_{\mathrm{b}}$.

We now construct a linear model that will lead to an estimator which will satisfy the above conditions. This linear model is given by:

$$\mathbf{y}^{\text{gls}} = \mathbf{X}\mathbf{d}_{\text{b}} + \boldsymbol{\epsilon}$$

where $\boldsymbol{\epsilon} \sim \text{N}_T(\mathbf{0}, \mathbf{V}^{\text{gls}})$ and $\mathbf{X}$ is a $T \times (J-1)$ design matrix that interacts with $\mathbf{d}_{\text{b}}$ in such a way that the effect size parameters in the resulting $T \times 1$ vector correspond with the effect size parameter estimates given in $\mathbf{y}^{\text{gls}}$. Since $\mathbf{H}$ contains all the linear combinations for converting $\mathbf{d}_{\text{b}}$ into any element of $\mathbf{d}$, $\mathbf{X}$ will be a matrix formed by vertically concatenating rows of $\mathbf{H}$. Let $\mathbf{h}_{\text{i}}$ correspond to the i^{th} row of $\mathbf{H}$. For our example:

$$\mathbf{y}^{\text{gls}} = \begin{pmatrix} \hat{d}_{AB} \\ \hat{d}_{AB\cdot} \\ \hat{d}_{AC\cdot} \\ \hat{d}_{BD} \\ \hat{d}_{AC\cdot\cdot} \\ \hat{d}_{AE\cdot\cdot} \\ \hat{d}_{AB\cdots} \\ \hat{d}_{AD\cdots} \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} d_{AB} \\ d_{AC} \\ d_{AD} \\ d_{AE} \end{pmatrix} = \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_5 \\ \mathbf{h}_2 \\ \mathbf{h}_4 \\ \mathbf{h}_1 \\ \mathbf{h}_3 \end{bmatrix} \begin{pmatrix} d_{AB} \\ d_{AC} \\ d_{AD} \\ d_{AE} \end{pmatrix} = \mathbf{X}\mathbf{d}_{\text{b}}$$

Now that $\mathbf{X}$ has been constructed, the estimates for the effect size parameters in $\mathbf{d}_{\text{b}}$ can be estimated by $\hat{\mathbf{d}}_{\text{b}}^{\text{gls}}$ which is given by:

$$\hat{\mathbf{d}}_{\text{b}}^{\text{gls}} = (\mathbf{X}'\mathbf{W}^{\text{gls}}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}^{\text{gls}}\mathbf{y}^{\text{gls}} = \begin{pmatrix} \hat{d}_{AB}^{\text{gls}} \\ \hat{d}_{AC}^{\text{gls}} \\ \hat{d}_{AD}^{\text{gls}} \\ \hat{d}_{AE}^{\text{gls}} \end{pmatrix}$$

Further, since this is the generalized least squares solution, we know $\hat{\mathbf{d}}_{\text{b}}^{\text{gls}}$ is the best linear unbiased estimator for $\mathbf{d}_{\text{b}}$. Finally, since this estimate also satisfies the linear

constraint $\mathbf{d} = \mathbf{H}\mathbf{d}_{\mathrm{b}}$, we receive the estimates for the effect size parameters in $\mathbf{d}$ by computing:

$$\hat{\mathbf{d}}^{\mathrm{gls}} = \mathbf{H}\hat{\mathbf{d}}_{\mathrm{b}}^{\mathrm{gls}} = \begin{pmatrix} \hat{d}_{AB}^{\mathrm{gls}} \\ \hat{d}_{AC}^{\mathrm{gls}} \\ \hat{d}_{AD}^{\mathrm{gls}} \\ \hat{d}_{AE}^{\mathrm{gls}} \\ \hat{d}_{BD}^{\mathrm{gls}} \end{pmatrix}$$

It is an easy exercise to see that $\mathbb{E}\left(\hat{\mathbf{d}}^{\mathrm{gls}}\right) = \mathbf{d}$ and $\mathrm{Cov}\left(\hat{\mathbf{d}}^{\mathrm{gls}}\right) = \mathbf{H}\left(\mathbf{X}'\mathbf{W}^{\mathrm{gls}}\mathbf{X}\right)^{-1}\mathbf{H}'$. The effect size estimates for all $\binom{J}{2}$ possible comparisons can then be computed by using linear combinations of the effect size estimates given in $\hat{\mathbf{d}}^{\mathrm{gls}}$. The variances for these estimates can also be computed using $\mathrm{Cov}\left(\hat{\mathbf{d}}^{\mathrm{gls}}\right)$. For example, the effect size estimate and variance for comparing compound C to compound D is given by:

$$\hat{d}_{CD}^{\mathrm{gls}} = (0\ -1\ 1\ 0\ 0)\,\hat{\mathbf{d}}^{\mathrm{gls}} = \hat{d}_{AC}^{\mathrm{gls}} - \hat{d}_{AD}^{\mathrm{gls}}$$

$$Var(\hat{\mathrm{d}}_{CD}^{\mathrm{gls}}) = \mathrm{Var}((0\ -1\ 1\ 0\ 0)\,\hat{\mathbf{d}}^{\mathrm{gls}})$$

$$Var(\hat{\mathrm{d}}_{CD}^{\mathrm{gls}}) = (0\ -1\ 1\ 0\ 0)'\mathrm{Cov}(\hat{\mathbf{d}}^{\mathrm{gls}})(0\ -1\ 1\ 0\ 0)$$

$$Var(\hat{\mathrm{d}}_{CD}^{\mathrm{gls}}) = \mathrm{Cov}(\hat{\mathbf{d}}^{\mathrm{gls}})_{22} + \mathrm{Cov}(\hat{\mathbf{d}}^{\mathrm{gls}})_{33} - 2\mathrm{Cov}(\hat{\mathbf{d}}^{\mathrm{gls}})_{23}$$

$$\mathrm{Var}(\hat{\mathrm{d}}_{CD}^{\mathrm{gls}}) = \mathrm{Var}(\hat{\mathrm{d}}_{AC}^{\mathrm{gls}}) + \mathrm{Var}(\hat{\mathrm{d}}_{AD}^{\mathrm{gls}}) - 2\mathrm{Cov}(\hat{\mathrm{d}}_{AC}^{\mathrm{gls}}, \hat{\mathrm{d}}_{AD}^{\mathrm{gls}})$$

2.2.2 Electrical Network Theory

We will now describe the electrical network theory (ENT) method of analysis. This approach uses electrical networks as the theoretical foundation for generating estimates as opposed to the linear regression techniques typically employed in statistics.

This method is motivated by the notion presented in (Bailey, 2007) that variances in microarray experiments combine similar to resistances in electrical networks.

We will once again use our example to demonstrate how estimates are created; however, the example dataset must be converted from an individual summary measures dataset to a contrast summary measures dataset. In other words, the ENT method generates estimates using data where treatment arm comparisons are already calculated as opposed to the GLS method which requires data for each individual treatment arm.

Experiment	Comparison	Sample Size	Sample Mean Difference	Pooled Standard Error
1	$A-B$	$n_{11}+n_{12}$	$\delta_{11}=\bar{x}_{11}-\bar{x}_{12}$	$s_{\delta_{11}}$
2	$A-B$	$n_{21}+n_{22}$	$\delta_{21}=\bar{x}_{21}-\bar{x}_{22}$	$s_{\delta_{21}}$
3	$A-B$	$n_{31}+n_{32}$	$\delta_{31}=\bar{x}_{31}-\bar{x}_{32}$	$s_{\delta_{31}}$
3	$A-C$	$n_{31}+n_{33}$	$\delta_{32}=\bar{x}_{31}-\bar{x}_{33}$	$s_{\delta_{32}}$
3	$B-C$	$n_{32}+n_{33}$	$\delta_{33}=\bar{x}_{32}-\bar{x}_{33}$	$s_{\delta_{33}}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
8	$A-B$	$n_{81}+n_{82}$	$\delta_{81}=\bar{x}_{81}-\bar{x}_{82}$	$s_{\delta_{81}}$
8	$A-C$	$n_{81}+n_{83}$	$\delta_{82}=\bar{x}_{81}-\bar{x}_{83}$	$s_{\delta_{82}}$
8	$B-C$	$n_{82}+n_{83}$	$\delta_{83}=\bar{x}_{82}-\bar{x}_{83}$	$s_{\delta_{83}}$

Table 3: Contrast Summary Measures

In the case where summary measures are given as contrasts instead of individual summary measures for each treatment arm, let the triplet $\left(\delta_{ij}, s_{\delta_{ij}}, n_{ij}\right)$ for $i = 1,2,\ldots,N, j = 1,2,\ldots,\binom{p_i}{2}$ be the sample mean difference, pooled variance, and sample size for the j^{th} comparison in experiment i. Table 3 displays the data layout for contrast summary measures. Let $p=\sum_{i=1}^{N} p_i$ and $\mathbf{y}^{\text{ent}}$ be the $p \times 1$ vector containing all the sample mean differences.

The first step is to create **B**, the edge-vertex incidence matrix, a $p \times J$ matrix where the p rows represent the treatment arm comparisons being made in each experiment and the J columns represent the J treatment arms. Further, let $\mathbf{B}_g$ be the $\binom{T_g}{2} \times J$ matrix representing the portions of the edge-vertex incidence matrix corresponding to an experiment with design g for $g \in \{g_1, g_2, \ldots, g_G\}$. For our example, compounds A, B, C, D, and E correspond to columns 1,2,3,4, and 5, respectively, and the following matrices are formed:

$$\mathbf{B}_{AB} = [1 \quad -1 \quad 0 \quad 0 \quad 0], \mathbf{B}_{ABC} = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \end{bmatrix}$$

$$\mathbf{B}_{BD} = [0 \quad 1 \quad 0 \quad -1 \quad 0], \mathbf{B}_{ACE} = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & -1 \end{bmatrix}$$

$$\mathbf{B}_{ABC} = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 & 0 \end{bmatrix}$$

So that:

$$\mathbf{B} = \begin{bmatrix} \mathbf{B}_{AB} \\ \mathbf{B}_{AB} \\ \mathbf{B}_{ABC} \\ \mathbf{B}_{BD} \\ \mathbf{B}_{ACE} \\ \mathbf{B}_{ABD} \\ \mathbf{B}_{ABD} \\ \mathbf{B}_{ABC} \end{bmatrix}$$

This edge-vertex incidence matrix is then used in conjunction with θ^{treat}, a $J \times 1$ vector containing the J treatment arms in the same order as they appear as the columns in

B. For our example, $\theta^{\text{treat}} = (\theta_A, \theta_B, \theta_C, \theta_D, \theta_E)'$. The effect size parameters are then estimated according to the linear model below.

$$\mathbf{y}^{\text{ent}} = \mathbf{B}\theta^{\text{treat}} + \boldsymbol{\epsilon}$$

$$\begin{pmatrix} \delta_{11} \\ \delta_{21} \\ \delta_{31} \\ \delta_{32} \\ \delta_{33} \\ \vdots \\ \delta_{81} \\ \delta_{82} \\ \delta_{83} \end{pmatrix} = \begin{bmatrix} \mathbf{B}_{AB} \\ \mathbf{B}_{AB} \\ \mathbf{B}_{ABC} \\ \vdots \\ \mathbf{B}_{ABC} \end{bmatrix} \begin{pmatrix} \theta_A \\ \theta_B \\ \theta_C \\ \theta_D \\ \theta_E \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \\ \epsilon_5 \\ \vdots \\ \epsilon_{16} \\ \epsilon_{17} \\ \epsilon_{18} \end{pmatrix} = \begin{pmatrix} d_{AB} \\ d_{AB} \\ d_{AB} \\ d_{AC} \\ d_{BC} \\ \vdots \\ d_{AB} \\ d_{AC} \\ d_{BC} \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \\ \epsilon_5 \\ \vdots \\ \epsilon_{16} \\ \epsilon_{17} \\ \epsilon_{18} \end{pmatrix}$$

where $\boldsymbol{\epsilon} \sim \text{N}_p(\mathbf{0}, \mathbf{V}^{\text{ent}})$ and $\mathbf{V}^{\text{ent}}$ is a diagonal covariance matrix. More specifically, $\mathbf{V}^{\text{ent}} = (\mathbf{W}^{\text{ent}})^{-1}$, where $\mathbf{W}^{\text{ent}} = \text{diag}(\mathbf{W}_1, \dots, \mathbf{W}_{\text{N}})$ is the weight matrix used for this method. $\mathbf{W}_i$ is a $\binom{p_i}{2} \times \binom{p_i}{2}$ diagonal matrix which contains the weight information pertaining to experiment i. When $p_i = 2$, $\mathbf{W}_{\text{i}}$ is simply a scalar representing the inverse of the pooled variance for the sample mean difference of the two treatment arms being compared:

$$\mathbf{W}_{\text{i}} = \frac{1}{s^2_{\delta_{i1}}}$$

When $p_i > 2, \mathbf{W}_i$ has a more complicated derivation. Let $\underline{\mathbf{B}}_g$ be the $\binom{T_g}{2} \times \binom{T_g}{2}$ sub edge-vertex incidence matrix pertaining to design g for designs where $T_g > 2$. $\underline{\mathbf{B}}_g$ is simply $\mathbf{B}_g$ where columns that do not have nonzero entries are removed. For our example:

$$\underline{\mathbf{B}}_{ABC} = \begin{bmatrix} 1 & -1 & 0 \\ 1 & 0 & -1 \\ 0 & 1 & -1 \end{bmatrix}, \underline{\mathbf{B}}_{ACE} = \begin{bmatrix} 1 & -1 & 0 \\ 1 & 0 & -1 \\ 0 & 1 & -1 \end{bmatrix}, \underline{\mathbf{B}}_{ABD} = \begin{bmatrix} 1 & -1 & 0 \\ 1 & 0 & -1 \\ 0 & 1 & -1 \end{bmatrix}$$

Note that although the matrices appear to be the same, the columns in the matrices represent different treatment arms. The columns in $\underline{\mathbf{B}}_{ABC}$ represent compounds A, B, and C; the columns in $\underline{\mathbf{B}}_{ACE}$ represent compounds A, C, and E; and the columns in $\underline{\mathbf{B}}_{ABD}$ represent compounds A, B, and D.

The sub edge-vertex incidence matrices are then used in conjunction with $\underline{\mathbf{V}}_i$, the $\binom{p_i}{2} \times \binom{p_i}{2}$ sub variance matrix pertaining to experiment i for experiments where $p_i > 2$. $\underline{\mathbf{V}}_i$ is a symmetric matric with 0 along the diagonal and entries (q, r) contain the pooled variance for the sample mean difference comparing the treatment arms in the q^{th} and r^{th} columns of the sub edge-vertex incidence matrix. For our example:

$$\underline{\mathbf{V}}_3 = \begin{bmatrix} 0 & s^2_{\delta_{31}} & s^2_{\delta_{32}} \\ s^2_{\delta_{31}} & 0 & s^2_{\delta_{33}} \\ s^2_{\delta_{32}} & s^2_{\delta_{33}} & 0 \end{bmatrix}, \underline{\mathbf{V}}_5 = \begin{bmatrix} 0 & s^2_{\delta_{51}} & s^2_{\delta_{52}} \\ s^2_{\delta_{51}} & 0 & s^2_{\delta_{53}} \\ s^2_{\delta_{52}} & s^2_{\delta_{53}} & 0 \end{bmatrix}, \ldots, \underline{\mathbf{V}}_8 = \begin{bmatrix} 0 & s^2_{\delta_{81}} & s^2_{\delta_{82}} \\ s^2_{\delta_{81}} & 0 & s^2_{\delta_{83}} \\ s^2_{\delta_{82}} & s^2_{\delta_{83}} & 0 \end{bmatrix}$$

The next step is to form the $\mathbf{L}_i^+$ matrix using the $\underline{\mathbf{B}}_g$ and $\underline{\mathbf{V}}_i$ matrices for each experiment i with design g where $T_g > 2$.

$$\mathbf{L}_i^+ = \frac{1}{2p_i^2} \underline{\mathbf{B}}'_g \underline{\mathbf{B}}_g \underline{\mathbf{V}}_i \underline{\mathbf{B}}'_g \underline{\mathbf{B}}_g$$

To obtain the entries that will form the diagonal of $\mathbf{W}_i$, $\mathbf{L}_i^+$must be converted to $\mathbf{L}_i$ using the Moore-Penrose pseudo-inverse (Albert, 1972). Let $\mathbf{J}_k$ be a $k \times k$ matrix where every entry is 1. For any $k \times k$ matrix $\mathbf{A}$, let the Moore-Penrose pseudo-inverse of $\mathbf{A}$, $\mathbf{A}^+$, be defined as:

$$\mathbf{A}^{+} = \left(\mathbf{A} - \frac{1}{k}\mathbf{J}_k\right)^{-1} - \frac{1}{k}\mathbf{J}_k$$

Let $l_{i(q,r)}$ be the (q, r) entry of the matrix $\mathbf{L}_i$. The inverse of the negative non-diagonal elements of $\mathbf{L}_i$ are meant to serve as "adjusted" variances to replace the pooled variances found in the corresponding slot of $\underline{\mathbf{V}_i}$ by inflating them. This is done to adjust for any within-experiment correlation between treatment arms that may be present. $\mathbf{W}_i$ when $p_i > 2$ is then given by:

$$\mathbf{W}_i = diag(-l_{i(q,r)}) \; for \; 1 \leq q < r \leq p_i$$

For our examples, $\mathbf{W}_i = diag\left(\left(-l_{i(1,2)}\right), \left(-l_{i(1,3)}\right), \left(-l_{i(2,3)}\right)\right)$ for $i = 3, 5, 6, 7, 8$ since $p_i = 3$ for these experiments. Once $\mathbf{W}_i$ has been formed for $i = 1, \ldots, N$, $\mathbf{W}^{\text{ent}}$ is formed and used with $\mathbf{B}$ to create $\mathbf{L} = \mathbf{B}'\mathbf{W}^{\text{ent}}\mathbf{B}$. Next, we compute the Moore-Penrose pseudo-inverse of $\mathbf{L}$, $\mathbf{L}^{+}$. Finally, the estimates for the effect size parameters in $\mathbf{y}^{\text{ent}}$ are given by:

$$\hat{\mathbf{y}}^{\text{ent}} = \mathbf{B}\mathbf{L}^{+}\mathbf{B}'\mathbf{W}^{\text{ent}}\mathbf{y}^{\text{ent}}$$

$$\begin{pmatrix} \hat{d}_{AB}^{\text{ent}} \\ \hat{d}_{AB}^{\text{ent}} \\ \hat{d}_{AB}^{\text{ent}} \\ \hat{d}_{AC}^{\text{ent}} \\ \hat{d}_{BC}^{\text{ent}} \\ \vdots \\ \hat{d}_{AB}^{\text{ent}} \\ \hat{d}_{AC}^{\text{ent}} \\ \hat{d}_{BC}^{\text{ent}} \end{pmatrix} = \mathbf{B}\mathbf{L}^{+}\mathbf{B}'\mathbf{W}^{\text{ent}} \begin{pmatrix} \delta_{11} \\ \delta_{21} \\ \delta_{31} \\ \delta_{32} \\ \delta_{33} \\ \vdots \\ \delta_{81} \\ \delta_{82} \\ \delta_{83} \end{pmatrix}$$

$\hat{\mathbf{y}}^{\text{ent}}$ contains the effect size parameter estimates for all the direct comparisons available. Like the GLS method, the effect size parameter estimates for indirect

comparisons are computed using linear combinations of the estimates provided in $\hat{\mathbf{y}}^{\text{ent}}$; however, the variance of any effect size parameter estimate is computed differently. Regardless of whether the effect size parameter estimate corresponds to a direct or indirect comparison, the variance is computed as follows: the variance of the effect size parameter estimate which compares the treatment arm associated with column i in B to the treatment arm associated with column j in B is given by:

$$\mathbf{L}^{+}_{(ii)} + \mathbf{L}^{+}_{(jj)} - 2\mathbf{L}^{+}_{(ij)}$$

For our example the effect size estimate and variance for comparing compound C to compound D is given by:

$$Var\left(\hat{\mathbf{d}}^{\text{ent}}_{CD}\right) = \mathbf{L}^{+}_{(33)} + \mathbf{L}^{+}_{(44)} - 2\mathbf{L}^{+}_{(34)}$$

Note that unlike the effect size parameters estimated by $\hat{\mathbf{d}}^{\text{gls}}$, $\hat{\mathbf{y}}^{\text{ent}}$ may contain multiple (identical) estimates for the same parameter. Further, since a baseline choice does not have to be made for all designs with more than two treatment arms to make sure the design matrix is full rank, more effect size parameters can be immediately deduced. For example, the ninth element of $\hat{\mathbf{y}}^{\text{ent}}$ contains an estimate for the effect size parameter d_{CE}, while $\hat{\mathbf{d}}^{\text{gls}}$ will not contain an estimate for d_{CE} unless a different baseline choice configuration is chosen.

Further, whether individual or contrast summary measures data is available will influence which method should be used. In the case where individual summary measures are available either method can be used since the conversion from individual to contrast summary measures is a simple feat. When only contrast summary measures are available one must use methods to impute the off-diagonal variances that will be placed in the

variance matrices to use the GLS method (Riley, 2009; Franchini, Dias, Ades, Jansen, & Welton, 2012).

It should be noted that there is also plenty of literature detailing how to conduct network meta-analysis with under a Bayesian framework (Lu & Ades, 2004; Dias, Sutton, Ades, & Welton, 2013; Hong, et al., 2013). While these Bayesian methods are useful, we will remain in the realm of the frequentist methods.

2.3 A Comparison Between the GLS and ENT Methods

Rücker and Schwarzer assert that the effect size parameter estimates and their variances will be identical for both methods (Rücker & Schwarzer, 2014). This cannot be correct because we have found a counterexample in the form of a dataset heavily based on preclinical data for which the two methods are in fact not equivalent.

We will now apply the GLS method and the ENT method to two separate datasets and provide the results for each. Each section will provide the data being used, the computations of the basic matrices needed in each method, and the effect size parameter estimates yielded from each method, all of which can be checked by the reader independently to ensure total transparency.

For this section we will use Dataset A provided in Table 4. This dataset contains 3 different experiments $(N = 3)$, two different designs $(G = 2)$, and four different treatment arms $(J = 4)$. This experiment does not have a balanced design. While this dataset may not seem ideal, it is heavily based on data from a pharmaceutical company. See Figure 11 for the experiment design for this dataset.

2.3.1 A Counterexample Where Methods Are Not Equivalent

Experiment	Treatment Arm	Sample Size	Sample Mean	Standard Error
1	A	5	100.00	15.83
1	B	10	76.63	5.75
2	A	5	100.00	2.05
2	B	10	100.28	3.56
3	A	5	100.00	11.19
3	B	8	76.84	3.26
3	C	8	65.41	3.07
3	D	8	81.72	5.25

Table 4: Dataset A (Counterexample)

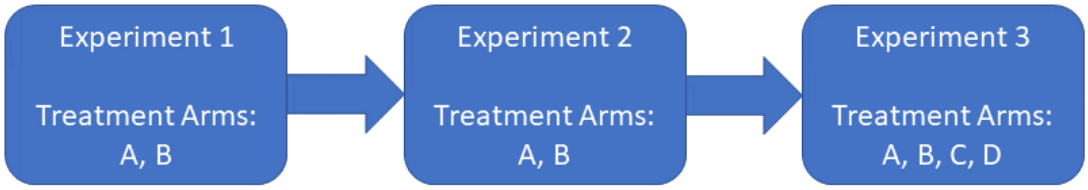

Figure 11: Dataset A Experiment Designs

We will begin by finding results with the GLS method using the individual summary measures version of this dataset. The two designs are $\{AB, ABCD\}$. For design $ABCD$ we will choose our baseline choice to be Compound C. So $\mathbf{y}^{\text{gls}}$ and $\mathbf{V}^{\text{gls}}$ will be:

$$\mathbf{y}^{\text{gls}} = \begin{pmatrix} \hat{d}_{AB} \\ \hat{\mathbf{d}}_{ABC} \end{pmatrix} = \begin{pmatrix} \hat{d}_{AB} \\ \hat{d}_{AC\cdot} \\ \hat{d}_{BC\cdot} \\ \hat{d}_{CD\cdot} \end{pmatrix} = \begin{pmatrix} 2.864 \\ 34.590 \\ 11.430 \\ -16.310 \end{pmatrix}$$

$$\mathbf{V}^{\text{gls}} = (\mathbf{W}^{\text{gls}})^{-1} = \text{diag}\left(s^2_{\hat{d}_{AB}}, \mathbf{V}^{ABCD}\right) = \begin{bmatrix} 24.505 & 0 & 0 & 0 \\ 0 & 89.585 & 9.425 & -9.425 \\ 0 & 9.425 & 20.053 & -9.425 \\ 0 & -9.425 & -9.425 & 36.987 \end{bmatrix}$$

Note that the covariances for certain entries are negative. This is because the pair of effect size parameters for which the covariance is being estimated have the baseline treatment arm in a different comparitive positions. For example:

$$Cov(d_{BC}, d_{CD}) = Cov(\theta_B - \theta_C, \theta_C - \theta_D)$$

$$\mathrm{Cov}(d_{BC}, d_{CD}) = \mathrm{Cov}(\theta_B, \theta_C - \theta_D) + \mathrm{Cov}(-\theta_C, \theta_C - \theta_D)$$

$$\mathrm{Cov}(d_{BC}, d_{CD}) = \mathrm{Cov}(\theta_B, \theta_C) + \mathrm{Cov}(\theta_B, -\theta_D) + \mathrm{Cov}(-\theta_C, \theta_C) + \mathrm{Cov}(-\theta_C, -\theta_D)$$

$$\mathrm{Cov}(d_{BC}, d_{CD}) = -\mathrm{Var}(\theta_C)^1$$

Our vector containing the effect size parameters is then $\mathbf{d} = (d_{AB}, d_{AC}, d_{BC}, d_{CD})'$. Further, we will choose $\mathbf{d}_\mathrm{b} = (d_{AB}, d_{AC}, d_{CD})'$ to be our basic parameter vector and $\mathbf{d}_\mathrm{f} = (d_{BC})'$ to be our functional parameter vector. As such:

$$\mathbf{d} = \begin{pmatrix} \mathbf{d}_\mathrm{b} \\ \mathbf{d}_\mathrm{f} \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{d}_\mathrm{b} = \mathbf{H}\mathbf{d}_\mathrm{b}$$

so that

$$\mathbf{y}^{\mathrm{gls}} = \begin{pmatrix} \hat{d}_{AB} \\ \hat{d}_{AC\cdot} \\ \hat{d}_{BC\cdot} \\ \hat{d}_{CD\cdot} \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} d_{AB} \\ d_{AC} \\ d_{CD} \end{pmatrix} = \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \\ \mathbf{h}_4 \end{bmatrix} \begin{pmatrix} d_{AB} \\ d_{AC} \\ d_{CD} \end{pmatrix} = \mathbf{X}\mathbf{d}_\mathrm{b}$$

and finally

[1] To reach this result we assume that $\{\theta_A, \theta_B, \theta_C, \theta_D\}$ are mutually independent as mentioned earlier in section 2.2.1.

$$\hat{\mathbf{d}}^{\text{gls}} = \begin{pmatrix} \hat{d}_{AB}^{\text{gls}} \\ \hat{d}_{AC}^{\text{gls}} \\ \hat{d}_{BC}^{\text{gls}} \\ \hat{d}_{CD}^{\text{gls}} \end{pmatrix} = \mathbf{H}(\mathbf{X}'\mathbf{W}^{\text{gls}}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}^{\text{gls}}\mathbf{y}^{\text{gls}} = \begin{pmatrix} 7.178 \\ 20.479 \\ 13.301 \\ -16.310 \end{pmatrix}$$

The effect size parameter estimates for the remaining direct comparisons are calculated by taking linear combinations of $\hat{\mathbf{d}}^{\text{gls}}$. Specifically, they are given by $\hat{d}_{AC}^{\text{gls}} = 4.169$ and $\hat{d}_{BD}^{\text{gls}} = -3.009$.

Next we will use the contrast summary measures version of the dataset and use the ENT method to generate results. First recall that $\hat{\mathbf{y}}^{\text{ent}}$ is the 8×1 vector containing all of the sample mean differences. Next we will form the edge-vertex incidence matrix, B. Compounds A, B, C, and D will correspond to columns 1, 2, 3, and 4 of B, respectively, and the following matrices are formed. So:

$$\mathbf{B}_{AB} = [1 \quad -1 \quad 0 \quad 0], \mathbf{B}_{ABCD} = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

so that

$$\mathbf{B} = \begin{bmatrix} \mathbf{B}_{AB} \\ \mathbf{B}_{AB} \\ \mathbf{B}_{ABCD} \end{bmatrix}$$

and

$$\mathbf{y}^{\text{ent}} = \mathbf{B}\theta^{\text{treat}} + \boldsymbol{\epsilon}$$

$$\begin{pmatrix}\delta_{11}\\ \delta_{21}\\ \delta_{31}\\ \delta_{32}\\ \delta_{33}\\ \delta_{34}\\ \delta_{35}\\ \delta_{36}\end{pmatrix} = \begin{bmatrix}\mathbf{B}_{AB}\\ \mathbf{B}_{AB}\\ \mathbf{B}_{ABCD}\end{bmatrix}\begin{pmatrix}\theta_A\\ \theta_B\\ \theta_C\\ \theta_D\end{pmatrix} + \begin{pmatrix}\epsilon_1\\ \epsilon_2\\ \epsilon_3\\ \epsilon_4\\ \epsilon_5\\ \epsilon_6\\ \epsilon_7\\ \epsilon_8\end{pmatrix} = \begin{pmatrix}d_{AB}\\ d_{AB}\\ d_{AB}\\ d_{AC}\\ d_{AD}\\ d_{BC}\\ d_{BD}\\ d_{CD}\end{pmatrix} + \begin{pmatrix}\epsilon_1\\ \epsilon_2\\ \epsilon_3\\ \epsilon_4\\ \epsilon_5\\ \epsilon_6\\ \epsilon_7\\ \epsilon_8\end{pmatrix}$$

Since design $ABCD$ has four treatment arms we must form a sub edge-vertex incidence matrix $\underline{\mathbf{B}}_{ABCD}$.

$$\underline{\mathbf{B}}_{ABCD} = \begin{bmatrix}1 & -1 & 0 & 0\\ 1 & 0 & -1 & 0\\ 1 & 0 & 0 & -1\\ 0 & 1 & -1 & 0\\ 0 & 1 & 0 & -1\\ 0 & 0 & 1 & -1\end{bmatrix}$$

Next we form a sub variance matrix $\underline{\mathbf{V}}_3$ for experiment 3 since this experiment has design $ABCD$.

$$\underline{\mathbf{V}}_3 = \begin{bmatrix}0 & 91.575 & 89.585 & 119.595\\ 91.575 & 0 & 20.053 & 38.190\\ 89.585 & 20.053 & 0 & 36.987\\ 119.595 & 38.190 & 36.987 & 0\end{bmatrix}$$

Next, we compute $\mathbf{L}_3^+$ and form $\mathbf{W}_3$ for experiment 3. We will leave this process for the reader to complete. Finally, with these computations completed we can now form $\mathbf{W}^{\text{ent}}$, which is given by:

$$\mathbf{W}^{\text{ent}} = diag(0.005, 0.035, 0.005, 0.007, -\mathbf{0.0002}, 0.039, 0.15, 0.016)$$

Finally, after computing $\mathbf{L} = \mathbf{B}'\mathbf{W}^{\text{ent}}\mathbf{B}$, we arrive at the result:

$$\begin{pmatrix} \hat{d}_{AB}^{\text{ent}} \\ \hat{d}_{AB}^{\text{ent}} \\ \hat{d}_{AB}^{\text{ent}} \\ \hat{d}_{AC}^{\text{ent}} \\ \hat{d}_{AD}^{\text{ent}} \\ \hat{d}_{BC}^{\text{ent}} \\ \hat{d}_{BD}^{\text{ent}} \\ \hat{d}_{CD}^{\text{ent}} \end{pmatrix} = \mathbf{B}\mathbf{L}^{+}\mathbf{B}'\mathbf{W}^{\text{ent}} \begin{pmatrix} \delta_{11} \\ \delta_{21} \\ \delta_{31} \\ \delta_{32} \\ \delta_{33} \\ \delta_{34} \\ \delta_{35} \\ \delta_{36} \end{pmatrix} = \begin{pmatrix} 7.149 \\ 7.149 \\ 7.149 \\ 20.506 \\ 3.158 \\ 13.357 \\ -3.991 \\ -17.348 \end{pmatrix}$$

Here it is seen that $\hat{d}_{c_1c_2}^{\text{gls}} \neq \hat{d}_{c_1c_2}^{\text{ent}}$ for every possible choice of c_1 and c_2. In other words, the effect size parameter estimates for each method are not equivalent. We leave it to the reader to confirm that the variances for all of the effect size parameter estimates are not equivalent.

There are a few things that should be noted here. First, the SSE for the GLS method results is less than the SSE for the ENT method results. This is not surprising since the estimator provided by the GLS method is the best unbiased linear estimator in terms of SSE. This is why it is so important that the ENT method provides equivalent results, which in this example it does not.

Second, different results for the GLS method are possible depending on the baseline choice we use for calculating the initial fixed effects estimates for design $ABCD$. This is because when the baseline choice is compound A or compound B, the effect size parameter vector for design $ABCD$ includes d_{AB}, which is then used in conjunction with the effect size parameter for design AB to create $\hat{d}_{AB}^{\text{gls}}$. On the other hand, when the baseline choice is compound C or compound D, the effect size parameter vector for design $ABCD$ does not include d_{AB}, so $\hat{d}_{AB}^{\text{gls}}$ is formed with the information from experiments with design AB only.

Lu et. al insist that the final results for the GLS method are invariant to the baseline choice for designs with more than two treatment arms (Lu, Welton, Higgins, White, & Ades, 2011). We invite readers to apply the GLS method to Dataset A for every other baseline choice to see that this is not true. We calculated the GLS estimates for every choice, and used Compound C as the baseline choice because this estimate had the smallest SSE. Also, none of the estimates generated with different baseline choices for the GLS method yielded equivalent estimates to the ENT method.

Third, note that one of the weights in $\mathbf{W}^{\mathrm{ent}}$ has been bolded. The weights calculated for the ENT method for experiments with more than two treatment arms should all be positive according to (Schwarzer, Carpenter, & Rücker, 2015). This is because they are always supposed to be retrieved as the negative non-diagonal entries from $\mathbf{L}_{\mathrm{i}}^{+}$ for experiments with more than two treatment arms. In our example, one of the non-diagonal entries from $\mathbf{L}_{3}^{+}$ is not negative and leads to the weight -0.0002. This means that this dataset is not compatible with the theoretical framework that the ENT method uses. As such, the ENT method should not be used for this dataset.

All of these items together show that for this particular dataset, not only are the GLS and ENT methods not equivalent, but the GLS method is better both in practical terms since it minimizes the SSE by construction, and theoretically since the ENT method produces negative weights. As such, the original GLS method should be used whenever possible; however, in Section 2.3.3 we offer potential conditions for when the two methods may be used interchangably.

Of course, since the the baseline choices for experiments with more than two treatment arms determine what the final results for the GLS method will be, this method

should be further optimized as well by choosing the baseline choice which minimizes the SSE of the final effect size parameter estimates. In the next section we present a dataset for which the methods are equivalent.

2.3.2 Dataset for Which Methods Are Equivalent

For this section we will use Dataset B given in Table 5. This dataset contains six different experiments $(N = 6)$, three different designs $(G = 3)$, and three different treatment arms $(J = 3)$. Each experiment has a balanced design and they all have the same sample size $\left(n_{ij} = 10,\ for\ i = 1, \dots, 6,\ j = 1, \dots, p_i\right)$. This dataset was simulated by the author. See Figure 12 for the experiment design for this dataset.

Experiment	Treatment Arm	Sample Size	Sample Mean	Standard Error
1	A	10	-0.328	0.253
1	B	10	5.836	0.278
1	C	10	12.388	0.280
2	A	10	-3.023	0.203
2	B	10	1.461	0.210
2	C	10	10.778	0.305
3	B	10	4.455	0.329
3	C	10	9.153	0.376
4	A	10	-1.006	0.318
4	B	10	4.305	0.297
5	B	10	2.680	0.280
5	C	10	8.447	0.297
6	A	10	-1.157	0.239
6	B	10	2.939	0.255
6	C	10	12.032	0.370

Table 5: Dataset B

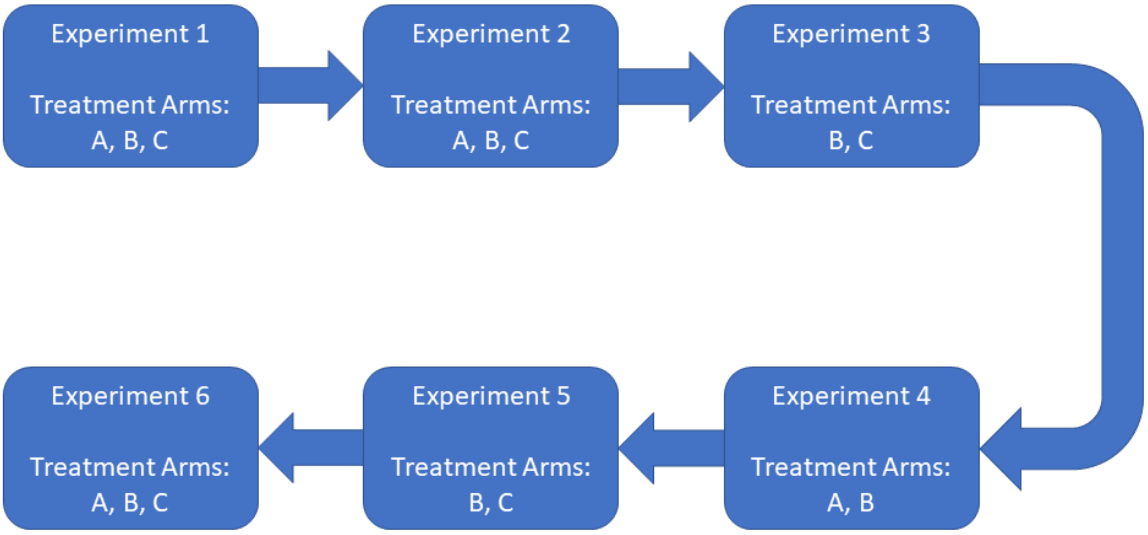

Figure 12: Dataset B Experiment Designs

We will begin by finding results with the GLS method using the individual summary measures version of this dataset. The three designs are $\{ABC, BC, AB\}$. For design ABC we will choose our baseline choice to be Compound A. So $\mathbf{y}^{\text{gls}}$ and $\mathbf{V}^{\text{gls}}$ will be:

$$\mathbf{y}^{\text{gls}} = \begin{pmatrix} \hat{\mathbf{d}}_{ABC} \\ \hat{d}_{BC} \\ \hat{d}_{AB} \end{pmatrix} = \begin{pmatrix} \hat{d}_{AB\cdot} \\ \hat{d}_{AC\cdot} \\ \hat{d}_{BC} \\ \hat{d}_{AB} \end{pmatrix} = \begin{pmatrix} -4.798 \\ -13.148 \\ -5.339 \\ -5.311 \end{pmatrix}$$

$$\mathbf{V}^{\text{gls}} = (\mathbf{W}^{\text{gls}})^{-1} = \text{diag}\left(\mathbf{V}^{ABCD}, s^2_{\hat{d}_{AB}}, s^2_{\hat{d}_{BC}}\right) = \begin{bmatrix} 0.037 & 0.017 & 0 & 0 \\ 0.017 & 0.051 & 0 & 0 \\ 0 & 0 & 0.100 & 0 \\ 0 & 0 & 0 & 0.190 \end{bmatrix}$$

Our vector containing the effect size parameters is then $\mathbf{d} = (d_{AB}, d_{AC}, d_{BC})'$. Further, we will choose $\mathbf{d}_{\text{b}} = (d_{AB}, d_{AC})'$ to be our basic parameter vector and $d_{\text{f}} = (\mathbf{d}_{BC})'$ to be our functional parameter vector. As such:

$$\mathbf{d} = \begin{pmatrix} \mathbf{d}_{\mathrm{b}} \\ \mathbf{d}_{\mathrm{f}} \end{pmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 1 \end{bmatrix} \mathbf{d}_{\mathrm{b}} = \mathbf{H}\mathbf{d}_{\mathrm{b}}$$

so that

$$\mathbf{y}^{\mathrm{gls}} = \begin{pmatrix} \hat{d}_{AB\cdot} \\ \hat{d}_{AC\cdot} \\ \hat{d}_{BC} \\ \hat{d}_{AB} \end{pmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 1 \\ 1 & 0 \end{bmatrix} \begin{pmatrix} d_{AB} \\ d_{AC} \end{pmatrix} = \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \\ \mathbf{h}_1 \end{bmatrix} \begin{pmatrix} d_{AB} \\ d_{AC} \end{pmatrix} = \mathbf{X}\mathbf{d}_{\mathrm{b}}$$

and finally

$$\hat{\mathbf{d}}^{\mathrm{gls}} = \begin{pmatrix} \hat{d}_{AB}^{\mathrm{gls}} \\ \hat{d}_{AC}^{\mathrm{gls}} \\ \hat{d}_{BC}^{\mathrm{gls}} \end{pmatrix} = \mathbf{H}(\mathbf{X}'\mathbf{W}^{\mathrm{gls}}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}^{\mathrm{gls}}\mathbf{y}^{\mathrm{gls}} = \begin{pmatrix} -5.206 \\ -12.505 \\ -7.299 \end{pmatrix}$$

Now we will use the contrast summary measures version of the dataset and use the ENT method to generate results. First recall that $\mathbf{y}^{\mathrm{ent}}$ is the 12×1 vector containing all the sample mean differences. Next, we will form the edge-vertex incidence matrix, **B**. Compounds A, B, and C will correspond to columns 1, 2, and 3, respectively, and the following matrices are formed. So:

$$\mathbf{B}_{ABC} = \begin{bmatrix} 1 & -1 & 0 \\ 1 & 0 & -1 \\ 0 & 1 & -1 \end{bmatrix}, \mathbf{B}_{BC} = [0 \quad 1 \quad -1], \mathbf{B}_{AB} = [1 \quad -1 \quad 0]$$

so that

$$\mathbf{B} = \begin{bmatrix} \mathbf{B}_{ABC} \\ \mathbf{B}_{ABC} \\ \mathbf{B}_{BC} \\ \mathbf{B}_{AB} \\ \mathbf{B}_{BC} \\ \mathbf{B}_{ABC} \end{bmatrix}$$

and

$$\mathbf{y}^{\text{ent}} = \mathbf{B}\theta^{\text{treat}} + \boldsymbol{\epsilon}$$

$$\begin{pmatrix} \delta_{11} \\ \delta_{12} \\ \delta_{13} \\ \vdots \\ \delta_{61} \\ \delta_{62} \\ \delta_{63} \end{pmatrix} = \begin{bmatrix} \mathbf{B}_{ABC} \\ \vdots \\ \mathbf{B}_{ABC} \end{bmatrix} \begin{pmatrix} \theta_A \\ \theta_B \\ \theta_C \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \vdots \\ \epsilon_{10} \\ \epsilon_{11} \\ \epsilon_{12} \end{pmatrix} = \begin{pmatrix} d_{AB} \\ d_{AC} \\ d_{BC} \\ \vdots \\ d_{AB} \\ d_{AC} \\ d_{BC} \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \vdots \\ \epsilon_{10} \\ \epsilon_{11} \\ \epsilon_{12} \end{pmatrix}$$

Since design ABC has three treatment arms we must form a sub edge-vertex incidence matrix $\underline{B}_{ABC}$.

$$\underline{\mathbf{B}}_{ABC} = \begin{bmatrix} 1 & -1 & 0 \\ 1 & 0 & -1 \\ 0 & 1 & -1 \end{bmatrix}$$

Next, we form a sub variance matrix $\underline{\mathbf{V}}_i$ for each experiment with design ABC $(i = 1,2,6)$.

$$\underline{\mathbf{V}}_1 = \begin{bmatrix} 0 & 0.141 & 0.142 \\ 0.141 & 0 & 0.159 \\ 0.142 & 0.159 & 0 \end{bmatrix}, \underline{\mathbf{V}}_2 = \begin{bmatrix} 0 & 0.085 & 0.134 \\ 0.085 & 0 & 0.137 \\ 0.134 & 0.137 & 0 \end{bmatrix},$$

$$\underline{\mathbf{V}}_6 = \begin{bmatrix} 0 & 0.122 & 0.194 \\ 0.122 & 0 & 0.202 \\ 0.194 & 0.202 & 0 \end{bmatrix}$$

Finally, we compute $\mathbf{L}_i^+$ and form $\mathbf{W}_i$ for each experiment with design ABC $(i = 1,2,6)$. We will leave this process for the reader to complete. Finally, with these computations completed we can now form $\mathbf{W}^{\text{ent}}$, given by:

$$\mathbf{W}^{\text{ent}} = diag(4.894, 4.823, 3.988, 9.529, \ldots, 6.007, 6.677, 3.183, 2.795)$$

A few entries have been omitted, but the reader can confirm that all entries of $\mathbf{W}^{\text{ent}}$ are non-negative. Finally, after computing $\mathbf{L} = \mathbf{B}'\mathbf{W}^{\text{ent}}\mathbf{B}$, we arrive at the result:

$$\begin{pmatrix} \hat{d}_{AB}^{\text{ent}} \\ \hat{d}_{AC}^{\text{ent}} \\ \hat{d}_{\text{BC}}^{\text{ent}} \\ \vdots \\ \hat{d}_{AB}^{\text{ent}} \\ \hat{d}_{AC}^{\text{ent}} \\ \hat{d}_{BC}^{\text{ent}} \end{pmatrix} = \mathbf{B}\mathbf{L}^{+}\mathbf{B}'\mathbf{W}^{\text{ent}} \begin{pmatrix} \delta_{11} \\ \delta_{31} \\ \delta_{32} \\ \vdots \\ \delta_{34} \\ \delta_{35} \\ \delta_{36} \end{pmatrix} = \begin{pmatrix} -5.206 \\ -12.505 \\ -7.299 \\ \vdots \\ -5.206 \\ -12.505 \\ -7.299 \end{pmatrix}$$

Here it is seen that $\hat{d}_{c_1c_2}^{\text{ent}} = \hat{d}_{c_1c_2}^{\text{ent}}$ for every possible choice of c_1 and c_2. In other words, the effect size parameter estimates for each method are equivalent. We leave it to the reader to confirm that the variances for all the effect size parameter estimates are also the same.

2.3.3 Possible Method Equivalence Requirements

We now propose conditions for which the datasets may be equivalent. We believe an important condition for the methods to be equivalent is that the weights generated for the electrical network theory method be positive. This is because in the theoretical framework for which the method is based on, the weights should always be positive.

Negative weights are directly connected to how consistent variances are in the network. In Dataset A, note that the variances for any initial effect size estimate including treatment arm A in a contrast summary measures dataset will be much higher than the variances for effect size estimates which do not.

Note that this fact alone does not in essence violate the theoretical assumptions necessary since the method assumes each effect size parameter estimate will have a

distinct variance. Still, it does ultimately play a role in producing a negative weight in $\mathbf{W}^{\text{ent}}$ which violates the underlying theoretical framework. As such, in order to be sure that the methods are equivalent, the variances should appear to be consistent. Further, after analysis is conducted, $\mathbf{W}^{\text{ent}}$ should not contain any negative entries.

Another factor that may control whether the methods will yield equivalent results is the amount of experiments with similar designs. In Dataset A there is only one experiment with the multi-arm design $ABCD$ whereas in Dataset B there are three experiments with the multi-arm design ABC. Further, in the Dataset A there was only one experiment that contained treatment arms C and D. Even when the designs are distinct, each experiment in Dataset B has at least one treatment arm in common with another experiment.

This high level of inconsistency in variances can be common in preclinical experiments with few experiments and many treatment arms, each with only a few subjects. More so, these preclinical experiments can also have wildly different designs as they develop new compounds to test and stop testing old compounds that have been deemed inefficient. Both of these qualities lead us to suggest that if analysts desire to use network meta-analysis on this type of data, the generalized least squares method should be used.

There may be other conditions governing whether the methods will be equivalent that have not been discovered. As such, the generalized least squares method of network meta-analysis should be used whenever possible to ensure the most optimal results in terms of SSE.

2.4 Simulation Results Comparing Network Meta-Analysis to Mixed-Effect Linear Models

In this section we provide simulation results which compare the GLS method of network meta-analysis to mixed-effects linear models. We simulated data by using datasets from two different sets of experiments. The dataset used in Section 2.4.1 is artificial data that was simulated by the author, and the dataset used in Section 2.4.2 is the data from a set of non-clinical experiments from a pharmaceutical company.

To begin, we fit a mixed-effects linear model with treatment arm as a fixed effect and experiment as a random effect. We then find the predicted values and the residuals, $\hat{\mathbf{y}} = \left(\hat{y}_{11}, \hat{y}_{12}, \dots, \hat{y}_{Mn_M}\right)'$ and $\boldsymbol{\varepsilon} = \left(\varepsilon_{11}, \varepsilon_{12}, \dots, \varepsilon_{Mn_M}\right)'$, respectively, for each dataset. This model will contain the "true" coefficients that each method should be estimating.

Then we create the simulated datasets, $\mathbf{y}^{\text{boot}}$ and $\mathbf{y}^{\text{norm}}$. $\mathbf{y}^{\text{boot}}$ is created by adding residuals to the elements of $\hat{\mathbf{y}}$ by bootstrapping elements of $\boldsymbol{\varepsilon}$. $\mathbf{y}^{\text{norm}}$ is created by adding residuals to the elements of $\hat{\mathbf{y}}$ by randomly sampling from $\text{N}(0, \sigma_{\boldsymbol{\varepsilon}}^2)$, where $\sigma_{\boldsymbol{\varepsilon}}$ is the standard deviation of $\boldsymbol{\varepsilon}$. We then used both methods 1000 times to create and analyze data.

In each simulation, we compare the GLS method of network meta-analysis and mixed effects linear model by assessing the bias, mean squared error, and 95% confidence interval coverage rate for each individual estimate. It is well known that random effects meta-analysis models perform better than fixed effects meta-analysis models. As such, the pooled random effects model given is used to generate estimates for the GLS method instead of the fixed effects estimates described in Section 2.2.1. This

model uses the heterogeneity estimate $\hat{\tau}^2_{pooled}$ as defined in (Lu, Welton, Higgins, White, & Ades, 2011).

2.4.1 Simulated Data with Moderately Low Sample Sizes

In this section we explore the simulation results with moderately low sample sizes which would be relatively high in the context of preclinical trials for each experiment, decide indirect comparisons are of interest, and ensure that all experiment designs are represented more than once. The dataset for this section was simulated by the author.

The data was created according to the following simulation model:

$$y_{ijk} \sim N(\mu_{ij}, 3);\ \mu_{ij} \sim \mathrm{N}(\theta_j, 0.5) \text{ for } i = 1,2,\dots,N;\ j \in \mathcal{A};\ k = 1,2,\dots,n_{ij}$$

where

- μ_{ij} is the true mean response for treatment arm j in experiment i
- θ_j is the true mean response for treatment arm j, N is the number of experiments
- $\mathcal{A}$ is the set of possible treatment arms
- n_{ij} is the number of subjects receiving treatment arm j in experiment i

So y_{ijk} is the response for subject k in experiment i after receiving treatment arm j. Note that μ_{ij} is randomly sampled from a normal distribution with θ_j at the center to reflect the heterogeneity that is often present between different experiments. For the dataset we simulated, $N = 8, \mathcal{A} = \{A, B, C, D, E\}, \theta_A = 10, \theta_B = 8, \dots \theta_E = 0$, and $n_{ij} = 50$ for every combination of i and j defined by the experiment designs given in Figure 13.

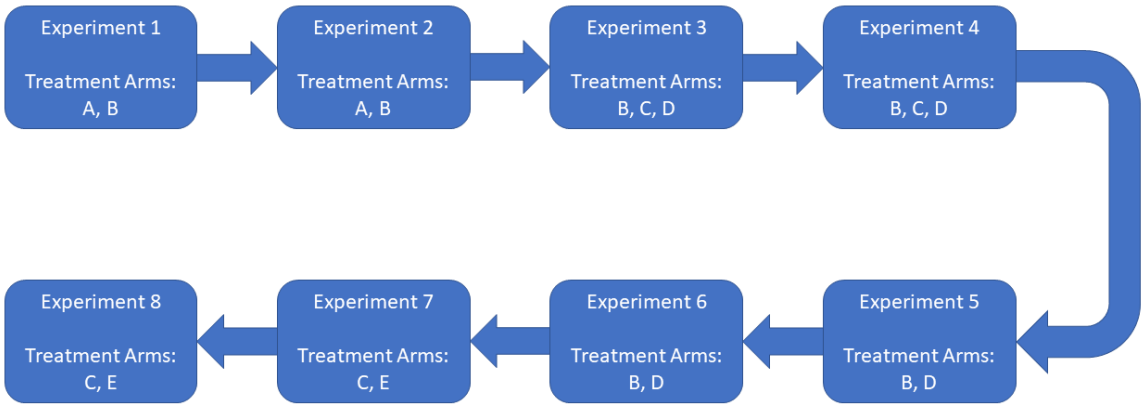

Figure 13: Experiment Designs for Simulated Data

Below are multiple tables providing the simulation results for the coefficients comparing the difference between the treatment arms A, B, C, and D to treatment arm E. Table 6 compares the bias, Table 7 compares the mean squared error, and Table 8 compares the 95% confidence interval for the different methods for each type of simulated dataset. Note that this is only a handful of the effect size estimates given by the network meta-analysis and three correspond to indirect comparisons.

According to the simulation results, the absolute value of the bias for the network meta-analysis results is less than the absolute value of the bias for the mixed effects linear model results for at least one type of data simulation for every effect size parameter.

Parameter	MELM Norm	NMA Norm	MELM Boot	NMA Boot
d_{AE}	0.036	0.033	0.006	-0.008
d_{BE}	-0.032	0.033	-0.045	0.012
d_{CE}	-0.022	0.017	-0.022	0.014
d_{DE}	-0.064	0.034	-0.087	0.005

Table 6: Bias Results

Parameter	MELM Norm	NMA Norm	MELM Boot	NMA Boot
d_{AE}	0.273	0.376	0.269	0.377
d_{BE}	0.182	0.25	0.177	0.244
d_{CE}	0.118	0.135	0.115	0.135
d_{DE}	0.198	0.254	0.192	0.249

Table 7: Mean Squared Error Results

Parameter	MELM Norm	NMA Norm	MELM Boot	NMA Boot
d_{AE}	0.96	0.953	0.967	0.973
d_{BE}	0.96	0.947	0.977	0.973
d_{CE}	0.953	0.95	0.94	0.947
d_{DE}	0.947	0.943	0.953	0.963

Table 8: 95% Confidence Interval Coverage Rate Results

The mean squared error for the network meta-analysis results is greater than the mean squared error for both data simulation types for every effect size parameter, which suggests that there is more volatility for network meta-analysis estimates than mixed effect linear model estimates. The 95% confidence interval coverage rate for the network meta-analysis results is greater than the 95% confidence interval coverage rate for the mixed effects linear model results for the bootstrap type of data simulation for all but one effect size parameter (d_{BE}).

Note that the effect size parameter estimates corresponding to indirect comparisons (d_{AE}, d_{BE}, d_{DE}) from the network meta-analysis results suffer in particular compared to mixed effects linear model results. Still, in terms of bias and 95% confidence interval coverage rate they perform just as well as d_{CE}, an effect size parameter corresponding to a direct comparison.

2.4.2 True Preclinical Data

The dataset for this section involves four non-clinical experiments that were conducted independently in 2015. The goal of these experiments was to determine if any of the developed compounds (Compounds B, C, D, E, F, and G) provided a statistically significant increase in a desired response mechanism according to a pre-specified metric compared to a compound serving as a negative control (Compound A). See Figure 14 for the experiment design for this dataset.

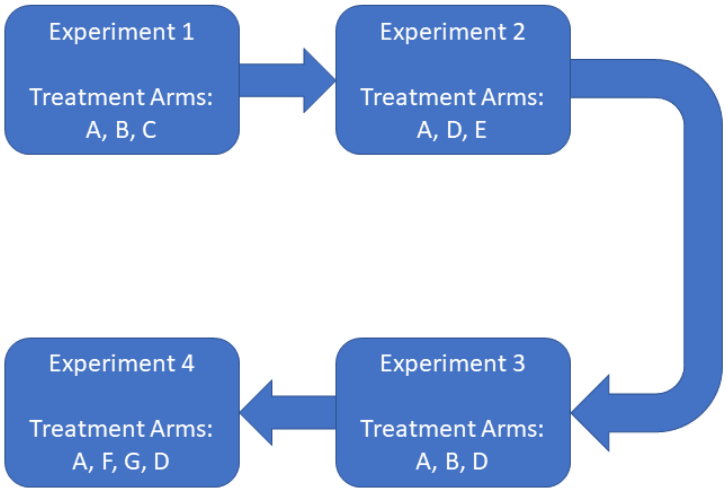

Figure 14: Experiment Designs for True Preclinical Data

Below are multiple tables providing the simulation results for the coefficients comparing the difference between the Compounds B, C, D, E, F, and G to Compound A. Table 9 compares the bias, Table 10 compares the mean squared error, and Table 11 compares the 95% confidence interval for the different methods for each type of simulated dataset. Note that this is only a handful of the effect size estimates given by network meta-analysis and they are all direct comparisons.

Parameter	MELM Norm	NMA Norm	MELM Boot	NMA Boot
d_{BA}	-0.568	-0.06	-0.476	0.084
d_{CA}	-0.755	-0.048	-0.844	-0.133
d_{DA}	0.237	0.049	0.205	0.002
d_{EA}	0.124	0.013	0.006	-0.262
d_{FA}	0.621	0.063	0.618	-0.004
d_{GA}	0.56	0.002	0.555	-0.067

Table 9: Bias Results

Parameter	MELM Norm	NMA Norm	MELM Boot	NMA Boot
d_{BA}	5.65	6.159	5.694	5.356
d_{CA}	10.013	11.922	10.179	11.789
d_{DA}	3.532	3.841	3.28	3.327
d_{EA}	9.371	11.437	8.79	10.695
d_{FA}	7.743	8.521	7.294	7.572
d_{GA}	6.694	9.456	6.39	8.938

Table 10: Mean Squared Error Results

Parameter	MELM Norm	NMA Norm	MELM Boot	NMA Boot
d_{BA}	0.926	0.804	0.941	0.807
d_{CA}	0.932	0.852	0.915	0.852
d_{DA}	0.949	0.852	0.962	0.873
d_{EA}	0.946	0.846	0.952	0.865
d_{FA}	0.936	0.898	0.934	0.905
d_{GA}	0.916	0.903	0.932	0.936

Table 11: 95% Confidence Interval Coverage Rate Results

It is worth noting that although the set of experiments used in this simulation each had balanced designs, they all suffered from very small sample sizes. Experiments 1, 2, and 3 each only had three subjects per treatment arm, and Experiment 4 only had five different subjects per treatment arm. Also, in this experiment the goal was to generate estimates for a set of effect size parameters which were all direct comparisons. Finally, each time a new experiment was conducted the designs were completely changed.

According to the simulation results, the absolute value of the bias for the network meta-analysis results is less than the absolute value of the bias for the mixed effects linear model results for at least one type of data simulation for every effect size parameter. This suggests that network meta-analysis does a better job of returning the point estimate on average than mixed effects linear models.

The mean squared error for the network meta-analysis results is greater than the mean squared error for the mixed effects linear model results for both data simulation types for all but one effect size parameter (d_{BA}), which suggests that there is a higher level of volatility in the estimates provided by network meta-analysis when compared to the mixed effects linear model estimates.

This is further demonstrated by the fact that the 95% confidence interval coverage rate for the network meta-analysis results is less than the 95% confidence interval coverage rate for the mixed effects linear model for both types of data simulation for all but one effect size parameter (d_{GA}). This lack of coverage may be a cause for concern and cast doubt on the utility of the method for this type of dataset.

2.5 Discussion

We have described in rigorous detail the steps for conducting two different methods of mixed treatment comparisons meta-analysis, more succinctly dubbed network meta-analysis, as well as compared them in the context of datasets with small sample sizes which are often typical in preclinical data.

We have demonstrated that while for some datasets, the electrical network theory method proposed in (Rücker & Schwarzer, 2012) is equivalent to the generalized

least squares method given in (Lu, Welton, Higgins, White, & Ades, 2011), there exists a class of datasets such that the two methods are not equivalent. As such, the generalized least squares method, along with the correct set of baseline choices to ensure SSE minimization, should be used instead whenever possible.

We have developed the R package *NMA* which contains functions for conducting the generalized least squares method and finding the set of baseline choices which minimizes the SSE. The function which conducts network meta-analysis also allows the user to choose the electrical network theory method, which is already currently available in the R package *netmeta* developed by Rücker et al. (Rücker, Schwarzer, Krahn, & König, 2018). There are also functions in the package which provide the graphical interpretation of the treatment arms in the network as well as a forest plot which allows for comparisons of the final estimates yielded from the network meta-analysis. Currently, *netmeta* is the only package in R which performs network meta-analysis within a frequentist framework (Neupane, et al., 2014). Consequently, *NMA* is currently the only R package that allows for performing network meta-analysis using the generalized least squares method.

Finally, we provided simulation results which showed that with the correct design of experiments, network meta-analysis using only summary measures provides similar results to mixed effects linear models using every data point in terms of bias and coverage rate. While mixed effects linear models provide noticeably better results in terms of mean squared error, the increase in variance of the estimates is to be expected since the reduction from an entire dataset to its summary measures results in a loss of information. This should intuitively lead to more uncertainty in the estimates.

More work can be done for finding the specific class of data for which the GLS and ENT methods are not equivalent. Methods could also be developed to determine precisely when a given set of experiments include variances which are not consistent enough for the GLS and ENT method to produce equivalent results. Finally, more simulations can be created to study how the results fare for binary or survival outcomes for the type of sets of experiments generated in preclinical environment.

Chapter 3: Power and Sample Size Calculations for Designing Experiments with Ordinal Categorical Responses with Small Range Scales

3.1 Introduction

Power calculations for hypothesis tests are a crucial part of designing both clinical and non-clinical trials (Amaratunga, 1999). Scientists rely heavily on the results from these calculations to decide the number of subjects they will collect for experiments. Since there is typically a higher cost, monetary or otherwise, for conducting experiments for large sample sizes, there is an interest in choosing a small number of subjects; however, the ability for most hypothesis tests to detect a statistically significant difference in treatment effect can depend heavily on the sample size. As such, the method for calculating power to determine the optimal sample size must be carefully chosen.

In experiments with ordinal categorical data, the data are assumed to follow a multinomial distribution with a probability assigned to each category. As a motivating example, suppose that scientists at a pharmaceutical company are attempting to develop a drug for reducing pain. According to the pain metric developed by the scientists, any subject's pain falls into one of three categories: 1 is no pain, 2 is mild pain, and 3 is extreme pain. The scientists have completed a drug they plan to test, Compound A. They will deem the compound successful for a patient if their pain score is reduced by at least one point. Finally, they would like to be able to ensure that they collect a large enough number of subjects so that the hypothesis test used to analyze the data has at least 90% power.

In this paper we explore a method for calculating power for experiments such as these where there are between three and five different ordinal categories. This method creates and shifts the quantiles of a standard normal distribution to define alternative multinomial distributions. In Section 3.2 we explain the method in the context of the motivating example. Section 3.2 expands the method by adding multiple types of uncorrelated ordinal categorical variables. Section 3.4 expands the method further by introducing correlation between pairs of ordinal categorical variables, as well as the derivation of a new estimator to estimate this correlation. This section also offers simulation results comparing this new correlation estimator, ξ, to Spearman's ρ and Kendall's τ in terms of bias. Section 3.5 applies the method to the dataset provided by a pharmaceutical company which served as the motivation for developing this method. Section 3.6 discusses this new method of power calculation and how it may be expanded further. This section also describes an R package that has been developed to apply the method.

3.2 Power Calculation for One Variable

Continuing our motivating example, suppose the scientists conducting the experiment have already taken an initial sample of N_0 subjects and recorded their pain scores to obtain an estimate of how much pain subjects experience when left untreated. Let n_k be the number of subjects that exhibit a pain score of k for $k = 1,2,3$. The scientists plan to use the outcome yielded from this initial sample to help design a future experiment.

We will assume that this future experiment will be randomized with a balanced design. N subjects will be assigned to the “Placebo” group (Group P) and receive a

placebo as their treatment and N subjects will be assigned to the "Treatment" group (Group T) to recieve Compound A as their treatment. N will be chosen to achieve the amount of power desired by the scientists.

Let $\mathbf{x} = (\mathbf{x}^P, \mathbf{x}^T)' = (x_1^P, \dots, x_N^P, x_1^T, \dots, x_N^T)'$ where x_i^G is the pain score for subject i in group G prior to receiving a treatment. We then assume,

$$x_i^G \sim Multi(1, \mathbf{p}) \text{ for } i = 1,2, \dots, N;\ G \in \{P, T\}$$

where $\mathbf{p} = (p_1, p_2, p_3)'$ and $p_k = \frac{n_k}{N_0}$, the relative frequency of subjects achieving pain score k in the initial sample for $k = 1,2,3$.

After receiving treatment, pain scores are recorded again to create the $\mathbf{y} = (\mathbf{y}^P, \mathbf{y}^T)' = (y_1^P, \dots, y_N^P, y_1^T, \dots, y_N^T)'$ where y_i^G is the pain score for subject i in group G after receiving the treatment for group G. We then further assume,

$$y_i^P \sim Multi(1, \mathbf{p}) \text{ for } i = 1,2, \dots, N$$

$$y_i^T \sim Multi(1, \mathbf{p}^T) \text{ for } i = 1,2, \dots, N$$

Note that the probability parameter for group P remains the same because we are assuming the placebo will have no effect on the probability of achieving any particular pain score. On the other hand, the probability parameter for group T has been changed to $\mathbf{p}^T$ because we are assuming that Compound A will have some effect on the probability of achieving any particular pain score. Specifically, the probability for patients achieving higher scores decreases and the probability for patients achieving lower scores increases after receiving treatment with Compound A.

Let $\mathbf{d} = \mathbf{x} - \mathbf{y}$ so that $\mathbf{d}$ contains the change in pain score for all $2N$ subjects. Recall that Compound A is considered successful for a subject if their pain score is reduced by at least one point; let this be the "success" condition. The subjects are then stratified into a contingency table with cells based upon two factors: the treatment group and whether the success condition is satisfied. The table that would be created for our motivating example is shown by Table 12. Let $\mathbb{1}(\mathcal{A})$ be the indicator function so that $\mathbb{1}(\mathcal{A}) = 1$ when $\mathcal{A}$ is true and $\mathbb{1}(\mathcal{A}) = 0$ otherwise.

Group \ Successful?	Yes	No
Placebo	$\sum_{i=1}^{N} \mathbb{1}(\mathbf{d}_i^P \geq 1)$	$\sum_{i=1}^{N} \mathbb{1}(\mathbf{d}_i^P < 1)$
Treatment	$\sum_{i=1}^{N} \mathbb{1}(\mathbf{d}_i^T \geq 1)$	$\sum_{i=1}^{N} \mathbb{1}(\mathbf{d}_i^T < 1)$

Table 12: Contingency Table for Testing Compound A

We then use Fisher's exact test to discover if there is a statistically significant difference between the placebo and Compound A according to the success condition (Fisher, 1935). As such, our power calculation method is based on the results yielded from simulating data according to our assumptions and using Fisher's exact test to analyze the simulated data for many iterations, and then dubbing the proportion of times the hypothesis test rejected the null hypothesis as our power for the hypothesis test.

The most important aspect of simulating data in this manner is how to choose $\mathbf{p}^T$, the probability parameter for group T. Each element of $\mathbf{p}^T$ can range from 0 to 1 in various ways. Without a concise, uniform way to choose this parameter the possibilities can be overwhelming.

We propose choosing $\mathbf{p}^T$ by shifting the elements of $\mathbf{p}$ in a uniform, controlled manner using the standard normal distribution. This proposition was inspired by the probit model (Bliss, 1934). The algorithm for shifting the probability vector for one ordinal categorical variable is described in **ALGORITHM 4**. We also illustrate the shifting process in Figure 15. Let $Z \sim N(0,1)$.

ALGORITHM 4: Generate $\mathbf{p}^T$ given initial probability vector parameter $\mathbf{p}$ and shift δ.

1. Divide the area under the probability density function for a standard normal distribution into K portions using vertical lines $q_0, q_1, q_2, \dots, q_K$. These lines divide the probability density function so that the portion which lies between q_{k-1} and q_k has p_k area under the curve for $k = 1,2, \dots, K$. More specifically, q_k is a vertical line that intersects the x-axis at $(Q_k, 0)$ where Q_k satisfies $P(Q_{k-1} < Z < Q_k) = p_k$. Note that $Q_0 = -\infty$ and $Q_K = \infty$.
2. Uniformly shift all the vertical lines with δ to create new vertical lines, q'_k, which intersect the x-axis at $(Q_k + \delta, 0)$.
3. The area under the curve which lies between vertical lines q'_{k-1} and q'_k corresponds to p_k^T, the hypothesized probability that a subject which has received treatment will achieve level k. This is also the k^{th} element of $\mathbf{p}^T$.

Note that each individual quantile $Q_k + \delta$ could also be shifted additionally by some γ_k. In the simplest setting which we provide above, we set $\gamma_k = 0$ for $k = 1, 2, \dots, K$.

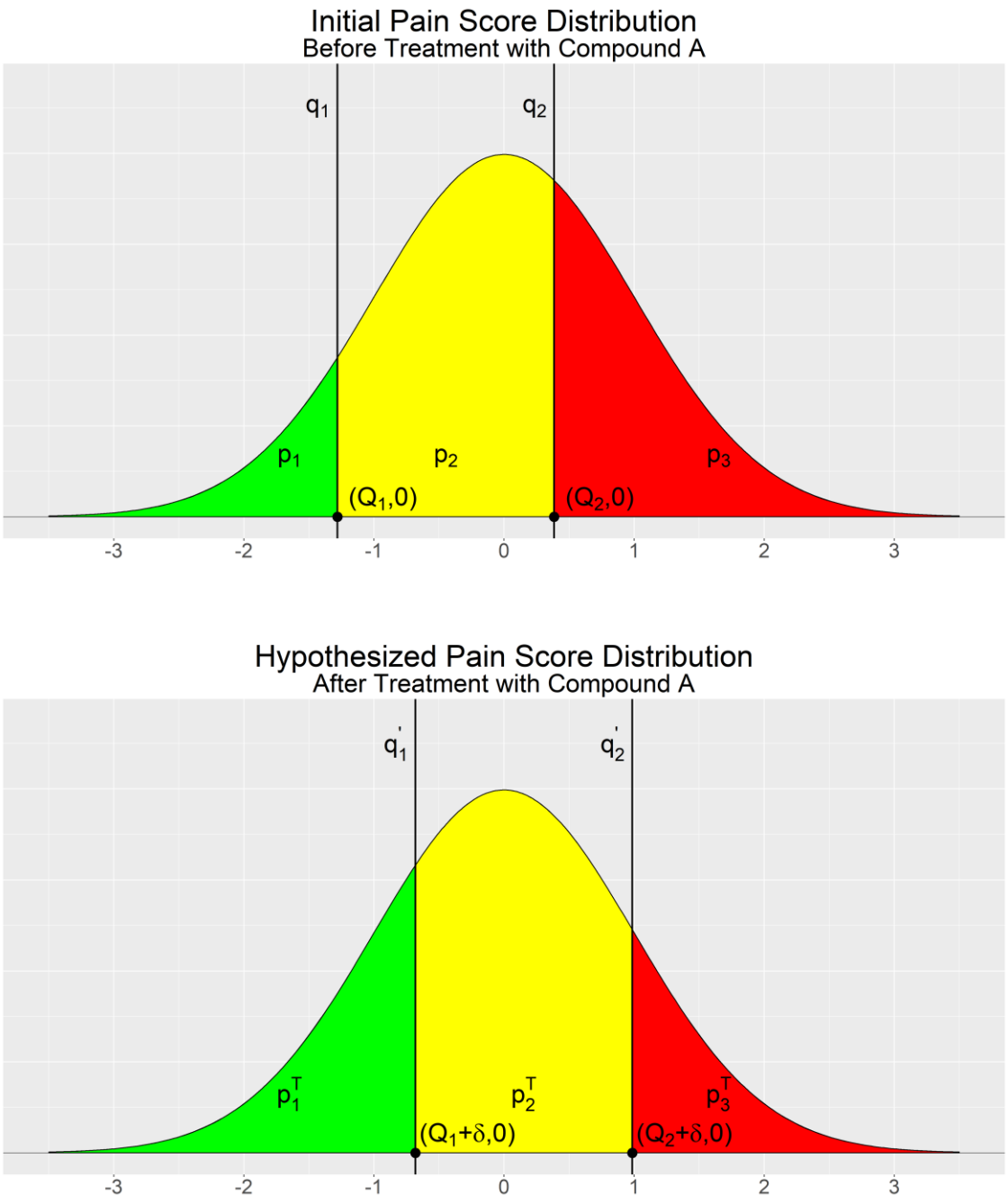

Figure 15: Shifting Process for One Ordinal Categorical Variable

The $\mathbf{p}^T$ generated by δ is then used to simulate the data for the power calculations. By varying the sample size N and shift δ, power curves for a particular

success condition can be generated. The algorithm for calculating power for any combination of N, $\mathbf{p}$, δ, and success condition S is given in **ALGORITHM 5**.

ALGORITHM 5: Perform power calculations for one ordered categorical variable given sample size N, initial probability parameter $\mathbf{p}$, quantile shift δ, success condition S, and number of iterations u.

1. Draw $2N$ random observations from $Multi(1, \mathbf{p})$ to simulate $\mathbf{x} = (\mathbf{x}^P, \mathbf{x}^T)'$.
2. Conduct **ALGORITHM 4** with δ and p to generate $\mathbf{p}^T$.
3. Draw N random observations from $Multi(1, \mathbf{p})$ and draw N random observations from $Multi(1, \mathbf{p}^T)$ to simulate $\mathbf{y}^P$ and $\mathbf{y}^T$, respectively. Then form $\mathbf{y} = (\mathbf{y}^P, \mathbf{y}^T)'$.
4. Calculate $\mathbf{d} = \mathbf{x} - \mathbf{y}$ and form a contingency table stratified by success condition S and treatment group.
5. Conduct Fisher's exact test.
6. Perform steps 1 through 5 for u iterations. The power for $(N, \delta, \mathbf{p}, S)$ is given by the proportion of iterations where Fisher's exact test rejects the null hypothesis.

Sometimes in experiments such as these there are multiple ordered categorical variables that are being analyzed together. A natural extension of this method should be able to take this into account. The next section does so under the assumption that these variables are all uncorrelated.

3.3 Power Calculation for M Uncorrelated Variables

This method can also be applied for multiple uncorrelated variables. As a motivating example for this section, suppose the scientists are developing another drug to reduce pain in three independent and mutually exclusive sections of subjects' bodies:

head, upper body, and lower body. The drug they have completed and plan to test is Compound B.

To test the efficacy of this new compound, they are conducting a new experiment where a pain score is assigned to each body section, so $M = 3$. The scores are then added to produce a total pain score. Note that the number of levels for each variable do not necessarily have to be the same, but for simplicity every section of the body in this experiment will have the same possible pain scores as in the example given in Section 2.

The goal of Compound B is to reduce pain in all three sections of the body as much as possible. As such, Compound B is considered successful for a subject if their total pain score is reduced by at least two points and there is a reduction in pain for at least two different body sections. Success is defined this way to ensure that compound B is not deemed effective if it only reduces pain in one section of the body, even if that reduction is by a significant amount.

Once again, the scientists have taken an initial sample of N_0 subjects and recorded their pain scores for each body section to obtain an estimate of how much pain each subject experiences at each body section. Let n_{mk} be the number of subjects with pain score k at body section m for $k = 1,2,3$ and $m \in \{H, U, L\}$ where H corresponds to a subject's head, U to their upper body, and L to their lower body.

This future experiment will also be randomized with a balanced design so that N subjects are assigned to Group P and will receive a placebo and N subjects are assigned to group T and will receive Compound B. Let $\mathbf{X}$ be given by:

$$\mathbf{X} = [\mathbf{x_1} \quad \mathbf{x_2} \quad \mathbf{x_3}] = \begin{bmatrix} \mathbf{x_1^P} & \mathbf{x_2^P} & \mathbf{x_3^P} \\ \mathbf{x_1^T} & \mathbf{x_2^T} & \mathbf{x_3^T} \end{bmatrix} = \begin{bmatrix} x_{H\mathbf{1}}^P & x_{U\mathbf{1}}^P & x_{L\mathbf{1}}^P \\ \vdots & \vdots & \vdots \\ x_{HN}^P & x_{UN}^P & x_{LN}^P \\ \vdots & \vdots & \vdots \\ x_{H\mathbf{1}}^T & x_{U1}^T & x_{L\mathbf{1}}^T \\ \vdots & \vdots & \vdots \\ x_{HN}^T & x_{UN}^T & x_{LN}^T \end{bmatrix}$$

where x_{mi}^G is the pain score for subject i at body section m in group G prior to receiving any treatment. We then assume:

$$x_{mi}^G \sim Multi(1, \mathbf{p}_m) \text{ for } m \in \{H, U, L\};\ i = 1,2, \dots, N;\ G \in \{P, T\}$$

where $\mathbf{p}_m = (p_{m1}, p_{m2}, p_{m3})'$ and $p_{mk} = \frac{n_{mk}}{N_0}$, the relative frequency of subjects achieving pain score k at body section m in the initial sample for $k = 1,2,3;\ m \in \{H, U, L\}$. After receiving treatment, pain scores are recorded again to create **Y** which is given by:

$$\mathbf{Y} = [\mathbf{y_1} \quad \mathbf{y_2} \quad \mathbf{y_3}] = \begin{bmatrix} \mathbf{y_1^P} & \mathbf{y_2^P} & \mathbf{y_3^P} \\ \mathbf{y_1^T} & \mathbf{y_2^T} & \mathbf{y_3^T} \end{bmatrix} = \begin{bmatrix} y_{H\mathbf{1}}^P & y_{U\mathbf{1}}^P & y_{L\mathbf{1}}^P \\ \vdots & \vdots & \vdots \\ y_{HN}^P & y_{UN}^P & y_{LN}^P \\ \vdots & \vdots & \vdots \\ y_{H\mathbf{1}}^T & y_{U1}^T & y_{L\mathbf{1}}^T \\ \vdots & \vdots & \vdots \\ y_{HN}^T & y_{UN}^T & y_{LN}^T \end{bmatrix}$$

Where y_{mi}^G is the pain score at body portion m for subject i in group G after receiving the treatment for group G. We then further assume,

$$y_{mi}^P \sim Multi(1, \mathbf{p}_m) \text{ for } i = 1,2, \dots, N;\ m \in \{H, U, L\}$$

$$y_{mi}^T \sim Multi(1, \mathbf{p}_m^T) \text{ for } i = 1,2, \dots, N;\ m \in \{H, U, L\}$$

Let $\mathbf{D} = \mathbf{X} - \mathbf{Y}$ so that $\mathbf{D}$ contains the change in pain score for all three sections of the body for all $2N$ subjects. Further, let $\mathbf{D}$ be given by:

$$\mathbf{D} = \begin{bmatrix} \mathbf{d}_{\mathbf{1}}^{P} \\ \vdots \\ \mathbf{d}_{N}^{P} \\ \mathbf{d}_{\mathbf{1}}^{T} \\ \vdots \\ \mathbf{d}_{N}^{T} \end{bmatrix} = \begin{bmatrix} d_{H\mathbf{1}}^{P} & d_{U\mathbf{1}}^{P} & d_{L\mathbf{1}}^{P} \\ \vdots & \vdots & \vdots \\ d_{HN}^{P} & d_{UN}^{P} & d_{LN}^{P} \\ \vdots & \vdots & \vdots \\ d_{H\mathbf{1}}^{T} & d_{U1}^{T} & d_{L\mathbf{1}}^{T} \\ \vdots & \vdots & \vdots \\ d_{HN}^{T} & d_{UN}^{T} & d_{LN}^{T} \end{bmatrix}$$

where d_i^G is the 1×3 vector containing the change in pain score for all three sections of the body for subject i in group G and d_{mi}^G is the change in pain score at body section m for subject i in group G.

Recall that the compound is considered successful for a subject if their pain score is reduced by at least two points and there is a reduction in pain for at least two different body sections. The subjects are then once stratified into a contingency table. Let $\mathbf{c}$ be a 3×1 vector with 1 in every entry. The table that would be created is shown by Table 13.

Group \ Successful?	Yes	No
Placebo	a	b
Treatment	c	d

Table 13: Contingency Table for Testing Compound B

where

$$a = \sum_{i=1}^{N} \mathbb{1}(c' d_i^P \geq 2 \cap (d_{mi}^P \geq 1 \cap d_{li}^P \geq 1), m \neq l)$$

$$b = \sum_{i=1}^{N} \mathbb{1}(c' d_i^P < 2 \cup (d_{mi}^P = 0 \cap d_{li}^P = 0), m \neq l)$$

$$c = \sum_{i=1}^{N} \mathbb{1}(c' d_i^T \geq 2 \cap (d_{mi}^T \geq 1 \cap d_{li}^T \geq 1), m \neq l)$$

$$d = \sum_{i=1}^{N} \mathbb{1}(c' d_i^T < 2 \cup (d_{mi}^T = 0 \cap d_{li}^T = 0), m \neq l)$$

After this contingency table is created, we use Fisher's exact test to discover if there is a statistically significant difference between the placebo and Compound B according to the success condition. Once again, the most important part of simulating the data to perform the power calculation is generating $\mathbf{p}_m^T$ for every body section m.

Similar to what is done for one ordinal categorical variable, we propose shifting the elements of each probability vector parameter using independent normal distributions. We are assuming that the normal distributions used for shifting the vectors are independent because we have assumed that the variables are uncorrelated. Let $A = \{m_1, m_2, \dots, m_M\}$ be a set of M ordinal categorical variables, each with $K_1, K_2, \dots, K_M$ levels, respectively. The algorithm for shifting the probability vector for M ordinal categorical variables is described in **ALGORITHM 6**.

ALGORITHM 6: Generate p_m^T given initial probability vector parameters p_m and shifts δ_m for all $m \in A$.

For $l = 1,2, \dots, M$:

1. Divide the area under the probability density function for a standard normal distribution into K_l portions using vertical lines $q_{l0}, q_{l1}, q_{l2}, \ldots, q_{lK_l}$. These lines will divide the probability density function so that the portion which lies between $q_{l(k-1)}$ and q_{lk} has $p_{m_l k}$ area under the curve for $k = 1,2, \ldots, K_l$. More specifically, q_{lk} is a vertical line that intersects the x-axis at $(Q_{lk}, 0)$ where Q_{lk} satisfies $P\left(Q_{l(k-1)} < Z < Q_{lk}\right) = p_{m_l k}$. Note that $Q_{l0} = -\infty$ and $Q_{lK_l} = \infty$.
2. Uniformly shift all the vertical lines with δ_{m_l} to create new vertical lines, q'_{lk}, which intersect the x-axis at $(Q_{lk} + \delta, 0)$.
3. The area under the curve which lies between vertical lines $q'_{l(k-1)}$ and q'_{lk} corresponds to $p^T_{m_l k}$, the hypothesized probability that a subject which has received treatment will achieve level k for variable m_l. This is also the k^{th} element of $\mathbf{p}^T_{m_l}$.

Once again, it should be noted that each individual quantile for each ordinal categorical variable m_l, $Q_{lk} + \delta_l$ could also be shifted additionally by some γ_{lk}. In the simplest setting which we provide above, we set $\gamma_{lk} = 0$ for $k = 1, 2, \ldots, K;\ l = 1, 2, \ldots M$.

The $\mathbf{p}^T_m$ generated by δ_m for all $m \in A$ is then used to simulate the data for the power calculations. The algorithm for calculating power for any combination of N, $\left(\mathbf{p}_{m_1}, \ldots, \mathbf{p}_{m_M}\right)$, $\left(\delta_{m_1}, \ldots, \delta_{m_M}\right)$, and success condition S is given in **ALGORITHM 7**.

ALGORITHM 7: Perform power calculations for M ordinal categorical variables. Given sample size N, initial probability vector parameters $\left(\mathbf{p}_{m_1}, \ldots, \mathbf{p}_{m_M}\right)$, shifts $\left(\delta_{m_1}, \ldots, \delta_{m_M}\right)$, success condition S, and number of iterations u.

1. Draw $2N$ random observations from $Multi(1, \mathbf{p}_m)$ to simulate $\mathbf{x}_m = (\mathbf{x}_m^P, \mathbf{x}_m^T)'$ for all $m \in A$. Then form X.
2. Conduct **ALGORITHM 6** with $(\delta_{m_1}, \ldots, \delta_{m_M})$ and $(\mathbf{p}_{m_1}, \ldots, \mathbf{p}_{m_M})$ to generate $(\mathbf{p}_{m_1}^T, \ldots, \mathbf{p}_{m_M}^T)$.
3. Draw N random observations from $Multi(1, \mathbf{p}_m)$ and draw N random observations from $Multi(1, \mathbf{p}_m^T)$ to simulate $\mathbf{y}_m^P$ and $\mathbf{y}_m^T$, respectively for all $m \in A$. Then form $\mathbf{y}_m = (\mathbf{y}_m^P, \mathbf{y}_m^T)'$ for all $m \in A$. Then finally form $\mathbf{Y}$.
4. Calculate $\mathbf{D} = \mathbf{X} - \mathbf{Y}$ and form a contingency table stratified by success condition S and treatment group.
5. Conduct Fisher's exact test.
6. Perform steps 1 through 5 for u iterations and the power for $(N, \delta_{m_1}, \ldots, \delta_{m_M}, \mathbf{p}_{m_1}, \ldots, \mathbf{p}_{m_M}, S)$ is given by the proportion of iterations where Fisher's exact test rejects the null hypothesis.

Naturally, the more variables included the more computationally intensive this shifting process becomes. Nevertheless, this enables one to easily customize the hypothesized effectiveness of the drug of interest for each variable. For example, if the scientists believe that Compound B will be more effective in reducing pain in a subject's head than their lower body, they could choose the shifts so that $\delta_L < \delta_H$.

As more variables are added, believing that they are all uncorrelated may become too strong of an assumption. As such, the method should be extended to allow for correlation among variables to influence the probability vector parameter after applying the shifts. The next section examines this extension.

3.4 Power Calculation for Pairs of Correlated Variables

Finally, this method can be applied for pairs of correlated ordinal categorical variables. In our final motivating example, suppose the scientists are developing yet another drug to reduce pain in the upper and lower sections of the body only. Also, for this experiment they are assuming that there is some amount of correlation between the amount of pain experienced in the upper body and the amount of pain experienced in the lower body. The drug they have completed and plan to test is Compound C.

To test the efficacy of this new compound, they are conducting a new experiment where a pain score is assigned to the upper and lower body sections and the scores are then added to produce a total pain score. Once again, although it is not necessary, for simplicity both variables will have the same possible pain scores as in the example given in Section 3.2.

The goal of Compound C is to reduce pain in these two sections of the body as much as possible. As such, Compound C is considered successful for a subject if the pain score for each section of the body is reduced by at least one point.

Once again, the scientists have taken an initial sample of N_0 subjects and recorded their pain scores for each body section to obtain an estimate of how much pain each subject experiences at each body section before any treatment is administered. Let n_{jk} be the number of subjects with pain score j at their upper body and pain score k at their lower body for $j = 1,2,3$ and $k = 1,2,3$.

Note that this notation is different from the notation found in Section 3. In this section we are examining the scores for each section simultaneously rather than

individually for each subject. This is because we are assuming that the pain scores for the two sections of the body are correlated,

The new experiment will once again be randomized with a balanced design so that N subjects are assigned to Group P and will receive a placebo and N subjects are assigned to Group T and will receive Compound C. Let X be given by:

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}^P \\ \mathbf{X}^T \end{bmatrix} = \begin{bmatrix} \mathbf{x}_\mathbf{1}^P \\ \vdots \\ \mathbf{x}_N^P \\ \mathbf{x}_\mathbf{1}^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix} = \begin{bmatrix} (x_{U\mathbf{1}}^P, x_{L\mathbf{1}}^P) \\ \vdots \\ (x_{UN}^P, x_{L\boldsymbol{N}}^P) \\ (x_{U\mathbf{1}}^T, x_{L\mathbf{1}}^T) \\ \vdots \\ (x_{UN}^T, x_{L\boldsymbol{N}}^T) \end{bmatrix}$$

where $\left(x_{Ui}^G, x_{Li}^G\right)$ is the pair of pain scores for subject i in group G prior to receiving any treatment. The first entry is the pain score for the upper body and the second entry is the pain score for the lower body. We then assume:

$$x_i^G \sim Multi(1, \mathbf{p}_{UL}) \text{ for } i = 1,2, \dots, N;\ G \in \{P, T\}$$

where $\mathbf{p}_{UL} = (p_{11}, p_{12}, \dots, p_{33})'$ and $p_{jk} = \frac{n_{jk}}{N_0}$, the relative frequency of subjects achieving pain score j in the upper body and pain score k in the lower body for $j = 1,2,3;\ k = 1,2,3$. In this formation of the distribution, each multinomial bin corresponds to a possible pair of pain scores. After receiving the treatment, pain scores are recorded again to create Y which is given by:

$$\mathbf{Y} = \begin{bmatrix} \mathbf{Y}^P \\ \mathbf{Y}^T \end{bmatrix} = \begin{bmatrix} \mathbf{y}_\mathbf{1}^P \\ \vdots \\ \mathbf{y}_N^P \\ \mathbf{y}_\mathbf{1}^T \\ \vdots \\ \mathbf{y}_N^T \end{bmatrix} = \begin{bmatrix} (y_{U\mathbf{1}}^P, y_{L\mathbf{1}}^P) \\ \vdots \\ (y_{UN}^P, y_{L\boldsymbol{N}}^P) \\ (y_{U\mathbf{1}}^T, y_{L\mathbf{1}}^T) \\ \vdots \\ (y_{UN}^T, y_{L\boldsymbol{N}}^T) \end{bmatrix}$$

where $\left(y_{Ui}^{G}, y_{Li}^{G}\right)$ is the pair of pain scores for subject i in group G prior to receiving the treatment for group G. The first entry is the pain score for the upper body and the second entry is the pain score for the lower body. We then further assume,

$$y_i^P \sim Multi(1, \mathbf{p}_{UL}) \text{ for } i = 1,2, \dots, N$$

$$y_i^T \sim Multi(1, \mathbf{p}_{UL}^T) \text{ for } i = 1,2, \dots, N$$

Let **D** contain the change in pain score for both sections of the body for all $2N$ subjects, so that **D** is given by:

$$\mathbf{D} = \begin{bmatrix} \mathbf{D}^P \\ \mathbf{D}^T \end{bmatrix} = \begin{bmatrix} \mathbf{d_1^P} \\ \vdots \\ \mathbf{d_N^P} \\ \mathbf{d_1^T} \\ \vdots \\ \mathbf{d_N^T} \end{bmatrix} = \begin{bmatrix} (d_{U\mathbf{1}}^P, d_{L\mathbf{1}}^P) \\ \vdots \\ (d_{UN}^P, d_{L\boldsymbol{N}}^P) \\ (d_{U\mathbf{1}}^T, d_{L\mathbf{1}}^T) \\ \vdots \\ (d_{UN}^T, d_{L\boldsymbol{N}}^T) \end{bmatrix} = \begin{bmatrix} (x_{U\mathbf{1}}^P - y_{U\mathbf{1}}^P, x_{L\mathbf{1}}^P - y_{L\mathbf{1}}^P) \\ \vdots \\ (x_{UN}^P - y_{UN}^P, x_{LN}^P - y_{LN}^P) \\ (x_{U\mathbf{1}}^T - y_{U\mathbf{1}}^T, x_{L\mathbf{1}}^T - y_{L\mathbf{1}}^T) \\ \vdots \\ (x_{UN}^T - y_{UN}^T, x_{LN}^T - y_{LN}^T) \end{bmatrix}$$

where d_i^G is the pair of the difference in pain scores for both sections of the body for subject i in group G and d_{mi}^G is the change in pain score at body section m for subject i in group G.

Recall that the compound is considered successful for a subject if there is a reduction in pain for at least two different body sections. The subjects are once again stratified into a contingency table based on this success condition. The table that would be created is shown by Table 14.

Group \ Successful?	Yes	No
Placebo	a	b
Treatment	c	d

Table 14: Contingency Table for Testing Compound C

where

$$a = \sum_{i=1}^{N} \mathbb{1}(d_{mi}^{P} \neq 0 \; \forall m \in \{U, L\})$$

$$b = \sum_{i=1}^{N} \mathbb{1}(\exists m \in \{U, L\} \; s.t. \; d_{mi}^{P} \neq 0)$$

$$c = \sum_{i=1}^{N} \mathbb{1}(d_{mi}^{T} \neq 0 \; \forall m \in \{U, L\})$$

$$d = \sum_{i=1}^{N} \mathbb{1}(\exists m \in \{U, L\} \; s.t. d_{mi}^{T} \neq 0)$$

After this contingency table is created, we use Fisher's exact test to discover if there is a statistically significant difference between the placebo and Compound C according to the success condition. Once again, the most important part of simulating the data to perform the power calculation is generating $\mathbf{p}_{UL}^{T}$.

As noted earlier, the scientists expect that there may be some correlation between upper body and lower body pain scores. As such, they would like to estimate this correlation and factor it into the power calculations. As opposed to using typical correlation estimators such as Spearman's ρ (Spearman, 1904) or Kendall's τ (Kendall,

1938), we propose estimating the correlation using a partition of a bivariate normal distribution. We will name this correlation estimator ξ.

To motivate this approach, we will first alter the experiment so that upper body and lower body pain scores each have only two different possible pain scores instead of three. Therefore, there are only four possible combinations of outcomes: $\{1,1\},\{1,2\},\{2,1\}$, and $\{2,2\}$, and we can assume each patient follows the distribution:

$$x_i^G \sim Multi(1, \mathbf{p}_{UL}) \text{ for } i = 1,2,\dots,N;\ G \in \{P,T\}$$

where $\mathbf{p}_{UL} = (p_{11}, p_{12}, p_{21}, p_{22})'$ and $p_{jk} = \frac{n_{jk}}{N_0}$, the relative frequency of subjects achieving pain score j in the upper body and pain score k in the lower body for $j = 1,2;\ k = 1,2$.

We now define the estimation of the correlation between two ordinal categorical variables using a partition of a bivariate normal distribution in **ALGORITHM 8**. Figure 16 illustrates this estimating process.

ALGORITHM 8: Estimate ξ, the correlation between two ordinal categorical variables (X_1, X_2) which each have two levels, given initial probability parameter $\mathbf{p}_{X_1X_2} = (p_{11}, p_{12}, p_{21}, p_{22})'$.

1. Let the marginal probabilities for variable X_1 and X_2 be $\mathbf{p}_1 = (p_{1\cdot}, p_{2\cdot})'$ and $\mathbf{p}_2 = (p_{\cdot 1}, p_{\cdot 2})'$, respectively, where $p_{i\cdot} = \sum_{j=1}^{2} p_{ij}$ and $p_{\cdot j} = \sum_{i=1}^{2} p_{ij}$.
2. Divide the volume under the probability density function for the standard bivariate normal distribution into four portions by following the steps below:

2.1. Place q_{11}, a vertical line that divides the area into left and right portions, so that the left portion has $p_{1.}$ volume under the curve and the right portion has $p_{2.}$ volume under the curve. More specifically, q_{11} is a vertical line that intersects the x-axis at $(Q_{11}, 0)$ where Q_{11} satisfies $P(Z < Q_{11}) = p_{1.}$.

2.2. Place q_{21}, a horizontal line that splits the area into upper and lower portions, is placed so that the lower portion has $p_{.1}$ volume under the curve and the upper portion has $p_{.2}$ volume under the curve. More specifically, q_{21} is a horizontal line that intersects the y-axis at $(0, Q_{21})$ where Q_{21} satisfies $P(Z < Q_{21}) = p_{.1}$.

3. Finally, the correlation, ξ, is estimated for a bivariate normal distribution with mean vector $(0,0)'$ and marginal variances of 1 so that the squared difference of the volume under the curve and the true relative frequency for each section is minimized.

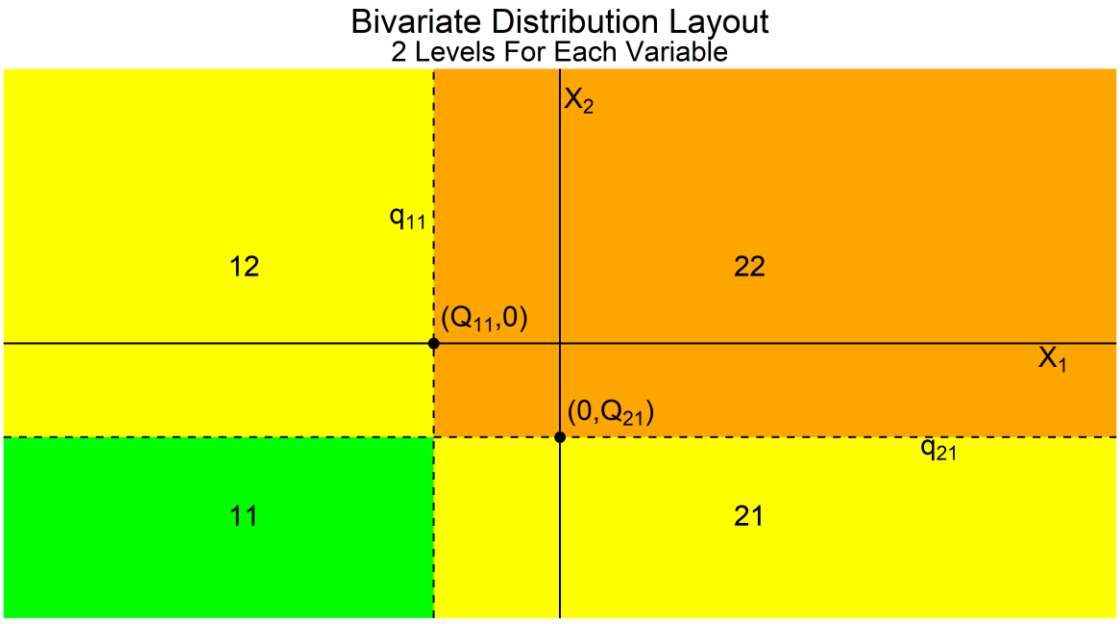

Figure 16: Correlation Estimation for 2 Ordinal Categorical Variables with 2 Levels

Referring to Figure 16, the volume under the curve that is above section k_1k_2 will correspond closely with the relative frequency observed in the initial experiment that each section represents after ξ is estimated.

Now let us return to the original example where each of the variables has three different levels in order to show how the estimation process is modified when pairs of variables have an arbitrary number of levels. This algorithm for performing this adjusted estimation process is given by **ALGORITHM 9**.

ALGORITHM 9: Calculate ξ, an estimation of the correlation between two ordinal categorical variables (X_1, X_2) which each have K_1 and K_2 levels, respectively, given initial probability parameter $\mathbf{p}_{X_1X_2} = \left(p_{11}, p_{12}, \dots, p_{(K_1-1)K_2}, p_{K_1K_2}\right)'$.

1. Let the marginal probabilities for variable X_1 and X_2 be $\mathbf{p}_1 = \left(p_{1\cdot}, p_{2\cdot}, \dots, p_{K_1\cdot}\right)'$ and $\mathbf{p}_2 = \left(p_{\cdot 1}, p_{\cdot 2}, \dots, p_{\cdot K_2}\right)'$, respectively, where $p_{i\cdot} = \sum_{j=1}^{K_1} p_{ij}$ and $p_{\cdot j} = \sum_{i=1}^{K_2} p_{ij}$.
2. Divide the volume under the probability density function for the standard bivariate normal distribution into K_1K_2 portions by following the steps below:
   2.1. Place vertical lines $q_{10}, q_{11}, q_{12}, \dots, q_{1K_1}$ which divide the area into K_1 rectangles extending infinitely with respect to the y-axis, so that the rectangle which lies between $q_{1(k-1)}$ and q_{1k} has $p_{k\cdot}$ volume under the curve. More specifically, q_{1k} is a vertical line that intersects the x-axis at $(Q_{1k}, 0)$ where Q_{1k} satisfies $P\left(Q_{1(k-1)} < Z < Q_{1k}\right) = p_{k\cdot}$. Note that $Q_{10} = -\infty$ and $Q_{1K_1} = \infty$.
   2.2. Place horizontal lines $q_{20}, q_{21}, q_{22}, \dots, q_{2K_2}$ which split the area into K_2 rectangles extending infinitely with respect to the x-axis, so that the rectangle which lies between $q_{2(k-1)}$ and q_{2k} has $p_{\cdot k}$ volume under the curve. More specifically, q_{2k}

is a vertical line that intersects the x-axis at $(Q_{2k}, 0)$ where Q_{2k} satisfies $P\left(Q_{2(k-1)}Z < Q_{2k}\right) = p_{\cdot k}$. Note that $Q_{20} = -\infty$ and $Q_{2K_2} = \infty$.

3. Finally, the correlation, ξ, is estimated for a bivariate normal distribution with mean vector $(0,0)'$ and marginal variances of 1 so that the squared difference of the volume under the curve and the true relative frequency for each section is minimized.

If any of the relative frequencies for a pair of scores are close to 0 this may not be effective. This is because it may not be possible to find a correlation which would create a bivariate normal distribution with the probabilities close to the relative frequencies for the corresponding portions. A skewed bivariate normal distribution may be more effective for cases such as these (Azzalini & Valle, 1996; Arslan, 2015). A bivariate t distribution could also be used (Kotz & Nadarajah, 2004).

Let m_1 and m_2 be two ordinal categorical variables. Once the correlation is estimated between m_1 and m_2 with **ALGORITHM 9**, we must generate $\mathbf{p}_{m_1 m_2}^T$. The algorithm for shifting the probability vector for a pair of correlated ordinal categorical variables is described in **ALGORITHM 10**. Like the previous shifting processes, shifts δ_1 and δ_2 move the dividers uniformly, which then changes the probabilities within each section of the bivariate normal distribution accordingly. The new probabilities are then used to form $\mathbf{p}_{m_1 m_2}^T$. This shifting process is illustrated in Figure 17.

ALGORITHM 10: Generate $\mathbf{p}_{m_1 m_2}^T$ given initial probability vector parameter $\mathbf{p}_{m_1 m_2}$ and shifts δ_{m_1} and δ_{m_2}.

1. Use **ALGORITHM 9** to divide the area under the probability density function into $K_1 * K_2$ sections, where K_1 and K_2 are the number of levels of variables m_1 and m_2, respectively.
2. For $l = 0,1, \dots, K_1$, shift vertical line q_{1l} with δ_{m_1} to create a new vertical line, q'_{1l}, which intersects the x-axis at $(Q_{1l} + \delta_1, 0)$. Recall $Q_{10} = -\infty$ and $Q_{1K_1} = \infty$.
3. For $l = 0,1, \dots, K_2$, shift horizontal line q_{2l} with δ_{m_2} to create a new horizontal line, q'_{2l}, which intersects the x-axis at $(Q_{2l} + \delta_2, 0)$. Recall $Q_{20} = -\infty$ and $Q_{2K_2} = \infty$.
4. The volume under the probability density function which lies within the vertical lines $q'_{1(k_1-1)}$ and q'_{1k_1} and the horizontal lines $q'_{2(k_2-1)}$ and q'_{2k_2} corresponds to $p^T_{k_1k_2}$, the hypothesized probability that a subject which has received treatment will achieve level k_1 for variable m_1 and k_2 for variable m_2. This is also the r^{th} element of $\mathbf{p}^T_{m_1m_2}$ where $r = K_1 * (k_1 - 1) + k_2$.

Once again, it should be noted that each individual quantile for each variable m_l, $Q_{lk} + \delta_l$ could also be shifted additionally by some γ_{lk}. In the simplest setting which we provide above, we set $\gamma_{lk} = 0$ for $k = 1, 2, \dots, K, l = 1, 2$.

The $p^T_{m_1m_2}$ generated by δ_{m_1} and δ_{m_2} is then used to simulate the data for the power calculations. The algorithm for calculating power for any combination of N, $p_{m_1m_2}$, $(\delta_{m_1}, \delta_{m_2})$, and success condition S is given in **ALGORITHM 11**.

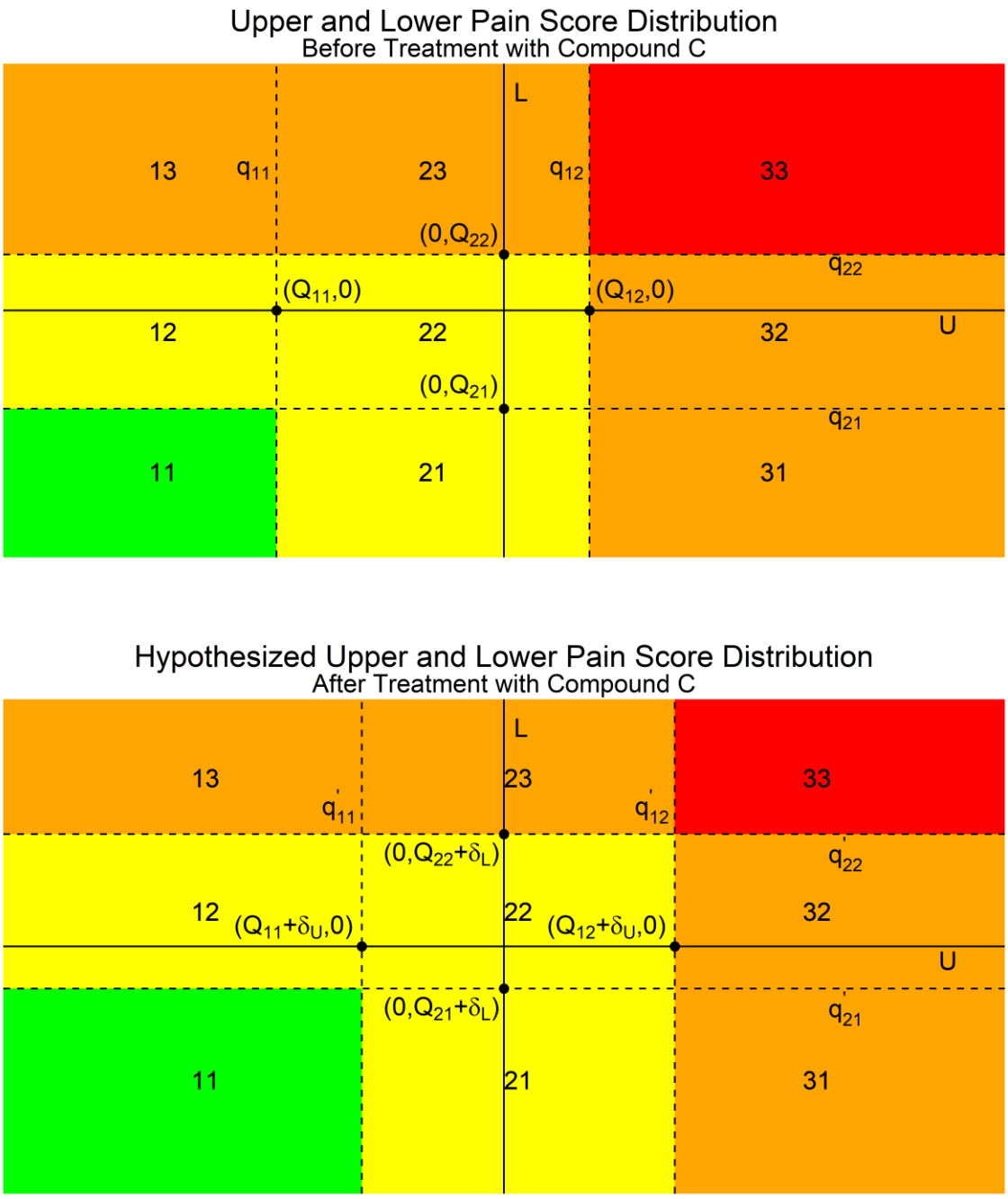

Figure 17: Shifting Process for a Pair of Correlated Ordinal Categorical Variables

ALGORITHM 11: Perform power calculations for two ordinal categorical variables given sample size N, initial probability vector parameter $p_{m_1 m_2}$, shifts $\left(\delta_{m_1}, \delta_{m_2}\right)$, success condition S, and number of iterations u.

1. Draw $2N$ random observations from $Multi\left(1, \boldsymbol{p_{m_1 m_2}}\right)$ to simulate $\boldsymbol{X} = (\boldsymbol{X^P}, \boldsymbol{X^T})'$.
2. Conduct **ALGORITHM 10** with $\left(\delta_{m_1}, \delta_{m_2}\right)$ and $\mathbf{p}_{m_1 m_2}$ to generate $\mathbf{p}^T_{m_1 m_2}$.
3. Draw N random observations from $Multi\left(1, \mathbf{p}_{m_1 m_2}\right)$ and draw N random observations from $Multi\left(1, \mathbf{p}^T_{m_1 m_2}\right)$ to simulate $\mathbf{Y}^{\mathrm{P}}$ and $\mathbf{Y}^T$, respectively. Then form $\mathbf{Y} = (\mathbf{Y}^P, \mathbf{Y}^T)'$.
4. Calculate $\mathbf{D} = \mathbf{X} - \mathbf{Y}$ and form a contingency table stratified by success condition S and treatment group.
5. Conduct Fisher's exact test.
6. Perform steps 1 through 5 for u iterations and the power for $\left(N, \mathbf{p}_{m_1 m_2}, \delta_{m_1}, \delta_{m_2}, S\right)$ is given by the proportion of iterations where Fisher's exact test rejects the null hypothesis.

Like **ALGORITHM 7**, as the number of possible levels for each variable rises so will the computational intensity. Before assuming any two pairs of variables are correlated, we suggest using the Chi-Square Test for Independence (Pearson, 1900). If the conclusion of this test is to reject the notion that the two variables are independent, then proceed with the method described in this section. Otherwise use the method described in Section 3.3. Section 4.3 details a simulation created to assess this bias of ξ and provides the results for a few cases.

3.4.1 Simulation to Assess the Bias of ξ

We used simulations to investigate the bias of ξ and compare it to the bias of Spearman's ρ and Kendall's τ. This was done by creating a partition of observations generated from a bivariate normal distribution with a “true” correlation estimate so that

each portion of the partition represents a pair of ordinal categorical variables. This simulation is given by Simulation 1.

SIMULATION 1: Given the true correlation, ρ_{true}, between two ordinal categorical variables which each have K_1 and K_2 levels, respectively, generated by a partition of a bivariate normal distribution, estimate the bias of ξ, Spearman's ρ, and Kendall's τ.

1. Choose $\Gamma_{11}, \ldots, \Gamma_{1K_1}$ and $\Gamma_{21}, \ldots, \Gamma_{2K_2}$ to be the values which create the vertical and horizontal lines, respectively, which will divide the flat area under the probability density function of a bivariate normal distribution into $K_1 * K_2$ portions and create the partition.
2. Draw N observations from a bivariate normal distribution with mean vector $(0,0)'$, marginal variances of 1, and correlation ρ_{true}.
3. Create a relative frequency table by counting how many of the N observations fall into each of the $K_1 * K_2$ sections created by $\Gamma_{11}, \ldots, \Gamma_{1K_1}$ and $\Gamma_{21}, \ldots, \Gamma_{2K_2}$ and dividing these counts by N.
4. Calculate ξ (using **ALGORITHM 9**), Spearman's ρ, and Kendall's τ for the relative frequency table created in step 3. Then calculate the bias of each: $b_1 = \rho_{\text{true}} - \xi$, $b_2 = \rho_{\text{true}} - \rho$, and $b_3 = \rho_{\text{true}} - \tau$.
5. Repeat steps 1 through 4 for 1000 iterations and the mean of the 1000 b_1's, b_2's, b_3's created in step 4 provides the estimated bias for ξ, Spearman's ρ, and Kendall's τ, respectively.

We chose three partition configurations for the bivariate normal distribution. Partition configuration 1 divides the probability distribution function into four portions with equal

probability when the correlation between the two variables is 0. Partition configuration 2 divides the probability distribution function into nine portions with equal probability when the correlation between the two variables is 0. Partition configuration 3 divides the probability distribution function into twelve portions as shown in the first plot in Figure 21 with probabilities matching the second relative frequency table in Table 15.

For each partition configuration we vary the true correlation and estimate the bias for four different sample sizes. Figure 18, Figure 19, and Figure 20 give the bias results for each estimator for each partition configuration.

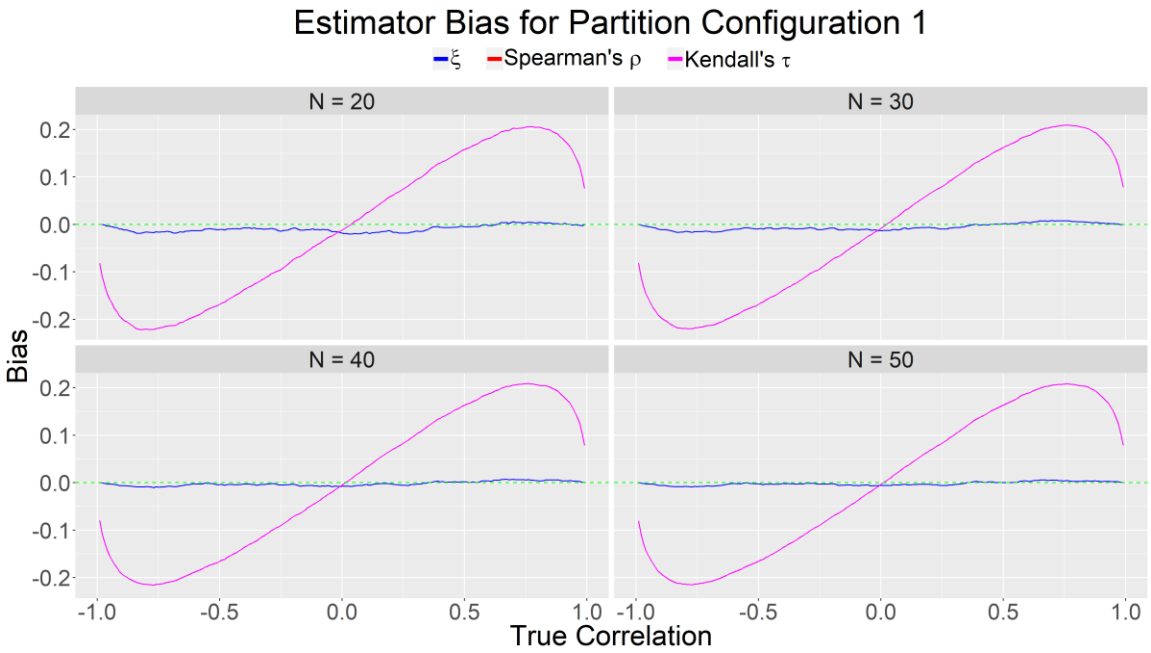

Figure 18: Bias Results for Partition Configuration 1

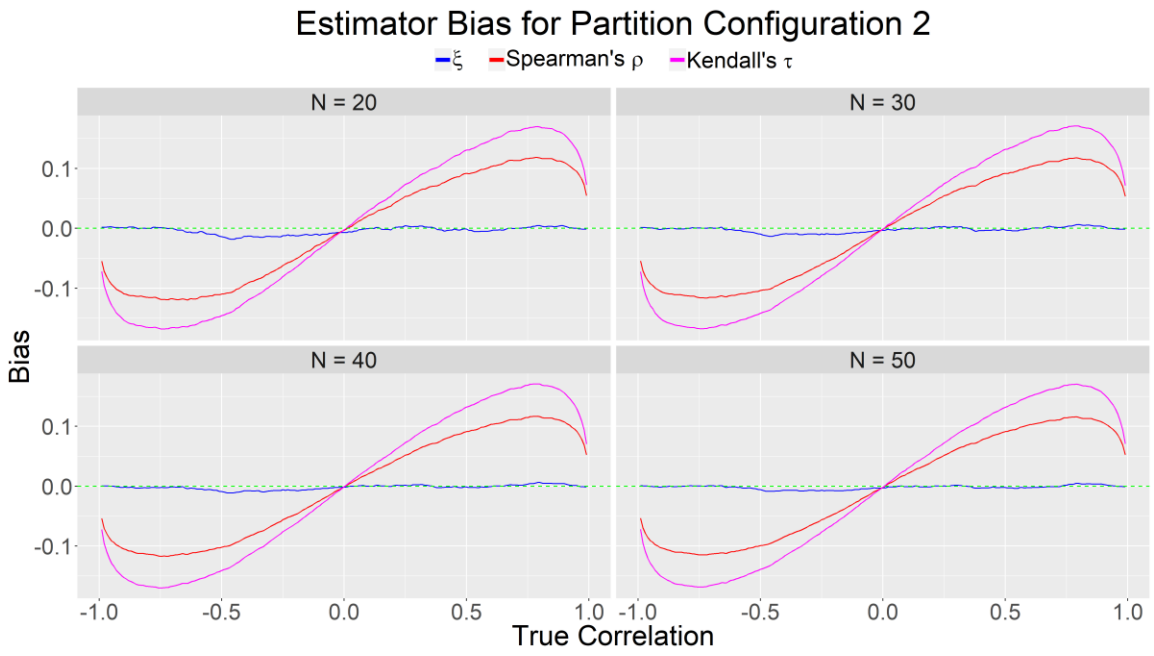

Figure 19: Bias Results for Partition Configuration 2

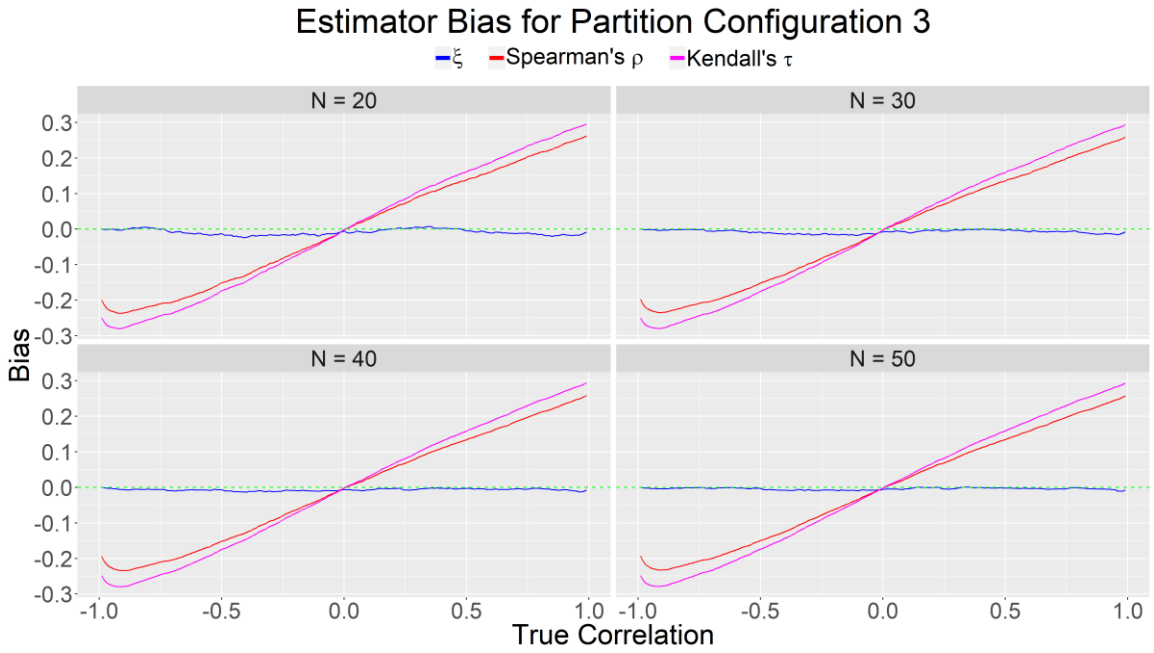

Figure 20: Bias Results for Partition Configuration 3

While there is certainly evidence of bias for ξ, when compared to the bias of both Spearman's ρ and Kendall's τ it appears minimal. Aside from when the correlation is near 0, ξ performes volumes better than the other two estimators in terms of bias.

3.5 Application to Preclinical Dataset

In this section we apply the method to a dataset provided to the author by a pharmaceutical company. In this experiment, a new compound is being designed to curb the effects of a disease. 55 subjects with this disease were assigned an "affliction" score which ranges from 0-8. This score is used to describe the amount of suffering the disease is causing for the subject and is composed of three subscores: subscore A which ranges from 0-2, subscore B which ranges from 0-3, and subscore C which ranges from 0-3. Table 15 gives the relative frequency table for the collected data.

Scores for Each Subscore

Score	A	B	C
0	0.04	0.03	0.01
1	0.27	0.19	0.79
2	0.69	0.47	0.18
3		0.31	0.02

Score Pairs for Subscores A and B

B \ A	0	1	2
3	0.01	0.12	0.19
2	0.01	0.14	0.32
1	0.01	0.01	0.17
0	0.01	0.01	0.01

Table 15: Relative Frequency Tables for Data from Initial Experiment

The objective of the compound is to decrease the affliction score for subjects and the compound is considered successful if the affliction score is reduced by at least two points and at least two subscores are reduced by at least one point. Let this be the success condition S. Before conducting the future experiment to decide whether to move forward with the compound, the scientists in charge of the experiment need to know how large of

a sample size is necessary to guarantee 90% power to detect the desired affliction score improvement.

After performing some preliminary analysis, some correlation between subscores A and B is discovered according to Spearman's ρ (-0.273) and Kendall's τ (-0.260). As such, **ALGORITHM 4** is used to shift the probability vector parameter for subscore C using a standard normal distribution and **ALGORITHM 10** is used to shift probability vector parameter for pairs of subscore A and subscore B using a bivariate normal distribution with correlation estimated by ξ (-0.302).

Let $\mathbf{p}_{AB} = (0.01,0.01,0.01,0.01,0.01,0.01,0.14,0.12,0.01,0.17,0.32,0.19)'$ and $\mathbf{p}_C = (0.01,0.79,0.18,0.02)'$. Note that the data has been adjusted to eliminate relative frequencies of 0. See Figure 21 and Figure 22 for a visual representation of how $\mathbf{p}_{AB}^T$ and $\mathbf{p}_C^T$, respectively, are generated.

Note that in the second graph of Figure 21, the volume under the curve over the area of section k_1k_2 corresponds to $p_{k_1k_2}^T$ in $\mathbf{p}_{AB}^T$. Further, in the second graph of Figure 22, the area under the curve of the k^{th} portion corresponds to p_k^T in $\boldsymbol{p}_C^T$.

An amalgamation of **ALGORITHM 5** and **ALGORITHM 11** is used to perform the power calculations by choosing combinations of N, $(\delta_A, \delta_B, \delta_C)$, and the success condition S. Figure 23 show the power curves for sample sizes (per treatment arm) 20 to 60 and various choices of $(\delta_A, \delta_B, \delta_C)$. The scientists believe that the compound will not have a large effect on subscore C so δ_C has been kept low for all shift configurations; however, δ_A and δ_B have been allowed to range between 1 and 1.5 because the scientists believe that the compound will have the most effect on subscores A and B.

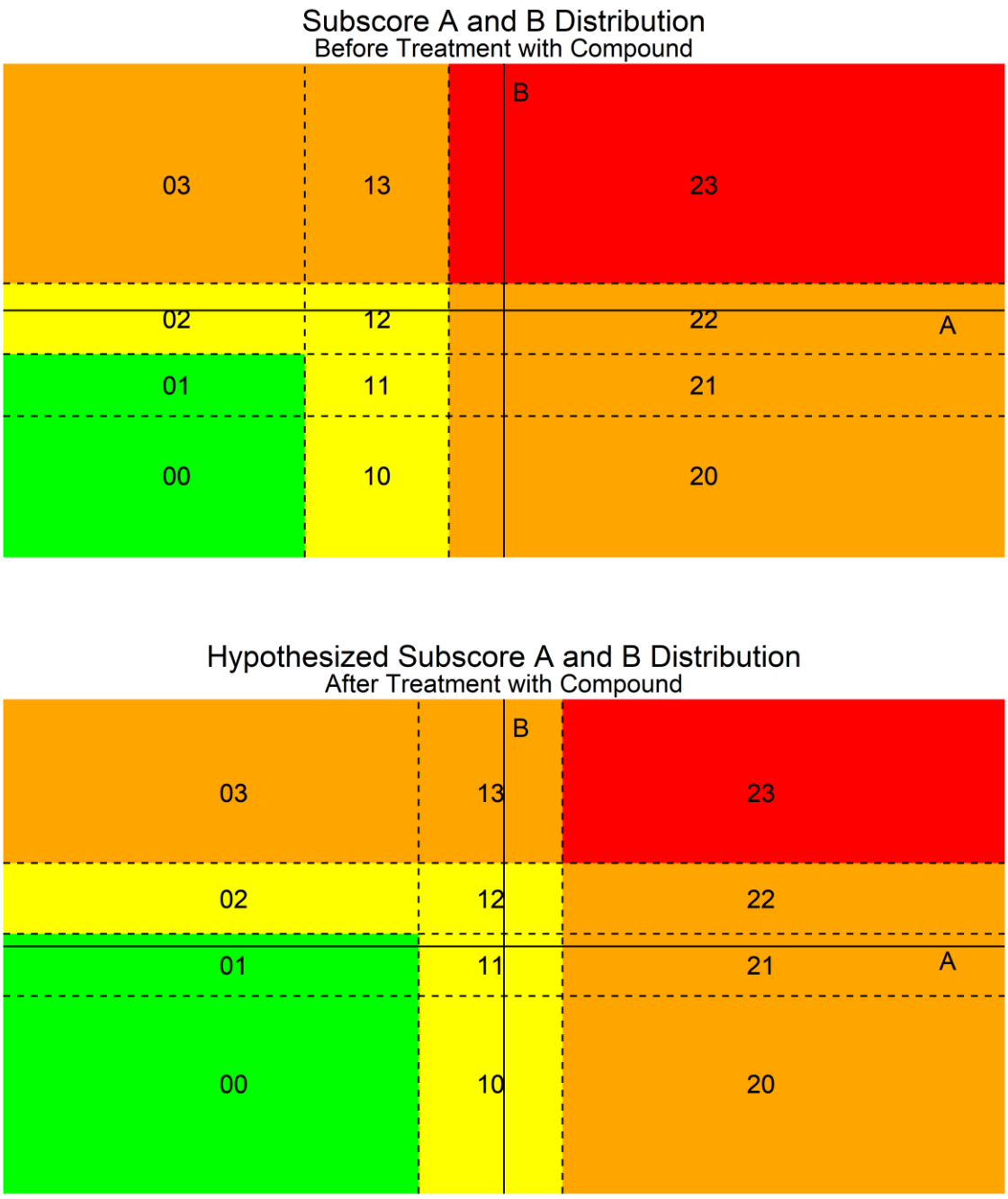

Figure 21:Generating $\boldsymbol{p}_{AB}^{T}$ *with* ***ALGORITHM 10***

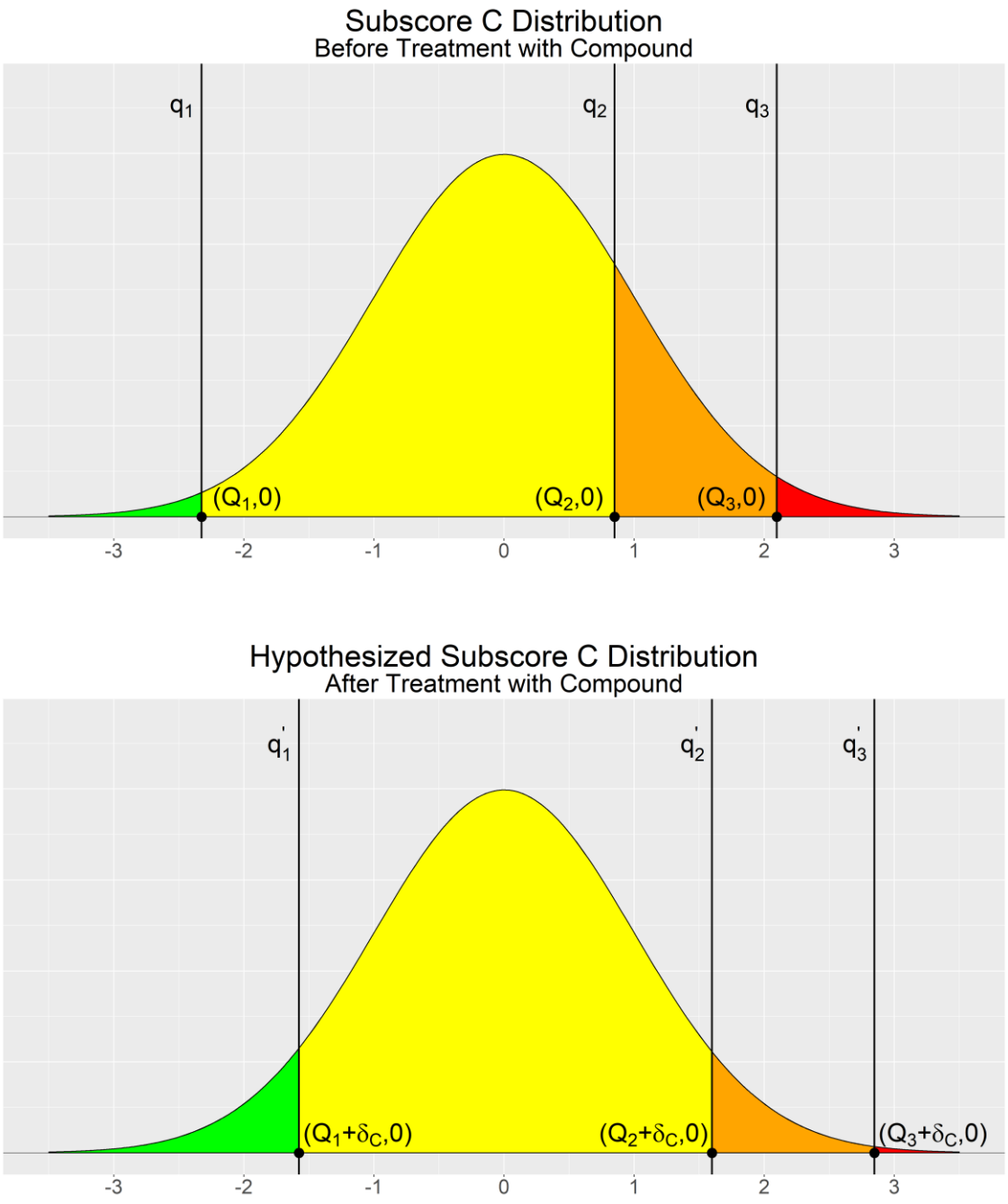

Figure 22: Generating $\boldsymbol{p}_C^T$ *with* ***ALGORITHM 1***

The power curves demonstrate that Fisher's exact test will have the desired power at some sample size for all nine configurations of $(\delta_A, \delta_B, \delta_C)$. For the alternative multinomial distribution corresponding to the least conservative configuration,

(1.5,1.5,0.5), a sample size of 23 per treatment arm would be needed to achieve at least 90% power; for the most conservative configuration (1,1,0.5), a sample size of 51 per treatment arm would be needed. Considering all configurations, a sample size of 30 to 40 subjects per treatment arm should be used to guarantee at least 90% power.

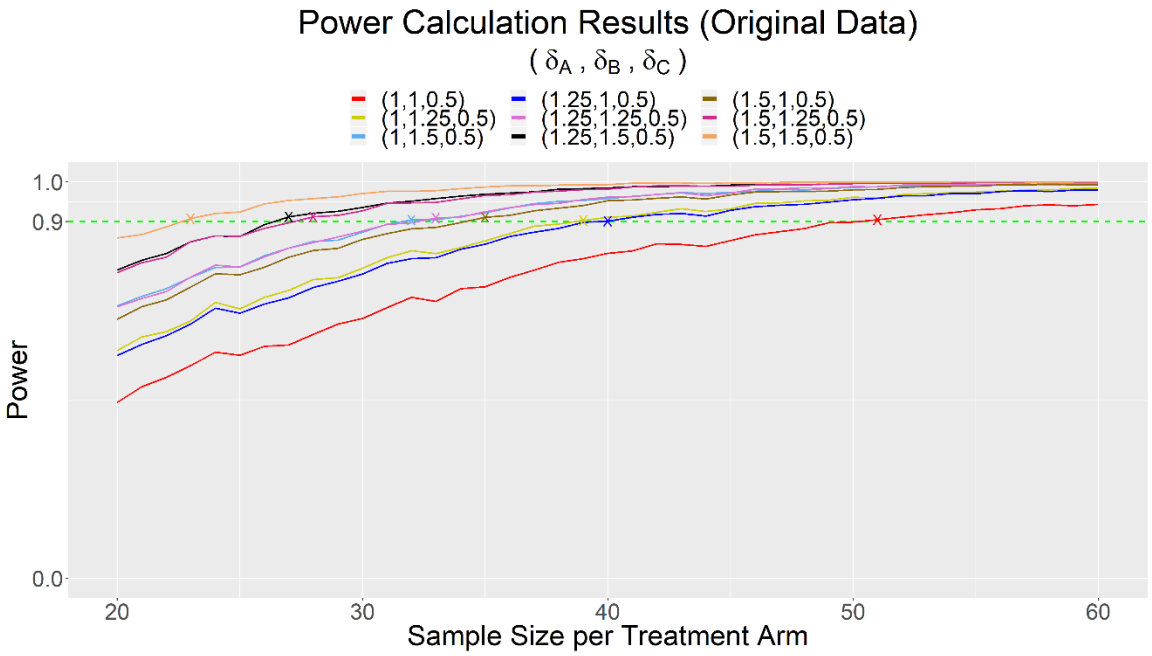

Figure 23: Power Curves for Original Data

The power curves demonstrate that Fisher's exact test will have the desired power at some sample size for all nine configurations of $(\delta_A, \delta_B, \delta_C)$. For the alternative multinomial distribution corresponding to the least conservative configuration, (1.5,1.5,0.5), a sample size of 23 per treatment arm would be needed to achieve at least 90% power; for the most conservative configuration (1,1,0.5), a sample size of 51 per treatment arm would be needed. Considering all configurations, a sample size of 30 to 40 subjects per treatment arm should be used to guarantee at least 90% power.

We also performed the power calculations again under less favorable conditions. In the original data, it was clear that many of the subjects were achieving very high scores prior to treatment. We adjusted the data to assess how many subjects would be needed if fewer subjects achieved very high scores in the initial sample. Once again, an amalgamation of **ALGORITHM 5** and **ALGORITHM 11** are used and Figure 24 show the power curves for sample sizes 20 to 60 and various choices of $(\delta_A, \delta_B, \delta_C)$.

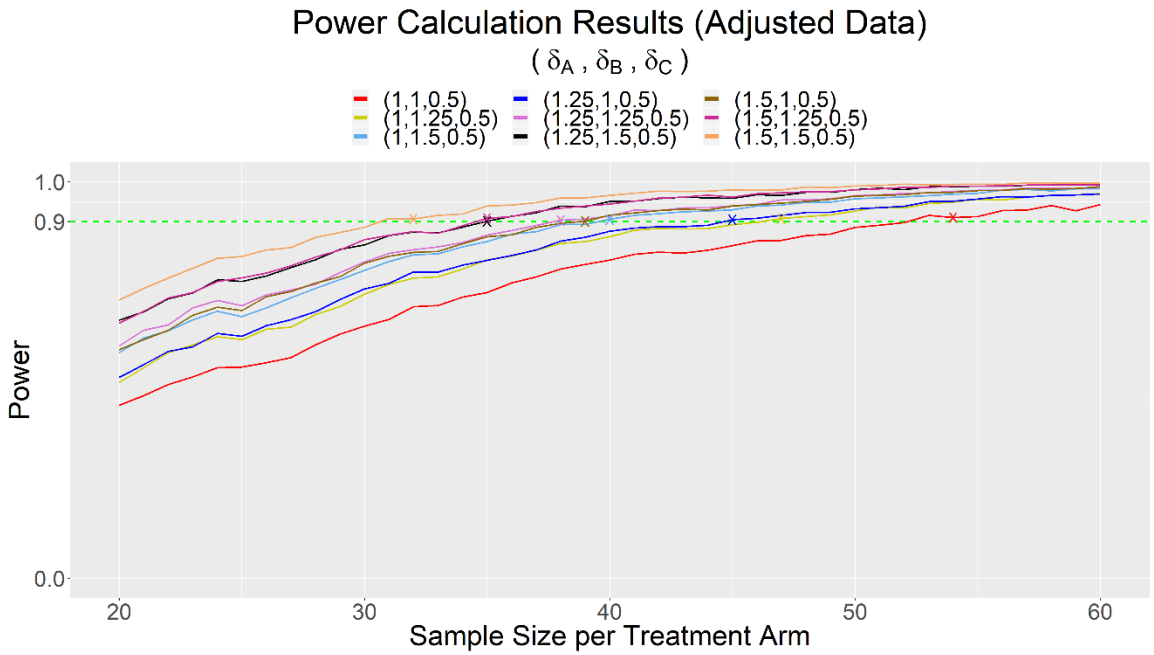

Figure 24: Power Curves for Adjusted Data

Once again, the power curves demonstrate that Fisher's exact test will have the desired power at some sample size for all nine configurations of $(\delta_A, \delta_B, \delta_C)$. For the alternative multinomial distribution corresponding to the least conservative configuration, $(1.5,1.5,0.5)$, a sample size of 32 per treatment arm would be needed to achieve at least 90% power; for the most conservative configuration $(1,1,0.5)$, a sample size of 54 per treatment arm would be needed.

Considering all configurations, a sample size of 35 to 45 subjects per treatment arm should be used to guarantee at least 90% power. Intuitively this makes sense, because if the subject's scores aren't initially very high it will be hard to reduce them by a considerable amount. As such, more subjects would be necessary to detect a treatment effect.

3.6 Discussion

We have described a method that can be used to avoid the overwhelming prospect of performing power calculations for hypothesis tests analyzing ordinal categorical data with small range scales. Instead of haphazardly manipulating the different entries in the probability vector parameter, we have proposed a method which shifts these entries in a uniform manner by creating and shifting the quantiles of a standard normal distribution for a single ordinal categorical variable or multiple uncorrelated ordinal categorical variables. We expanded it further for pairs of correlated ordinal categorical variables by applying the same concepts using a bivariate normal distribution.

We have also introduced ξ, a new estimator for estimating the correlation of ordinal categorical variables using a partition of a bivariate normal distribution. We assessed this estimator in terms of bias and compared it to other common estimators for measuring correlation among pairs of ordinal categorical variables. Finally, we applied the method to a dataset and delivered some results.

Note that this power calculation method could also be used for other hypothesis tests which analyze contingency tables such as the Chi-Square Test of Homogeneity. The R package *multinorm* has been developed to perform these power calculations with different hypothesis tests.

Future work can be done by possibly extending the methods described in Section 4 to work for three or more correlated categorical variables using a multivariate normal distribution. Work could also be done to determine the precise computational cost that arises from adding more variables and/or levels. Finally, it might be useful to see how this method can work for categorical variables with no discernable order, as well as the utility of the method when the scale range is increased.

Appendices

Appendix A: Selected R Code for Chapter 1

Section 2

```
# load packages
library(mvtnorm)
library(ggplot2)

# create data of random observations
sim.data = rmvnorm(5000,
                   mean = rep(0,5),
                   sigma = diag(5:1))

# create cluster configuration for Kmeans
K.cl = kmeans(sim.data,
                centers = 3,
                iter.max = 10,
                nstart = 5)


# retrieve principal components
PC.data = as.data.frame(princomp(sim.data)$scores)

# duplicate principal components
PC.data = rbind.data.frame(PC.data,
                           PC.data)

# retrieve cluster configurations
PC.data[, "Cluster"] = factor(c(rep(0,nrow(PC.data)/2),
                                K.cl$cluster))

# split the data into two sections
PC.data[, "Type"] = factor(c(rep("Original Data",nrow(PC.data)/2),
                             rep("Clustered Data",nrow(PC.data)/2)),
                           levels = c("Original Data",
                                      "Clustered Data"))

# create plot of kmeans clustering configurations
ggplot(data = PC.data,
       aes(x = Comp.1,
           y = Comp.2,
           color = Cluster)) +
  facet_wrap(~Type) +
  geom_point() +
  scale_color_manual(values = c("black",
                                "red",
                                "blue",
                                "limegreen")) +
  scale_x_continuous("") +
  scale_y_continuous("") +
  guides(color = FALSE) +
  theme_gray(base_size = 18) +
  theme(plot.title = element_text(hjust = 0.5),
        element_blank(),
        axis.ticks.x = element_blank(),
        axis.text.x = element_blank(),
        axis.ticks.y = element_blank(),
        axis.text.y = element_blank())

# create distance matrix for
sim.dist = dist(sim.data)

# create three clusters with hierarchical clustering
H.cl = hclust(sim.dist)
clusterCut = cutree(H.cl, 3)

# retrieve cluster configurations
PC.data[, "Cluster"] = factor(c(rep(0,nrow(PC.data)/2),
                                clusterCut))

# create plot for hierarchical clustering
ggplot(data = PC.data,
```

```
      aes(x = Comp.1,
          y = Comp.2,
          color = Cluster)) +
  facet_wrap(~Type) +
  geom_point() +
  scale_color_manual(values = c("black",
                                "red",
                                "blue",
                                "limegreen")) +
  scale_x_continuous("") +
  scale_y_continuous("") +
  guides(color = FALSE) +
  theme_gray(base_size = 18) +
  theme(plot.title = element_text(hjust = 0.5),
        element_blank(),
        axis.ticks.x = element_blank(),
        axis.text.x = element_blank(),
        axis.ticks.y = element_blank(),
        axis.text.y = element_blank())
```

Section 3

```
# load packages
library(ggplot2)
library(parallel)
library(doSNOW)
library(support)

# load functions
source("datanugget Package/create DN.R")
source("datanugget Package/create DN2.R")
source("datanugget Package/refine DN.R")
source("datanugget Package/WKmeans.R")
source("functions/density plot.R")

# set seed for replication
set.seed(103092)

# create x for noise data
x = rnorm(15000)

# create y for noise data
y = rnorm(15000)

# create noise data
noise.data = cbind(x,y)

# create smile data
smile.data = cbind(seq(-1.5,1.5,0.005),
                   .25*(seq(-1.5,1.5,0.005))^2-0.5)

# give column names to the data
colnames(noise.data) = c("X","Y")
colnames(smile.data) = c("X","Y")

# create entire dataset
original.data = rbind.data.frame(noise.data,
                                 smile.data)

# retrieve random sample of 2000 observations
random.sample.data = original.data[sample(1:nrow(original.data),2000), ]

# generate data nuggets
for.DN = create.DN(x = original.data,
                   RS.num = 10000,
                   DN.num = 2000)

# retrieve data nuggets
DN.information = for.DN$`Data Nuggets`

# create matrix for original data nuggets density plot
DN.z = matrix(0,
              nrow = 100,
              ncol = 100)

DN.zx = seq(length = 101,
            from = min(DN.information[, 2]),
            to = max(DN.information[, 2]))
```

```
DN.zy = seq(length = 101,
            from = min(DN.information[, 3]),
            to = max(DN.information[, 3]))

for(i in 1:100){

  for(j in 1:100){

    DN.z[i,j] = sum(DN.information[DN.information[, 2] > DN.zx[i] &
                                   DN.information[, 2] < DN.zx[i+1] &
                                   DN.information[, 3] > DN.zy[j] &
                                   DN.information[, 3] < DN.zy[j+1], "Weight"])


  }

}

# create top row of Figure 3 plots
par(mfrow = c(1,2),
    mar = c(0,0,0,0) + 0.5)

plot(original.data[,1],
     original.data[,2],
     # ylim = c(min(original.data[, 2]) + 1,
     #          max(original.data[, 2]) + 1),
     xaxt = "n",
     yaxt = "n",
     xlab = "",
     ylab = "")

density.plot(original.data[,1],
             original.data[,2],
             100,
             100)

# create bottom row of Figure 3 plots
density.plot(random.sample.data[, 1],
             random.sample.data[, 2],
             100,
             100)

image(z = DN.z,
      x = DN.zx,
      y = DN.zy,
      xlab = "",
      ylab = "",
      col = topo.colors(100),
      xaxt = "n",
      yaxt = "n")

# create refined data nuggets
for.DN2 = refine.DN(x = original.data,
                    DN = for.DN,
                    scale.tol = 1,
                    min.nugget.size = 2)

# retrieve refined data nuggets
DN.information2 = for.DN2$`Data Nuggets`

# create matrix for refined data nuggets density plot
DN2.z = matrix(0,
               nrow = 100,
               ncol = 100)

DN2.zx = seq(length = 101,
             from = min(DN.information2[, 2]),
             to = max(DN.information2[, 2]))

DN2.zy = seq(length = 101,
             from = min(DN.information2[, 3]),
             to = max(DN.information2[, 3]))

for(i in 1:100){

  for(j in 1:100){

    DN2.z[i,j] = sum(DN.information2[DN.information2[, 2] > DN2.zx[i] &
                                       DN.information2[, 2] < DN2.zx[i+1] &
                                       DN.information2[, 3] > DN2.zy[j] &
```

```
                                        DN.information2[, 3] < DN2.zy[j+1], "Weight"])

  }

}

# create top row of Figure 4 plots
plot(DN.information[,2],
     DN.information[,3],
     # ylim = c(min(original.data[, 2]) + 1,
     #          max(original.data[, 2]) + 1),
     #col = plot.colors,
     #pch = plot.shapes,
     xaxt = "n",
     yaxt = "n",
     xlab = "",
     ylab = "")

image(z = DN.z,
      x = DN.z x,
      y = DN.z y,
      xlab = "",
      ylab = "",
      col = topo.colors(100),
      xaxt = "n",
      yaxt = "n")

# create bottom row of Figure 4 Plots
plot(DN.information2[,2],
     DN.information2[,3],
     # ylim = c(min(original.data[, 2]) + 1,
     #          max(original.data[, 2]) + 1),
     xaxt = "n",
     yaxt = "n",
     xlab = "",
     ylab = "")

image(z = DN.z,
      x = DN.zx,
      y = DN.zy,
      xlab = "",
      ylab = "",
      col = topo.colors(100),
      xaxt = "n",
      yaxt = "n")

# binary data nuggets simulation ####

# set the number of iterations
iteration.num = 100

# retrieve the number of cores
no.cores = detectCores() - 1

# create list of probabilities
probs = seq(.8,.9,0.02)

# cycle through the list of probabilities
for (m in probs){

  # create the cluster for parallel processing
  cl = makeCluster(no.cores)

  # engage the cluster for parallel processing
  registerDoSNOW(cl)

  system.time({

    # initialize progress bar
    pb = txtProgressBar(min = 0, max = iteration.num)

    # update the progress bar
    progress = function(n){setTxtProgressBar(pb, n)}
    opts = list(progress = progress)

    # initalize probabilities for layers of data
    p1 = m
    p2 = 1-p1

    # create sample size
```

```
n = 100000

# perform simulation
results = foreach(i = 1:iteration.num,
                  .combine = rbind,
                  .options.snow = opts)  %dopar%

                  {

                    # set seed for replication
                    set.seed(i)

                    # create left layer of 1st cluster of data (5 zeros)
                    m1 = array(sample(0:1,
                                      n*5,
                                      prob = c(p1,1-p1),
                                      rep = T),
                               dim = c(n,5))

                    # create right layer of 1st cluster of data (5 ones)
                    m2 = array(sample(0:1,
                                      n*5,
                                      prob = c(p2,1-p2),
                                      rep = T),
                               dim = c(n,5))

                    # create left layer of 2nd cluster of data (5 ones)
                    m3 = array(sample(0:1,
                                      n*5,
                                      prob = c(p2,1-p2),
                                      rep = T),
                               dim = c(n,5))

                    # create right layer of 2nd cluster of data (5 zeros)
                    m4 = array(sample(0:1,
                                      n*5,
                                      prob = c(p1,1-p1),
                                      rep = T),
                               dim = c(n,5))


                    # create 3rd cluster of data (10 ones)
                    m5 = array(sample(0:1,
                                      n*10,
                                      prob = c(p2,1-p2),
                                      rep = T),
                               dim = c(n,10))

                    # create simulated data
                    sim.data = as.data.frame(rbind(cbind(m1,m2),
                                                   cbind(m3,m4),
                                                   m5))

                    # assign the true clusters
                    sim.data[, "Cluster"] = rep(1:3,
                                                each = 100000)

                    # convert the data to a data frame
                    sim.data = as.data.frame(sim.data)

                    # find the unique rows of the data
                    unique.rows = unique(sim.data[, 1:10])

                    # initialize the data nugget data
                    DN.info = data.frame(Data.Nugget = 1:nrow(unique.rows))

                    # make the data nuggets the unique rows
                    DN.info[, 2:11] = as.matrix(unique.rows)

                    # give the data nuggets colum names
                    colnames(DN.info) = c("Data Nugget",
                                          paste("Center",
                                                1:10,
                                                sep = ""))

                    # assign data nuggets to the original data
                    DN.assignments = apply(X = as.matrix(sim.data[, 1:10]),
                                           MARGIN = 1,
                                           FUN = function(input){
```

```
return(which(colSums(t(as.matrix(DN.info[, 2:11])) == input) == 10))

                                        })

                        # find the weights of the data nuggets
                        DN.info[, "Weight"] = table(DN.assignments)/nrow(sim.data)

                        # find the k means cluster configuration
                        Kmeans.results = kmeans(x = DN.info[, 2:11],
                                                centers = 3,
                                                nstart = 5,
                                                iter.max = 10)

                        # find the weighted k means cluster configuration
                        WKmeans.results = WKmeans(dataset = DN.info[, 2:11],
                                                  k = 3,
                                                  obs.weights = DN.info[, "Weight"],
                                                  num.init = 5,
                                                  K = 10)

                        # create a matrix of possible permutations clusters
                        perm.matrix = rbind(c(1,2,3),
                                            c(1,3,2),
                                            c(2,1,3),
                                            c(2,3,1),
                                            c(3,1,2),
                                            c(3,2,1))

                        # initialize the vector holding the correct classifications for
each permutation
                        correct.Kmeans = rep(0,6)
                        correct.WKmeans = rep(0,6)

                        # cycle through the permutations
                        for (j in 1:6){

                          # reassign the k means clusters according to the current
permutation
                          new.Kmeans.cluster = Kmeans.results$cluster + 3
                          new.Kmeans.cluster[new.Kmeans.cluster == 4] = perm.matrix[j,
][1]
                          new.Kmeans.cluster[new.Kmeans.cluster == 5] = perm.matrix[j,
][2]
                          new.Kmeans.cluster[new.Kmeans.cluster == 6] = perm.matrix[j,
][3]

                          # reassign the weighted k means clusters according to the
current permutation
                          new.WKmeans.cluster = WKmeans.results$`Cluster Assignments` + 3
                          new.WKmeans.cluster[new.WKmeans.cluster == 4] = perm.matrix[j,
][1]
                          new.WKmeans.cluster[new.WKmeans.cluster == 5] = perm.matrix[j,
][2]
                          new.WKmeans.cluster[new.WKmeans.cluster == 6] = perm.matrix[j,
][3]

                          # append the current cluster assignments for each method to the
data nuggets
                          DN.info[, "Kmeans.Cluster"] = new.Kmeans.cluster
                          DN.info[, "WKmeans.Cluster"] = new.WKmeans.cluster

                          # append each observation of the original data with its data
nugget assignment
                          sim.data[, "DN Assignment"] = DN.assignments

                          # append the current cluster assignments to each observation
                          # according to their assigned data nugget
                          sim.data[, "DN.Kmeans.Cluster"] = DN.info[DN.assignments,
"Kmeans.Cluster"]
                          sim.data[, "DN.WKmeans.Cluster"] = DN.info[DN.assignments,
"WKmeans.Cluster"]

                          # find the correct classifications for the k means method
                          correct.Kmeans[j] = mean(sim.data[, "Cluster"] ==
                                                   sim.data[, "DN.Kmeans.Cluster"])

                          # find the correct classifications for the weighted k means
method
                          correct.WKmeans[j] = mean(sim.data[, "Cluster"] ==
                                                    sim.data[, "DN.WKmeans.Cluster"])
```

```
                        }

                        # retrieve the best percentage of correct classification for each
method
                        best.correct.Kmeans = max(correct.Kmeans)
                        best.correct.WKmeans = max(correct.WKmeans)

                        # create the results for this iteration
                        iteration.result = c(i,
                                             best.correct.Kmeans,
                                             best.correct.WKmeans)

                        # return the results for this iterations
                        return(iteration.result)

                      }

    # stop the cluster
    stopCluster(cl)

  })

  # find the average percentage of correct classification for each method
  results2 = colMeans(results)[2:3]

  # find the average percentage of correct classification for each method for this
probability
  results2 = colMeans(results)[2:5]

  # print the results for this probability
  print(results2)

# quantile bias simulation ####

# set the number of iterations
num.iterations = 1000

# initialize the vector of estimated quantiles for each method
est.q.DN = NULL
est.q.SP = NULL

# cycle through the iterations
for (i in 1:num.iterations){

  # set seed for reproducibility
  set.seed(i)

  # sample 100000 observations from a random distribution
  x = rnorm(100000)

  # generate the support points
  SP = sp(100,
          1,
          ini = as.matrix(x))$sp

  # generate data nuggets
  for.DN = create.DN2(x = x,
                      RS.num = 1000,
                      DN.num = 100)

  # extract data nuggets
  DN.info = for.DN$`Data Nuggets`

  # order data nuggets
  DN.info = DN.info[order(DN.info[, "Center1"]), ]

  # extract data nugget centers and weights
  DN.centers = DN.info[, "Center1"]
  DN.weights = DN.info[, "Weight"]/length(x)

  # create true probabilities for quantile values
  true.p = seq(.95,.99,.01)

  # calculate the estimated quantiles according to the support point method
  est.q.SP = cbind(est.q.SP,
                   quantile(SP,
                            true.p))

  # calculate the estimated quantiles according to the data nuggets method
  est.q.DN = cbind(est.q.DN,
```

```
                    approx(cumsum(DN.weights),
                           DN.centers,
                           true.p)$y)

}

# find the bias for each iteration/percentile combination
SP.Bias = t(est.q.SP-qnorm(true.p))
DN.Bias = t(est.q.DN-qnorm(true.p))

# initialize the the vector that will store the bias results
SP.Bias.vec = SP.Bias[, 1]
DN.Bias.vec = DN.Bias[, 1]

# cycle through the percentiles
for (i in 95:99){

  SP.Bias.vec = c(SP.Bias.vec,
                  SP.Bias[, i])

  DN.Bias.vec = c(DN.Bias.vec,
                  DN.Bias[, i])

}

# form the boxplot data
BP.data = data.frame(Percentile = c(rep(paste(c(95:99),
                                              "%",
                                              sep = ""),
                                        each = num.iterations),
                                    rep(paste(c(95:99),
                                              "%",
                                              sep = ""),
                                        each = num.iterations)),
                     Method = c(rep("Support Points", 5*num.iterations),
                                rep("Data Nuggets", 5*num.iterations)),
                     Bias = c(SP.Bias.vec,
                              DN.Bias.vec))

# factor the method variable
BP.data[, "Method"] = factor(BP.data[, "Method"],
                             levels = unique(BP.data[, "Method"]))

# create the boxplot
ggplot(BP.data,
       aes(x = Percentile,
           y = Bias)) +
  facet_wrap(~Method) +
  geom_boxplot() +
  scale_y_continuous("Quantile Estimate Bias") +
  geom_hline(yintercept = 0,
             col = "green",
             linetype = "dashed") +
  ggtitle("Quantile Estimate Bias for Upper 5% of Normal Distribution") +
  theme_gray(base_size = 18) +
  theme(plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5))
```

Appendix B: Selected R Code for Chapter 2

Section 3

```
# load the NMA Function
source("NMA Package/NMA Function.R")

# create the Moore-Penrose Inverse Function
MP.inv = function(X){

  return(solve(X - matrix(1,ncol(X),ncol(X))/ncol(X)) +
           matrix(1,ncol(X),ncol(X))/ncol(X))

}

# Code for dataset A ####

# load Dataset A
Dataset.A = read.csv("data/Dataset A.csv")


# fit GLS method (baseline Trt. C)
NMA.fit = NMA.function(contrast = FALSE,
                       dataset = Dataset.B,
                       check.netmeta = FALSE,
                       method = "Regression",
                       baseline.choices = c(1,3))


# retrieve the contrast summary measures
contrast.data = NMA.fit$`Contrast Data`

# Code for GLS checkpoints for Dataset A ####

# Calculate numbers for design AB ####

# retrieve sample mean differences, standard errors, and weights
d.AB.1 = contrast.data[1,1]
s.AB.1 = contrast.data[1,2]
W.AB.1 = 1/(s.AB.1)^2
d.AB.2 = contrast.data[2,1]
s.AB.2 = contrast.data[2,2]
W.AB.2 = 1/(s.AB.2)^2

# calculate effect size estimate and standard error squared
d.hat.AB = (d.AB.1*W.AB.1 + d.AB.2*W.AB.2)/(W.AB.1 + W.AB.2)
s2.d.hat.AB.1 = 1/(W.AB.1 + W.AB.2)

# calculate effect size estimate and standard error squared
d.hat.AB = (d.AB.1*W.AB.1 + d.AB.2*W.AB.2)/(W.AB.1 + W.AB.2)
s2.d.hat.AB.1 = 1/(W.AB.1 + W.AB.2)

# Calculate numbers for design ABCD ####

# create sample mean difference vectors, covariance matrices, and weight matrices
(baseline Trt. C)
d.ABCD.1 = c(contrast.data[4,1],
             contrast.data[6,1],
             contrast.data[8,1])

V.ABCD.1 = matrix(c(contrast.data[4,2],Dataset.B[7,2],Dataset.B[7,2],
                    Dataset.B[7,2],contrast.data[6,2],Dataset.B[7,2],
                    Dataset.B[7,2],Dataset.B[7,2],contrast.data[8,2])^2,
                  nrow = 3,
                  byrow = TRUE)

V.ABCD.1[3,2] = -V.ABCD.1[3,2]
V.ABCD.1[2,3] = -V.ABCD.1[2,3]
V.ABCD.1[3,1] = -V.ABCD.1[3,1]
V.ABCD.1[1,3] = -V.ABCD.1[1,3]

W.ABCD.1 = solve(V.ABCD.1)

# Calculate the effect size estimate vector, covariance matrix, and weight matrix ####
W.ABCD = W.ABCD.1
d.hat.ABCD = solve(W.ABCD)%*%(W.ABCD.1%*%d.ABCD.1)
V.ABCD = solve(W.ABCD)
```

```
# combine all effect size estimates into one vector
y.gls = c(d.hat.AB,
          d.hat.ABCD)

# create variance and weight matrix for initial estimates
V.gls = matrix(0,
               nrow = 4,
               ncol = 4)

V.gls[1,1] = s2.d.hat.AB.1
V.gls[2:4,2:4] = V.ABCD

W.gls = solve(V.gls)

# Calculate the GLS fixed effects estimates ####

# create H matrix
H = matrix(c(1,0,0,
             0,1,0,
             -1,1,0,
             0,0,1),
           byrow = TRUE,
           nrow = 4)

# create X matrix
X = H

# calculate the GLS effect size estimates and the associated covariance matrix
d.hat.gls = H%*%solve(t(X)%*%W.gls%*%X)%*%t(X)%*%W.gls%*%y.gls

# Code for ENT checkpoints for Dataset A ####

# retrieve the sample mean differences from the contrast data
y.ent = contrast.data[, 1]

# create the sub edge-vertex incidence matrix for design AB
B.AB = c(1,-1,0,0)

# create the sub edge-vertex incidence matrix for design ABCD
B.ABCD = matrix(c(1,-1,0,0,
                  1,0,-1,0,
                  1,0,-0,-1,
                  0,1,-1,0,
                  0,1,0,-1,
                  0,0,1,-1),
                nrow = 6,
                byrow = TRUE)

# create the edge-vertex incidence matrix
B = rbind(B.AB,
          B.AB,
          B.ABCD)

# create the sub variance matrices for experiments with design ABCD
V1 = matrix(c(0,contrast.data[3,2],contrast.data[4,2],contrast.data[5,2],
              contrast.data[3,2],0,contrast.data[6,2],contrast.data[7,2],
              contrast.data[4,2],contrast.data[6,2],0,contrast.data[8,2],
              contrast.data[5,2],contrast.data[7,2],contrast.data[8,2],0)^2,
            byrow = TRUE,
            nrow = 4)

# create the Lplus matrices for experiments with design ABCD
L1.plus = (-1/(2*(4^2)))*t(B.ABCD)%*%B.ABCD%*%V1%*%t(B.ABCD)%*%B.ABCD

# create the L matrices for experiments with design ABCD
L1 = MP.inv(L1.plus)

# retrieve the weights for experiments with design ABCD
W1 = -c(L1[1,2],L1[1,3],L1[1,4],L1[2,3],L1[2,4],L1[3,4])

# create the weight matrix
W.ent = diag(c(1/(contrast.data[1:2,2])^2,W1))

# create L matrix
L = t(B)%*%W.ent%*%B

# create Lplus matrix
L.plus = MP.inv(L)
```

```
# create effect size estimate vector
d.hat.ent = B%*%L.plus%*%t(B)%*%W.ent%*%y.ent

# Code for dataset B ####

# load Dataset B
Dataset.B = read.csv("data/Dataset B.csv")

# fit GLS method (baseline Trt.A)
NMA.fit = NMA.function(contrast = FALSE,
                       dataset = Dataset.A,
                       check.netmeta = FALSE,
                       method = "Regression")

contrast.data = NMA.fit$`Contrast Data`

# Code for GLS checkpoints for Dataset A ####

# Calculate numbers for design AB ####

# retrieve sample mean differences, standard errors, and weights
d.AB.1 = contrast.data[8,1]
s.AB.1 = contrast.data[8,2]
W.AB.1 = 1/(s.AB.1)^2

# calculate effect size estimate and standard error squared
d.hat.AB = d.AB.1*W.AB.1/W.AB.1
s2.d.hat.AB.1 = 1/W.AB.1

# Calculate numbers for design BC ####

# retrieve sample mean differences, standard errors, and weights
d.BC.1 = contrast.data[7,1]
s.BC.1 = contrast.data[7,2]
W.BC.1 = 1/(s.BC.1)^2
d.BC.2 = contrast.data[9,1]
s.BC.2 = contrast.data[9,2]
W.BC.2 = 1/(s.BC.2)^2

# calculate effect size estimate and standard error squared
d.hat.BC = (d.BC.1*W.BC.1 + d.BC.2*W.BC.2)/(W.BC.1 + W.BC.2)
s2.d.hat.BC.1 = 1/(W.BC.1 + W.BC.2)

# Calculate numbers for design ABC ####

# create sample mean difference vectors, covariance matrices, and weight matrices
(baseline Trt. A)
d.ABC.1 = c(contrast.data[1,1], contrast.data[2,1])

V.ABC.1 = matrix(c(contrast.data[1,2],Dataset.A[1,2],
                   Dataset.A[1,2],contrast.data[2,2])^2,
                 nrow = 2,
                 byrow = TRUE)

W.ABC.1 = solve(V.ABC.1)

d.ABC.2 = c(contrast.data[4,1], contrast.data[5,1])

V.ABC.2 = matrix(c(contrast.data[4,2],Dataset.A[4,2],
                   Dataset.A[4,2],contrast.data[5,2])^2,
                 nrow = 2,
                 byrow = TRUE)

W.ABC.2 = solve(V.ABC.2)

d.ABC.3 = c(contrast.data[10,1], contrast.data[11,1])

V.ABC.3 = matrix(c(contrast.data[10,2],Dataset.A[13,2],
                   Dataset.A[13,2],contrast.data[11,2])^2,
                 nrow = 2,
                 byrow = TRUE)

W.ABC.3 = solve(V.ABC.3)

# Calculate the effect size estimate vector, covariance matrix, and weight matrix ####
W.ABC = W.ABC.1 + W.ABC.2 + W.ABC.3
d.hat.ABC = solve(W.ABC)%*%(W.ABC.1%*%d.ABC.1 + W.ABC.2%*%d.ABC.2 + W.ABC.3%*%d.ABC.3)
V.ABC = solve(W.ABC)

# combine all effect size estimates into one vector
```

```
y.gls = c(d.hat.ABC,
          d.hat.BC,
          d.hat.AB)

# create variance and weight matrix for intitial estimates
V.gls = matrix(0,
               nrow = 4,
               ncol = 4)

V.gls[1:2,1:2] = V.ABC
V.gls[3,3] = s2.d.hat.BC.1
V.gls[4,4] = s2.d.hat.AB.1

W.gls = solve(V.gls)

# Calculate and check the GLS fixed effects estimates ####

# create H matrix
H = matrix(c(1,0,
             0,1,
             -1,1),
           byrow = TRUE,
           nrow = 3)

# create X matrix
X = matrix(c(1,0,
             0,1,
             -1,1,
             1,0),
           byrow = TRUE,
           nrow = 4)

# calculate the GLS effect size estimates and the associated covariance matrix
d.hat.gls = H%*%solve(t(X)%*%W.gls%*%X)%*%t(X)%*%W.gls%*%y.gls

# Code for ENT checkpoints for Dataset A ####

# retrieve the sample mean differences from the contrast data
y.ent = contrast.data[, 1]

# create the sub edge-vertex incidence matrix for design ABC
B.ABC = matrix(c(1,-1,0,
                 1,0,-1,
                 0,1,-1),
               nrow = 3,
               byrow = TRUE)

# create the sub edge-vertex incidence matrix for design BC
B.BC = c(0,1,-1)

# create the sub edge-vertex incidence matrix for design AB
B.AB = c(1,-1,0)

# create the edge-vertex incidence matrix
B = rbind(B.ABC,
          B.ABC,
          B.BC,
          B.AB,
          B.BC,
          B.ABC)

# create the sub variance matrices for experiments with design ABC
V1 = matrix(c(0,contrast.data[1,2],contrast.data[2,2],
              contrast.data[1,2],0,contrast.data[3,2],
              contrast.data[2,2],contrast.data[3,2],0)^2,
            byrow = TRUE,
            nrow = 3)

V2 = matrix(c(0,contrast.data[4,2],contrast.data[5,2],
              contrast.data[4,2],0,contrast.data[6,2],
              contrast.data[5,2],contrast.data[6,2],0)^2,
            byrow = TRUE,
            nrow = 3)

V3 = matrix(c(0,contrast.data[10,2],contrast.data[11,2],
              contrast.data[10,2],0,contrast.data[12,2],
              contrast.data[11,2],contrast.data[12,2],0)^2,
            byrow = TRUE,
            nrow = 3)

# create the Lplus matrices for experiments with design ABC
```

```
L1.plus = (-1/(2*(3^2)))*t(B.ABC)%*%B.ABC%*%V1%*%t(B.ABC)%*%B.ABC
L2.plus = (-1/(2*(3^2)))*t(B.ABC)%*%B.ABC%*%V2%*%t(B.ABC)%*%B.ABC
L3.plus = (-1/(2*(3^2)))*t(B.ABC)%*%B.ABC%*%V3%*%t(B.ABC)%*%B.ABC

# create the L matrices for experiments with design ABC
L1 = MP.inv(L1.plus)
L2 = MP.inv(L2.plus)
L3 = MP.inv(L3.plus)

# retrieve the weights for experiments with design ABC
W1 = -c(L1[1,2],L1[1,3],L1[2,3])
W2 = -c(L2[1,2],L2[1,3],L2[2,3])
W3 = -c(L3[1,2],L3[1,3],L3[2,3])

# create the weight matrix
W.ent = diag(c(W1,W2,1/(contrast.data[7:9,2])^2,W3))

# create L matrix
L = t(B)%*%W.ent%*%B

# create Lplus matrix
L.plus = MP.inv(L)

# create effect size estimate vector
d.hat.ent = B%*%L.plus%*%t(B)%*%W.ent%*%y.ent
```

Appendix C: Selected R Code for Chapter 3

Section 2

```
# load packages
library(ggplot2)

# Create data for first plot of Figure 15 ####

# assign probabilities
p1 = .1
p2 = .55
p3 = .35

# create data
prob.plot.data = data.frame(x = c(-1.65,
                                  mean(c(qnorm(p1),qnorm(p1+p2))),
                                  1.65),
                            y = 0.065)

quant.plot.data = data.frame(x = c(qnorm(p1),
                                   qnorm(p1+p2))-0.15,
                             y = 0.45)

# Create first plot for Figure 15 ####
ggplot(data = data.frame(X = 0),
       aes(x = X)) +
  stat_function(fun = dnorm,
                args = list(mean = 0, sd = 1)) +
  scale_x_continuous("",
                     limits = c(-3.5,3.5),
                     breaks = -3:3) +
  scale_y_continuous("",
                     limits = c(0,.475)) +
  stat_function(fun = dnorm,
                xlim = c(-3.5, qnorm(p1)),
                geom = "area",
                fill = "green",
                color = "black") +
  stat_function(fun = dnorm,
                xlim = c(qnorm(p1), qnorm(p1+p2)),
                geom = "area",
                fill = "yellow",
                color = "black") +
  stat_function(fun = dnorm,
                xlim = c(qnorm(p1+p2), 3.5),
                geom = "area",
                fill = "red",
                color = "black") +
  geom_vline(xintercept = c(qnorm(p1),
                            qnorm(p1+p2)),
             size = 1) +
  geom_hline(yintercept = 0,
             size = 0.25) +
  geom_point(aes(x = qnorm(p1),
                 y = 0),
             size = 5,
             inherit.aes = FALSE) +
  geom_point(aes(x = qnorm(p1+p2),
                 y = 0),
             size = 5,
             inherit.aes = FALSE) +
  geom_text(data = prob.plot.data,
            aes(x = x,
                y = y),
            label = c(expression(p["1"]),
                      expression(p["2"]),
                      expression(p["3"])),
            size = 10,
            inherit.aes = FALSE) +
  geom_text(data = quant.plot.data,
            aes(x = x,
                y = y),
            label = c(expression(q["1"]),
                      expression(q["2"])),
            size = 10,
            inherit.aes = FALSE) +
```

```
  geom_text(x = qnorm(p1)+.35,
            y = .02,
            label = expression("("*Q[1]*",0)"),
            size = 10,
            inherit.aes = FALSE) +
  geom_text(x = qnorm(p1+p2)+.35,
            y = .02,
            label = expression("("*Q[2]*",0)"),
            size = 10,
            inherit.aes = FALSE) +
  ggtitle("Initial Pain Score Distribution",
          subtitle = "Before Treatment with Compound A") +
  theme(text = element_text(size = 28),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        axis.text.y = element_blank(),
        axis.ticks.y = element_blank())

# Create data for second plot of Figure 15 ####

# assign delta
delta.data = 0.6

# create data
prob.plot.data2 = data.frame(x = c(-1.65,
                                   mean(c(qnorm(p1),qnorm(p1+p2))) + delta.data,
                                   1.65),
                             y = 0.065)

quant.plot.data2 = data.frame(x = c(qnorm(p1),
                                    qnorm(p1+p2)) + delta.data - 0.15,
                              y = 0.45)

# create second plot of Figure 15 ####
ggplot(data = data.frame(X = 0),
       aes(x = X)) +
  stat_function(fun = dnorm,
                args = list(mean = 0, sd = 1)) +
  scale_x_continuous("",
                     limits = c(-3.5,3.5),
                     breaks = -3:3) +
  scale_y_continuous("",
                     limits = c(0,.475)) +
  stat_function(fun = dnorm,
                xlim = c(-3.5, qnorm(p1) + delta.data),
                geom = "area",
                fill = "green",
                color = "black") +
  stat_function(fun = dnorm,
                xlim = c(qnorm(p1), qnorm(p1+p2)) + delta.data,
                geom = "area",
                fill = "yellow",
                color = "black") +
  stat_function(fun = dnorm,
                xlim = c(qnorm(p1+p2) + delta.data, 3.5),
                geom = "area",
                fill = "red",
                color = "black") +
  geom_vline(xintercept = c(qnorm(p1),
                            qnorm(p1+p2)) + delta.data,
             size = 1) +
  geom_hline(yintercept = 0,
             size = 0.25) +
  geom_point(aes(x = qnorm(p1) + delta.data,
                 y = 0),
             size = 5,
             inherit.aes = FALSE) +
  geom_point(aes(x = qnorm(p1+p2) + delta.data,
                 y = 0),
             size = 5,
             inherit.aes = FALSE) +
  geom_text(data = prob.plot.data2,
            aes(x = x,
                y = y),
            label = c(expression(p["1"]^"T"),
                      expression(p["2"]^"T"),
                      expression(p["3"]^"T")),
            size = 10,
            inherit.aes = FALSE) +
  geom_text(data = quant.plot.data2,
            aes(x = x,
```

```
              y = y),
            label = c(expression(q["1"]^"'"),
                      expression(q["2"]^"'")),
            size = 10,
            inherit.aes = FALSE) +
  geom_text(x = qnorm(p1) + delta.data + .4,
            y = .02,
            label = expression("("*Q[1]*"+"*delta*",0)"),
            size = 10,
            inherit.aes = FALSE) +
  geom_text(x = qnorm(p1+p2) + delta.data + .4,
            y = .02,
            label = expression("("*Q[2]*"+"*delta*",0)"),
            size = 10,
            inherit.aes = FALSE) +
  ggtitle("Hypothesized Pain Score Distribution",
          subtitle = "After Treatment with Compound A") +
  theme(text = element_text(size = 28),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```

Section 4

```
# load packages
library(ggplot2)
library(data.table)

# load functions
source("multinorm Package/corr est.R")
source("multinorm Package/create exp cells.R")
source("multinorm Package/create obs cells.R")
source("functions/pre gen stats.R")
source("functions/gen stats.R")


# Create plot for Figure 16 ####

# Create data ####
tmp.table = create.exp.cells(rho = 0,
                             v.cuts = -1,
                             h.cuts = -1.5)

# retrieve the data for creating the sections
polydata = corr.est3(rho = 0,
                     prob.obs = tmp.table)

# convert the infinity values

# initialize the data for geom_polygon
polydata2 = data.frame(ID = rep(1:nrow(polydata), each = 4),
                       fill.value = rep(paste(polydata[,2],
                                              polydata[,3],
                                              sep = ""), each = 4),
                       X.coord = 0,
                       Y.coord = 0,
                       X.center = 0,
                       Y.center = 0)

# cycle through the sections
for (i in 1:nrow(polydata)){

  # retrieve the information for this section
  section.info = as.numeric(polydata[i,4:7])

  # retrieve the X coordinates for this section
  polydata2[(1+4*(i-1)):(4*i), 3] = c(section.info[1],
                                      section.info[3],
                                      section.info[3],
                                      section.info[1])

  # retrieve the Y coordinates for this section
  polydata2[(1+4*(i-1)):(4*i), 4] = c(section.info[2],
                                      section.info[2],
                                      section.info[4],
                                      section.info[4])

  # convert infinities to 4 for calculating center
```

```
  section.info[which(section.info == -Inf)] = -4
  section.info[which(section.info == Inf)] = 4

  # retrieve the X center
  polydata2[(1+4*(i-1)):(4*i), 5] = mean(c(section.info[1],
                                           section.info[3]))

  # retrieve the Y center
  polydata2[(1+4*(i-1)):(4*i), 6] = mean(c(section.info[2],
                                           section.info[4]))

}

# retrieve the horizontal and vertical dividers
v.dashed = unique(polydata[, 4])[-1]
h.dashed = unique(polydata[, 5])[-length(unique(polydata[, 5]))]

# create plot ####
ggplot(data = data.frame(X = 0),
       aes(x = X)) +
  scale_x_continuous("",
                     limits = c(-4,4)) +
  scale_y_continuous("",
                     limits = c(-4,4)) +
  geom_polygon(data = polydata2,
               aes(x = X.coord,
                   y = Y.coord,
                   fill = fill.value),
               inherit.aes = FALSE) +
  scale_fill_manual("",
                    values = c("green",
                               rep("yellow",2),
                               "orange")) +
  geom_text(data = polydata2,
            aes(x = X.center,
                y = Y.center,
                label = fill.value),
            size = 10) +
  geom_vline(xintercept = v.dashed[1],
             size = 1,
             linetype = "dashed") +
  geom_hline(yintercept = h.dashed[1],
             size = 1,
             linetype = "dashed") +
  geom_vline(xintercept = 0,
             size = 1) +
  geom_hline(yintercept = 0,
             size = 1) +
  geom_point(aes(x = v.dashed[1],
                 y = 0),
             size = 5,
             inherit.aes = FALSE) +
  geom_point(aes(x = 0,
                 y = h.dashed[1]),
             size = 5,
             inherit.aes = FALSE) +
  geom_text(x = 3.9,
            y = -.25,
            label = expression(X[1]),
            size = 10,
            inherit.aes = FALSE) +
  geom_text(x = .15,
            y = 3.9,
            label = expression(X[2]),
            size = 10,
            inherit.aes = FALSE) +
  geom_text(x = -1.2,
            y = 2,
            label = expression(q[11]),
            size = 10,
            inherit.aes = FALSE) +
  geom_text(x = 3,
            y = -1.7,
            label = expression(q[21]),
            size = 10,
            inherit.aes = FALSE) +
  geom_text(x = -.6,
            y = .4,
            label = expression("("*Q[11]*",0)"),
            size = 10,
            inherit.aes = FALSE) +
```

```
  geom_text(x = .4,
            y = -1.1,
            label = expression("(0,"*Q[21]*")"),
            size = 10,
            inherit.aes = FALSE) +
  ggtitle("Bivariate Distribution Layout",
          subtitle = "2 Levels For Each Variable") +
  theme(text = element_text(size = 28),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        axis.text.x = element_blank(),
        axis.ticks.x = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks.y = element_blank(),
        legend.position = "none")

# Create top plot for Figure 17 ####
# Create data ####

tmp.table = create.exp.cells(rho = 0,
                             v.cuts = c(-2, .75),
                             h.cuts = c(-1.75,1))

# retrieve the data for creating the sections
polydata = corr.est3(rho = 0,
                     prob.obs = tmp.table)

# convert the infinity values

# initialize the data for geom_polygon
polydata2 = data.frame(ID = rep(1:nrow(polydata), each = 4),
                       fill.value = rep(paste(polydata[,2],
                                              polydata[,3],
                                              sep = ""), each = 4),
                       X.coord = 0,
                       Y.coord = 0,
                       X.center = 0,
                       Y.center = 0)

# cycle through the sections
for (i in 1:nrow(polydata)){

  # retrieve the information for this section
  section.info = as.numeric(polydata[i,4:7])

  # retrieve the X coordinates for this section
  polydata2[(1+4*(i-1)):(4*i), 3] = c(section.info[1],
                                      section.info[3],
                                      section.info[3],
                                      section.info[1])

  # retrieve the Y coordinates for this section
  polydata2[(1+4*(i-1)):(4*i), 4] = c(section.info[2],
                                      section.info[2],
                                      section.info[4],
                                      section.info[4])

  # convert infinities to 4 for calculating center
  section.info[which(section.info == -Inf)] = -4
  section.info[which(section.info == Inf)] = 4

  # retrieve the X center
  polydata2[(1+4*(i-1)):(4*i), 5] = mean(c(section.info[1],
                                           section.info[3]))

  # retrieve the Y center
  polydata2[(1+4*(i-1)):(4*i), 6] = mean(c(section.info[2],
                                           section.info[4]))

}

# retrieve the horizontal and vertical dividers
v.dashed = unique(polydata[, 4])[-1]
h.dashed = unique(polydata[, 5])[-length(unique(polydata[, 5]))]

# Create plot ####
ggplot(data = data.frame(X = 0),
       aes(x = X)) +
  scale_x_continuous("",
                     limits = c(-4,4)) +
  scale_y_continuous("",
```

```
                   limits = c(-4,4)) +
geom_polygon(data = polydata2,
             aes(x = X.coord,
                 y = Y.coord,
                 fill = fill.value),
             inherit.aes = FALSE) +
scale_fill_manual("",
                   values = c("green",
                              "yellow",
                              "orange",
                              rep("yellow",2),
                              "orange",
                              rep("orange",2),
                              "red")) +
geom_text(data = polydata2,
          aes(x = X.center,
              y = Y.center,
              label = fill.value),
          size = 10) +
geom_vline(xintercept = v.dashed[1],
           size = 1,
           linetype = "dashed") +
geom_vline(xintercept = v.dashed[2],
           size = 1,
           linetype = "dashed") +
geom_hline(yintercept = h.dashed[1],
           size = 1,
           linetype = "dashed") +
geom_hline(yintercept = h.dashed[2],
           size = 1,
           linetype = "dashed") +
geom_vline(xintercept = 0,
           size = 1) +
geom_hline(yintercept = 0,
           size = 1) +
geom_point(aes(x = -2,
               y = 0),
           size = 5,
           inherit.aes = FALSE) +
geom_point(aes(x = .75,
               y = 0),
           size = 5,
           inherit.aes = FALSE) +
geom_point(aes(x = 0,
               y = -1.75),
           size = 5,
           inherit.aes = FALSE) +
geom_point(aes(x = 0,
               y = 1),
           size = 5,
           inherit.aes = FALSE) +
geom_text(x = 3.9,
          y = -.25,
          label = "U",
          size = 10,
          inherit.aes = FALSE) +
geom_text(x = .15,
          y = 3.9,
          label = "L",
          size = 10,
          inherit.aes = FALSE) +
geom_text(x = -2.2,
          y = 2.5,
          label = expression(q[11]),
          size = 10,
          inherit.aes = FALSE) +
geom_text(x = .55,
          y = 2.5,
          label = expression(q[12]),
          size = 10,
          inherit.aes = FALSE) +
geom_text(x = 3,
          y = -1.95,
          label = expression(q[21]),
          size = 10,
          inherit.aes = FALSE) +
geom_text(x = 3,
          y = .8,
          label = expression(q[22]),
          size = 10,
          inherit.aes = FALSE) +
```

```
  geom_text(x = -1.6,
            y = .35,
            label = expression("("*Q[11]*",0)"),
            size = 10,
            inherit.aes = FALSE) +
  geom_text(x = 1.15,
            y = .35,
            label = expression("("*Q[12]*",0)"),
            size = 10,
            inherit.aes = FALSE) +
  geom_text(x = -.4,
            y = -1.35,
            label = expression("(0,"*Q[21]*")"),
            size = 10,
            inherit.aes = FALSE) +
  geom_text(x = -.4,
            y = 1.35,
            label = expression("(0,"*Q[22]*")"),
            size = 10,
            inherit.aes = FALSE) +
  ggtitle("Upper and Lower Pain Score Distribution",
          subtitle = "Before Treatment with Compound C") +
  theme(text = element_text(size = 28),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        axis.text.x = element_blank(),
        axis.ticks.x = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks.y = element_blank(),
        legend.position = "none")

# Create bottom plot for Figure 17 ####
# Create data ####

delta.data = c(.75,1)

tmp.table = create.exp.cells(rho = 0,
                             v.cuts = c(-2, .75) + delta.data[1],
                             h.cuts = c(-1.75,1) + delta.data[2])

# retrieve the data for creating the sections
polydata = corr.est3(rho = 0,
                     prob.obs = tmp.table)

# initialize the data for geom_polygon
polydata2 = data.frame(ID = rep(1:nrow(polydata), each = 4),
                       fill.value = rep(paste(polydata[,2],
                                              polydata[,3],
                                              sep = ""), each = 4),
                       X.coord = 0,
                       Y.coord = 0,
                       X.center = 0,
                       Y.center = 0)

# cycle through the sections
for (i in 1:nrow(polydata)){

  # retrieve the information for this section
  section.info = as.numeric(polydata[i,4:7])

  # retrieve the X coordinates for this section
  polydata2[(1+4*(i-1)):(4*i), 3] = c(section.info[1],
                                      section.info[3],
                                      section.info[3],
                                      section.info[1])

  # retrieve the Y coordinates for this section
  polydata2[(1+4*(i-1)):(4*i), 4] = c(section.info[2],
                                      section.info[2],
                                      section.info[4],
                                      section.info[4])

  # convert infinities to 4 for calculating center
  section.info[which(section.info == -Inf)] = -4
  section.info[which(section.info == Inf)] = 4

  # retrieve the X center
  polydata2[(1+4*(i-1)):(4*i), 5] = mean(c(section.info[1],
                                           section.info[3]))

  # retrieve the Y center
```

```
  polydata2[(1+4*(i-1)):(4*i), 6] = mean(c(section.info[2],
                                           section.info[4]))

}

# retrieve the horizontal and vertical dividers
v.dashed = unique(polydata[, 4])[-1]
h.dashed = unique(polydata[, 5])[-length(unique(polydata[, 5]))]

# Create plot ####
ggplot(data = data.frame(X = 0),
       aes(x = X)) +
  scale_x_continuous("",
                     limits = c(-4,4)) +
  scale_y_continuous("",
                     limits = c(-4,4)) +
  geom_polygon(data = polydata2,
               aes(x = X.coord,
                   y = Y.coord,
                   fill = fill.value),
               inherit.aes = FALSE) +
  scale_fill_manual("",
                    values = c("green",
                               "yellow",
                               "orange",
                               rep("yellow",2),
                               "orange",
                               rep("orange",2),
                               "red")) +
  geom_text(data = polydata2,
            aes(x = X.center,
                y = Y.center,
                label = fill.value),
            size = 10) +
  geom_vline(xintercept = v.dashed[1],
             size = 1,
             linetype = "dashed") +
  geom_vline(xintercept = v.dashed[2],
             size = 1,
             linetype = "dashed") +
  geom_hline(yintercept = h.dashed[1],
             size = 1,
             linetype = "dashed") +
  geom_hline(yintercept = h.dashed[2],
             size = 1,
             linetype = "dashed") +
  geom_vline(xintercept = 0,
             size = 1) +
  geom_hline(yintercept = 0,
             size = 1) +
  geom_point(aes(x = v.dashed[1],
                 y = 0),
             size = 5,
             inherit.aes = FALSE) +
  geom_point(aes(x = v.dashed[2],
                 y = 0),
             size = 5,
             inherit.aes = FALSE) +
  geom_point(aes(x = 0,
                 y = h.dashed[1]),
             size = 5,
             inherit.aes = FALSE) +
  geom_point(aes(x = 0,
                 y = h.dashed[2]),
             size = 5,
             inherit.aes = FALSE) +
  geom_text(x = 3.9,
            y = -.25,
            label = "U",
            size = 10,
            inherit.aes = FALSE) +
  geom_text(x = .15,
            y = 3.9,
            label = "L",
            size = 10,
            inherit.aes = FALSE) +
  geom_text(x = -2.2 + delta.data[1],
            y = 2.5,
            label = expression(q[11]^"'"),
            size = 10,
            inherit.aes = FALSE) +
```

```
  geom_text(x = .55  + delta.data[1],
            y = 2.5,
            label = expression(q[12]^"'"),
            size = 10,
            inherit.aes = FALSE) +
  geom_text(x = 3,
            y = -1.95 + delta.data[2] -.15,
            label = expression(q[21]^"'"),
            size = 10,
            inherit.aes = FALSE) +
  geom_text(x = 3,
            y = .8 + delta.data[2] -.15,
            label = expression(q[22]^"'"),
            size = 10,
            inherit.aes = FALSE) +
  geom_text(x = -1.85,
            y = .35,
            label = expression("("*Q[11]*"+"*delta["U"]*",0)"),
            size = 10,
            inherit.aes = FALSE) +
  geom_text(x = .9,
            y = .35,
            label = expression("("*Q[12]*"+"*delta["U"]*",0)"),
            size = 10,
            inherit.aes = FALSE) +
  geom_text(x = -.55,
            y = -1.1,
            label = expression("(0,"*Q[21]*"+"*delta["L"]*")"),
            size = 10,
            inherit.aes = FALSE) +
  geom_text(x = -.55,
            y = 1.65,
            label = expression("(0,"*Q[22]*"+"*delta["L"]*")"),
            size = 10,
            inherit.aes = FALSE) +
  ggtitle("Hypothesized Upper and Lower Pain Score Distribution",
          subtitle = "After Treatment with Compound C") +
  theme(text = element_text(size = 28),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        axis.text.x = element_blank(),
        axis.ticks.x = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks.y = element_blank(),
        legend.position = "none")

# create the vector of sample sizes
SS.sim.vec = seq(20,50,10)

# create the vector of true rhos
rho.sim.vec = seq(-.99,.99,.01)

# initialize the graph data
graph.data3 = data.frame(True.Rho = rep(rho.sim.vec, each = length(SS.sim.vec)),
                         Sample.Size = rep(SS.sim.vec, length(rho.sim.vec)),
                         Rho1.Bias = 0,
                         Rho2.Bias = 0,
                         Rho3.Bias = 0)


# set number of iterations
num.iterations = 1000

# 1st configuration ####
v.cuts = 0
h.cuts = 0

# conduct simulation for configuration 1
for (rhos in rho.sim.vec){

  for (SS in SS.sim.vec){

    tmp.index = which(graph.data3[, "True.Rho"] == rhos & graph.data3[, "Sample.Size"] ==
SS)

    tmp.stats = gen.stats(iterations = num.iterations,
                          N = SS,
                          true.rho = rhos,
                          v.cuts = v.cuts,
                          h.cuts = h.cuts,
                          no.cores = 7)
```

```
    graph.data3[tmp.index, "Rho1.Bias"] = mean(tmp.stats[tmp.stats[, "Group"] == "Rho1
Bias", "Statistic"])
    graph.data3[tmp.index, "Rho2.Bias"] = mean(tmp.stats[tmp.stats[, "Group"] == "Rho2
Bias", "Statistic"])
    graph.data3[tmp.index, "Rho3.Bias"] = mean(tmp.stats[tmp.stats[, "Group"] == "Rho3
Bias", "Statistic"])

  }

}

# reconfigure data for graphing
for.bind1 = graph.data3[, 1:3]
for.bind2 = graph.data3[, c(1:2,4)]
for.bind3 = graph.data3[, c(1:2,5)]

colnames(for.bind1)[3] = "Bias"
colnames(for.bind2)[3] = "Bias"
colnames(for.bind3)[3] = "Bias"

graph.data4 = rbind.data.frame(for.bind1,
                               for.bind2,
                               for.bind3)

graph.data4[, "Estimator"] = rep(c("Xi",
                                   "Spearman",
                                   "Kendall"),
                                 each = nrow(for.bind1))

graph.data4[, "Estimator"] = factor(graph.data4[, "Estimator"],
                                    levels = unique(graph.data4[, "Estimator"]))

graph.data4[, "Sample.Size.Label"] = paste("N = ",
                                           graph.data4[, "Sample.Size"],
                                           sep = "")

# create Figure 18
ggplot(data = graph.data4,
       aes(x = True.Rho,
           y = Bias,
           color = Estimator)) +
  facet_wrap(~ Sample.Size.Label,
             nrow = 3) +
  geom_line() +
  geom_hline(yintercept = 0,
             color = "green",
             linetype = "dashed") +
  scale_x_continuous("True Correlation") +
  scale_y_continuous("Bias") +
  scale_color_manual(bquote("Estimator"),
                     values = c("blue",
                                "red",
                                "magenta"),
                     labels = c(expression(xi*"    "),
                                expression("Spearman's "*rho*"    "),
                                expression("Kendall's "*tau)),
                     guide = guide_legend(title.position = "top",
                                          ncol = 3,
                                          keywidth = .25,
                                          default.unit = "inch",
                                          override.aes = list(size = 2))) +
  ggtitle("Estimator Bias for Partition Configuration 1") +
  theme(text = element_text(size = 28),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        legend.title = element_blank(),
        legend.position = "top")

# conduct simulation for configuration 2
for (rhos in rho.sim.vec){

  for (SS in SS.sim.vec){

    tmp.index = which(graph.data3[, "True.Rho"] == rhos & graph.data3[, "Sample.Size"] ==
SS)

    tmp.stats = gen.stats(iterations = num.iterations,
                          N = SS,
                          true.rho = rhos,
                          v.cuts = v.cuts,
```

```
                          h.cuts = h.cuts,
                          no.cores = 7)

    graph.data3[tmp.index, "Rho1.Bias"] = mean(tmp.stats[tmp.stats[, "Group"] == "Rho1
Bias", "Statistic"])
    graph.data3[tmp.index, "Rho2.Bias"] = mean(tmp.stats[tmp.stats[, "Group"] == "Rho2
Bias", "Statistic"])
    graph.data3[tmp.index, "Rho3.Bias"] = mean(tmp.stats[tmp.stats[, "Group"] == "Rho3
Bias", "Statistic"])

  }

}

# reconfigure data for graphing
for.bind1 = graph.data3[, 1:3]
for.bind2 = graph.data3[, c(1:2,4)]
for.bind3 = graph.data3[, c(1:2,5)]

colnames(for.bind1)[3] = "Bias"
colnames(for.bind2)[3] = "Bias"
colnames(for.bind3)[3] = "Bias"

graph.data4 = rbind.data.frame(for.bind1,
                               for.bind2,
                               for.bind3)

graph.data4[, "Estimator"] = rep(c("Xi",
                                   "Spearman",
                                   "Kendall"),
                                 each = nrow(for.bind1))

graph.data4[, "Estimator"] = factor(graph.data4[, "Estimator"],
                                    levels = unique(graph.data4[, "Estimator"]))

graph.data4[, "Sample.Size.Label"] = paste("N = ",
                                           graph.data4[, "Sample.Size"],
                                           sep = "")

# create Figure 19
ggplot(data = graph.data4,
       aes(x = True.Rho,
           y = Bias,
           color = Estimator)) +
  facet_wrap(~ Sample.Size.Label,
             nrow = 3) +
  geom_line() +
  geom_hline(yintercept = 0,
             color = "green",
             linetype = "dashed") +
  scale_x_continuous("True Correlation") +
  scale_y_continuous("Bias") +
  scale_color_manual("",
                     values = c("blue",
                                "red",
                                "magenta"),
                     labels = c(expression(xi*"    "),
                                expression("Spearman's "*rho*"    "),
                                expression("Kendall's "*tau)),
                     guide = guide_legend(title.position = "top",
                                          ncol = 3,
                                          keywidth = .25,
                                          default.unit = "inch",
                                          override.aes = list(size = 2))) +
  ggtitle("Estimator Bias for Partition Configuration 2") +
  theme(text = element_text(size = 28),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        legend.title = element_blank(),
        legend.position = "top")
```

Appendix D: Publications

1. Data Nuggets: A Method for Reducing Large Datasets While Maintaining Data Structure. Also developing package. (In Preparation, 2019)
2. A New Understanding of Network Meta-Analysis Regarding Experiments with Small Sample Sizes. Also developing package. (In Preparation, 2019)
3. Determining Adequate Sample Sizes for Analyzing Ordinal Categorical Data with Small Range Scales. Also developing package. (In Preparation, 2019)
4. (2019) Cuccurullo S, Fleming T, (et.al. including Beavers, T). Impact of a Stroke Recovery Program Integrating Modified Cardiac Rehabilitation on All-cause Mortality, Cardiovascular Performance and Overall Function (Accepted to *American Journal of Physical Medicine & Rehabilitation* pending minor revisions)
5. (2019) Barbayannis G, Chiu I, (et.al. including Beavers, T). Relation Between Statewide Hospital Performance Reports on Myocardial Infarction and Cardiovascular Outcomes (Submitted to *American Journal of Cardiology*)

Bibliography

1. Albert, A. (1972). *Regression and the Moore-Penrose pseudoinverse.* New York: Academic Press.

2. Amaratunga, D. (1999). Searching for the Right Sample Size. *The American Statistician*, 52-55.

3. Amaratunga, D., Cabrera, J., & Shkedy, Z. (2014). *Exploration and analysis of DNA microarray and other high dimensional data.* Hoboken, New Jersey: John Wiley & Sons.

4. Arslan, O. (2015). Variance-mean mixture of the multivariate skew normal distribution. *Statistical Papers*, 353-378.

5. Ayramo, S., & Karkkainen, T. (2006). Introduction to partitioning-based cluster analysis methods with a robust example. *Reports of the Department of Mathematical Information Technology; Series C: Software and Computational Engineering*, 1-36.

6. Azzalini, A., & Valle, A. D. (1996). The Multivariate Skew-Normal Distribution. *Biometrika*, 715-726.

7. Bailey, R. (2007). Designs for two-colour microarray experiments. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 365–394.

8. Bliss, C. (1934). The Method of Probits. *Science*, 38–39.

9. Borenstein, M. (2009). *Introduction to meta-analysis.* Chichester, West Sussex, U.K.: John Wiley & Sons.

10. Cabrera, J., & McDougall, A. (2002). *Statistical Consulting.* New York, NY: Springer.

11. Dias, S., Sutton, A. J., Ades, A. E., & Welton, N. J. (2013). Evidence Synthesis for Decision Making 2: A Generalized Linear Modeling Framework for Pairwise and Network Meta-analysis of Randomized Controlled Trials. *Medical Decision Making*, 607-617.

12. Dias, S., Welton, N. J., Caldwell, D. M., Ades, A. E., & Hougaard, P. (2010). Checking consistency in mixed treatment comparison meta-analysis. *Statistics in Medicine*, 932-944.

13. Fisher, R. A. (1935). *The Design of Experiments, Second Edition.* Edinburgh: Oliver and Boyd.

14. Forgy, E. (1965). Cluster analysis of multivariate data: efficiency vs. interpretability of classification. *Biometrics*, 768.

15. Franchini, A., Dias, S., Ades, A., Jansen, J., & Welton, N. (2012). Accounting for correlation in network meta-analysis with multi-arm trials. *Research Synthesis Methods*, 142-160.

16. Friedman, J., & Tukey, J. (1974). A Projection Pursuit Algorithm for Exploratory Data Analysis. *IEEE Transactions on Computers*, 881-890.

17. Ghosh, D., Cabrera, J., Adam, T., Aggarwal, R., Levounis, P., & Adam, N. (2016). Comorbidity Patterns and its Impact on Health Outcomes: Two-way Clustering Analysis. *IEEE Transactions on Big Data*.

18. Hartigan, J., & Wong, M. (1979). AS136 A K-means clustering algorithm. *Applied Statistics*, 90.

19. Higgins, J., & Whitehead, A. (1996). BORROWING STRENGTH FROM EXTERNAL TRIALS IN A META-ANALYSIS. *Statistics in Medicine*, 2733–2749.

20. Hong, H., Carlin, B. P., Shamliyan, T. A., Wyman, J. F., Ramakrishnan, R., Sainfort, F., & Kane, R. L. (2013). Comparing Bayesian and Frequentist Approaches for Multiple Outcome Mixed Treatment Comparisons. *Medical Decision Making*, 702-714.

21. Jahan-Tigh, R. R., Ryan, C., Obermoser, G., & Schwarzenberger, K. (2012). Flow Cytometry. *Journal of Investigative Dermatology*, 1-6.

22. Jansen, J. P., & Cope, S. (2012). Meta-regression models to address heterogeneity and inconsistency in network meta-analysis of survival outcomes. *Jansen, Jeroen P ; Cope, Shannon*.

23. Katsanos, K. (2014). Appraising Inconsistency Between Direct and Indirect Estimates. In Biondi-Zoccai G, *Network Meta-Analysis: Evidence Synthesis with Mixed Treatment Comparison* (pp. 191–210). Hauppauge, NY: Nova Science Publishers.

24. Kendall, M. (1938). A New Measure of Rank Correlation. *Biometrika*, 81–93.

25. König, J., Krahn, U., & Binder, H. (2013). Visualizing the flow of evidence in network meta-analysis and characterizing mixed treatment comparisons. *Statistics in Medicine*, 5414-5429.

26. Kotz, S., & Nadarajah, S. (2004). *Multivariate T-Distributions and Their Applications.* Cambridge: Cambridge University Press.

27. Liu, J., & Xu, S. (2014). Applied research of weighted K-means algorithm in social networks. *Applied Mechanics and Materials*, 286–290.

28. Lloyd, S. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 129–137.

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

30. Lu, G., & Ades, A. E. (2006). Assessing Evidence Inconsistency in Mixed Treatment Comparisons. *Journal of the American Statistical Association*, 447-459.

31. Lu, G., Welton, N., Higgins, J., White, I., & Ades, A. (2011). Linear inference for mixed treatment comparison meta-analysis: A two-stage approach. *Research Synthesis Methods*, 43–60.

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

33. MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium On Mathematical Statistics and Probabilities*, 281-296.

34. Mak, S. (2018). support: Support Points. R package version 0.1.2.

35. Mak, S., & Joseph, V. (2018). Support points. *Annals of Statistics*, 2562–2592.

36. Morissette, L., & Chartier, S. (2013). The k-means clustering technique: General considerations and implementation in Mathematica. *Tutorials in Quantitative Methods for Psychology*, 15–24.

37. Neupane, B., Richer, D., Bonner, A., Kibret, T., Beyene, J., & Neupane, B. (2014). Network meta-analysis using R: a review of currently available automated packages. *PloS One*.

38. Neuvial, P., Bengtsson, H., & Speed, T. P. (2010). TumorBoost: Normalization of allele-specific tumor copy numbers from a single pair of tumor-normal genotyping microarrays. *BMC Bioinformatics*, 245.

39. Pearson, K. (1900). On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *Philosophical Magazine*, 157-175.

40. R Core Team (2018). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

41. Riley, R. (2009). Multivariate meta-analysis: the effect of ignoring within-study correlation. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 789–811.

42. Rokach, L., & Maimon, O. (2005). *Data mining and knowledge discovery handbook.* Boston, MA: Springer US.

43. Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 53-65.

44. Rücker, G., & Schwarzer, G. (2012). Network meta-analysis, electrical networks and graph theory. *Research Synthesis Methods*, 312–324.

45. Rücker, G., & Schwarzer, G. (2014). Reduce dimension or reduce weights? Comparing two approaches to multi-arm studies in network meta-analysis. *Statistics in Medicine,* 4353–4369.

46. Rücker, G., Schwarzer, G., Krahn, U., & König, J. (2018). netmeta: Network Meta-Analysis using Frequentist Methods. R package version 0.9-8.

47. Schwarzer, G., Carpenter, J., & Rücker, G. (2015). *Meta-analysis with R.* Cham, Switzerland: Springer.

48. Snedecor, S. J., Patel, D. A., & C. Cappeller, J. (2014). Chapter 2. From Pairwise To Network Meta-Analyses. In G. Biondi-Zoccai, *Network Meta-Analysis: Evidence Synthesis with Mixed Treatment Comparison* (pp. 21-41). Hauppauge, NY: Nova Science Publishers.

49. Spearman, C. (1904). The Proof and Measurement of Association between Two Things. *American Journal of Psychology*, 72–101.

50. Srinivasan, S. (2018). *Guide to Big Data Applications.* Springer International Publishing.

51. Székely, G. J., & Rizzo, M. L. (2013). Energy statistics: A class of statistics based on distances. *Journal Of Statistical Planning And Inference*, 1249–1272.

52. Tibshirani, R., Walther, G., & Hastie, T. (2001). Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 411-423.

53. Valkenhoef, G., Lu, G., Brock, B., Hillege, H., Ades, A., & Welton, N. (2012). Automating network meta-analysis. *Research Synthesis Methods*, 285-299.

54. White, I. R., Barrett, J. K., Jackson, D., & Higgins, J. P. (2012). Consistency and inconsistency in network meta-analysis: model estimation using multivariate meta-regression. *Research Synthesis Methods*, 111-125.