Methods to Simulate Correlated Binomial Random Variables

METHODS TO SIMULATE CORRELATED BINOMIAL RANDOM VARIABLES

BY

WINFIELD LAI, B.Sc.

A THESIS

SUBMITTED TO THE DEPARTMENT OF MATHEMATICS & STATISTICS

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

Master of Science (2021) McMaster University

(Mathematics & Statistics) Hamilton, Ontario, Canada

TITLE:	Methods to Simulate Correlated Binomial Random Variables
AUTHOR:	Winfield Lai B.Sc. (Mathematics and Statistics) McMaster University, Hamilton, Canada B.Sc. (Chemical Biology) McMaster University, Hamilton, Canada
SUPERVISOR:	Dr. Angelo Canty & Dr. Katherine Davies
NUMBER OF PAGES:	viii, 112

Abstract

Single nucleotide polymorphisms (SNPs) have been involved in describing the risk a person is at for developing diseases. Simulating a collection of d correlated autosomal biallelic SNPs is useful to acquire empirical results for statistical tests in settings such as having a low sample size. A collection of d correlated autosomal biallelic SNPs can be modeled as a random vector $\boldsymbol{X} = (X_1, \ldots, X_d)$ where $X_i \sim \text{binomial}(2, p_i)$ and p_i is the minor allele frequency for the i^{th} SNP. The pairwise correlations between components of $\boldsymbol{X}$ can be specified by a $d \times d$ symmetric positive definite correlation matrix having all diagonal entries equal to one. Two versions of a novel method to simulate $\boldsymbol{X}$ are developed in this thesis; one version is based on generating correlated binomials directly and the other is based on generating correlated Bernoulli random vectors and summing them component wise. Two existing methods to simulate $\boldsymbol{X}$ are also discussed and implemented. In particular, a method involving the multivariate normal by Madsen and Birkes (2013) is compared to our novel methods for $d \geq 3$. Our novel binomial method has a different variance for the Fisher transformed sample correlation than the other two methods. Overall, if the target pairwise correlations are smaller than the lowest upper bound possible and the number of SNPs is low, then our novel Bernoulli method works the best since it is faster than the Madsen and Birkes method and has comparable variability and bias for sample correlation.

Acknowledgments

I would like to thank my supervisors Dr. Angelo Canty and Dr. Katherine Davies for their support and endless patience during this thesis. I would also like to thank Dr. Narayanaswamy Balakrishnan for his guidance and involvement and Dr. Pratheepa Jeganathan for her role as examiner. I would also like to acknowledge Dr. Spencer Smith, Dr. Roman Viveros-Aguilera and especially Dr. Angelo Canty for their role in my application to the graduate program as well as my undergraduate experience. I also want to thank my family for their support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Medicine today benefits from using statistical associations to assess whether a person is at risk of developing a disease and to help determine which pathways in the body merit further investigation. For example, methods associating genes to diseases have been used in prenatal testing, newborn screening and carrier testing, all of which increase the average quality of life by preventing or treating medical issues before they become life-threatening (Pagon et al., 2001). Statistical associations can help determine which pathways of the body merit further research by screening a set of markers in the genome to determine which are significantly associated with the disease of interest. In the case of multiple sclerosis, the set of significant markers were linked to autoimmune pathways and led to more research being focused there; the autoimmune aspect of MS was not heavily considered beforehand (Visscher et al., 2012).

Associating genes to traits can involve a large number of genes and even environmental factors. It is on a case-by-case basis if a single gene or a collection of genes is adequate to describe the risk of developing a trait or disease. In the case of

Huntington's disease, the presence of a single gene is sufficient to describe the risk of developing it (Myers, 2004). In contrast, a single gene is insufficient to reliably describe the risk of developing type 2 diabetes (Franks et al., 2013) and a collection of DNA regions, not necessarily just genes, and environmental factors are considered instead.

Genes are made up of Deoxyribonucleic acid (DNA) which forms the basic unit of hereditary material. A sequence of DNA is composed of nucleotides: adenine (A), thymine (T), guanine (G) and cytosine (C). Furthermore, DNA has a helical double stranded structure where each A, T, C and G on one strand is paired to a T, A, G and C respectively on the opposite strand. DNA is stored in 23 paired chromosomes, 22 pairs of which are non-sex chromosomes known as autosomes. Each chromosome in a pair of autosomes or sex chromosomes is independently inherited from one of the parents. The nucleotides at each location on one autosome is not necessarily the same as the nucleotides on the paired autosome. There is variation in the sequence of nucleotides between paired autosomes.

Single nucleotide polymorphisms (SNPs) are a common form of genetic variation. These are locations in the genome where a nucleotide can differ between individuals (Korf, 2004). SNPs may have up to four different variants or alleles because there are four different nucleotides. For example, if the two DNA sequences AG**C**TA and AG**A**TA occur in the exact same location in the genome in different individuals, then the third position of the DNA sequence is an example of a SNP with two alleles. We will restrict our work to the case of biallelic SNPs which only have two possible alleles and are the most frequently seen variations in the human genome. The allele that is present in the majority of a population is known as the major allele (denoted A in

what follows) and the less common allele is known as the minor allele (denoted a). SNPs located on autosomes have an allele on each copy of the autosome for a total of two alleles. Therefore, the two alleles for a single autosomal biallelic SNP will be AA, Aa or aa. Other forms of variation also exist in DNA but they will not be covered in this thesis.

We assume that biallelic SNPs located on autosomes will follow Hardy Weinberg Equilibrium (HWE) which is a useful model that describes the frequencies of AA, Aa and aa in a population (Mayo, 2008). The assumptions for the HWE model are: generations are discrete and non-overlapping - meaning that everyone in the population produces children at the same time and all at once - parents are picked at random (random mating), populations are infinitely large, no external forces are applied to the generation (i.e., natural selection), there is no mutation and the allele is located on an autosome. In reality, these assumptions do not hold, however the assumptions of the HWE model and its specification of AA, Aa and aa frequencies are reasonable enough to use in practice. Under the HWE model, the frequencies of AA, Aa and aa are $(1-p)^2$, $2p(1-p)$ and p^2 respectively where p is the frequency of the minor allele or MAF (Mayo, 2008).

Under the assumptions of the HWE model, the two alleles for an autosomal biallelic SNP an individual inherits can be thought of as two Bernoulli trials, one for each allele, where each trial has success probability equal to the MAF. Inheritance of the minor allele is deemed a success and inheritance of the major allele is deemed a failure. Each Bernoulli trial has the same success probability and are independent due to random mating in an infinitely large population. Therefore, if X represents the number of copies of the minor allele inherited by an individual at a given biallelic

SNP, then $X \sim \text{binomial}(2, p)$. Furthermore, this leads to the following relationship between the set of alleles an individual has or their genotype and the value of X:

$$\begin{aligned} \text{AA} &\implies X = 0 \\ \text{Aa} &\implies X = 1 \\ \text{aa} &\implies X = 2. \end{aligned} \tag{1.1}$$

When the biallelic SNP is located on the sex chromosomes, then the number of minor alleles an individual has is modeled differently. Since females have two X chromosomes, the number of minor alleles can still be modeled as a binomial random variable with two trials and success probability equal to the MAF. However, males have an X and a Y chromosome and a single SNP is not located on both chromosomes unlike autosome pairs. Therefore, the X and Y chromosomes are modeled separately for males. As a result, the number of minor alleles for a biallelic SNP located on a sex chromosome in a male is modeled as a Bernoulli random variable with success probability equal to the MAF. In this thesis, we will only consider biallelic SNPs located on autosomes.

HWE does not describe the joint distribution of genotypes from multiple SNPs. The joint distribution of genotypes from multiple SNPs is affected by recombination, a process that occurs during DNA inheritance. Recombination occurs in the parent and is when a DNA segment on one chromosome swaps with the corresponding DNA segment on its paired chromosome (Korf, 2004). SNPs that are closer together on the chromosome are more likely to be on the same DNA segment during a recombination event; SNPs that are further apart on the chromosome are more likely to be on different DNA segments during recombination (Korf, 2004). As a result, SNPs that are

closer together tend to have a higher positive correlation between their genotypes and SNPs further apart tend to have a correlation closer to zero between their genotypes.

Simulating multiple SNPs is useful for acquiring empirical results about statistical tests (Morris et al., 2019). Understanding the limitations or robustness of a statistical test is extremely important so that it can be used in a wide variety of situations. Some key situations include understanding sample size effects, calculating power and generally how well the statistical test works under different underlying populations (Morris et al., 2019). In clinical settings, sample sizes may be too small to adequately reach theoretical asymptotic distributions. Statistical tests are often developed with a well known practical sample and/or with simulated samples. One drawback with using simulated data is that the data set may not accurately represent a real-world sample. However, the advantage is that the actual population parameters are known and under direct control of the researcher.

Simulating from independent SNPs can be done by generating from the marginal distributions separately. Simulating correlated SNPs is more difficult since the correlation structure must be accounted for in the simulation. In general, there are multiple different joint probability mass functions (PMFs) that satisfy the constraints imposed by the correlation structure and the marginal distributions in a collection of correlated autosomal biallelic SNPs. A collection of correlated autosomal biallelic SNPs is more useful for researchers as it is closer to reality.

Many different methods to simulate the genome have been developed. Evolutionary based simulations follow sequences of DNA as different evolutionary processes impact them (e.g., recombination). The effect of evolutionary processes on DNA sequences are approximated by various models. Most models incorporate evolutionary

processes through an evolutionary Markovian or a hidden Markov model applied to DNA sequences (Carvajal-Rodriguez, 2010).

Some simulation methods are based solely on the binomial marginal distributions of a collection of correlated autosomal biallelic SNPs and the correlation structure. The genotypes of a collection of d correlated autosomal biallelic SNPs can be represented as a random vector $\boldsymbol{X} = (X_1, \ldots, X_d)$ with correlation matrix $\boldsymbol{\rho}_X$ and where the i^{th} component is distributed as a binomial random variable with two trials and success probability equal to the MAF of the i^{th} SNP. Madsen and Birkes (2013) and Ferrari and Barbiero (2012) improve on a simulation method described in Cario and Nelson (1997). Both methods first generate observations from the multivariate normal distribution $\boldsymbol{Z} = (Z_1, \ldots, Z_d)$ with mean vector zero, variances equal to one and correlation matrix $\boldsymbol{\rho}_Z$. Then, each component of $\boldsymbol{Z}$ is mapped to the corresponding values of the target binomial random variable as follows:

$$\begin{aligned} \Phi(Z_i) \in \left[0,\ (1-p_i)^2\right] &\implies X_i = 0 \\ \Phi(Z_i) \in \left((1-p_i)^2,\ 1-p_i^2\right] &\implies X_i = 1 \\ \Phi(Z_i) \in \left(1-p_i^2,\ 1\right] &\implies X_i = 2, \end{aligned} \tag{1.2}$$

where p_i is the MAF of the i^{th} SNP. The choice of correlation matrix for the multivariate normal dictates the pairwise correlation in the resulting discrete random vector. In most cases, the pairwise correlation in the resulting random vector is different than the pairwise correlation between components in $\boldsymbol{Z}$. The main aim of the work by Madsen and Birkes (2013) and Ferrari and Barbiero (2012) is to find a correlation matrix $\boldsymbol{\rho}_Z$ which results in the desired correlation matrix for $\boldsymbol{X}$. A similar concept is used by Montana (2005) where a random vector with Bernoulli marginal distributions

is simulated instead.

Kang and Jung (2001) describe a simulation method that first calculates a joint PMF for a vector of correlated Bernoulli random variables and then simulates using that PMF. Their method also accounts for the correlation between components. As a result, their simulation method can be used to generate from a random vector with a correlation structure where the i^{th} component is distributed as a Bernoulli random variable with success probability equal to the MAF of the i^{th} SNP. In Chapter 2, we examine how the Kang and Jung method is used to generate from a vector of correlated binomial random variables.

1.1 Organization

The main aim of this work is to simulate a collection of d correlated autosomal bial-lelic SNPs. In Chapter 2, the Kang and Jung method and the Madsen and Birkes method is described and used to generate samples of two, three and five SNPs with varying MAFs and target correlations. Summaries of findings will be provided and discussed. In Chapter 3, we define two new methods that use techniques from linear programming to handle scenarios with multiple SNPs, different marginal distributions and pairwise correlations. In Chapter 4 we compare the methods discussed in Chap-ters 2 and 3 in terms of how many confidence intervals contain the target value, the upper bound on correlation, the processing speed, the bias of the sample correlation and the empirical distribution of $\boldsymbol{X}$. We conclude in Chapter 5 with a summary of the main contributions of the thesis and discuss potential directions for future work.

Chapter 2

Simulation Methods

Two methods to simulate a collection of d correlated autosomal biallelic SNPs will be covered in this chapter. The method by Madsen and Birkes (2013) will be explained, followed the method by Kang and Jung (2001). In Section 2.4, we will examine the observed sample pairwise correlation and sample MAF of each SNP. The target or input MAFs and pairwise correlations will be varied and compared to the sample estimates.

2.1 Bounds on Pearson Correlation Coefficient

The choice of target MAFs restricts the range on the possible pairwise correlation. Given a bivariate random vector $\boldsymbol{X} = (X_1, X_2)$, the correlation between the components may be bounded by values other than negative and positive one depending on the choice of marginal distributions for X_1 and X_2 (Demirtas and Hedeker, 2011). These bounds are sometimes known as the Fréchet-Hoeffding bounds. Separately, Fréchet and Hoeffding established that there exist two joint cumulative distribution

functions (CDFs) such that the two joint CDFs would obtain the lowest and highest possible correlation between X_1 and X_2 respectively (Frechet, 1951; Hoeffding, 1994). If F_1 and F_2 are the marginal CDFs of X_1 and X_2 respectively, then the joint CDFs that obtain the lowest and highest possible correlation are defined as follows:

$$\begin{aligned} H_L(x_1, x_2) &= \max\left(F_1(x_1) + F_2(x_2) - 1, 0\right) \\ H_U(x_1, x_2) &= \min\left(F_1(x_1), F_2(x_2)\right). \end{aligned} \tag{2.1}$$

The correlation coefficients calculated from H_L and H_U are the lowest and highest possible correlations between X_1 and X_2 respectively.

In the case when X_1 and X_2 are both binomially distributed with two trials and success probability equal to p_1 and p_2 respectively, then the CDFs are $F_1(x_1) = \sum_{i=0}^{x_1} \binom{2}{i} p_1^i (1-p_1)^{2-i}$ and $F_2(x_2) = \sum_{i=0}^{x_2} \binom{2}{i} p_2^i (1-p_2)^{2-i}$. The correlation coefficient can be calculated from the joint PMF for the random vector (X_1, X_2) and the probability of each element in the support can be obtained by taking differences of the joint CDF (H_L or H_U). The joint PMF of (X_1, X_2) is used to compute the correlation coefficient as follows:

$$\begin{aligned} \rho_X &= \frac{\mathrm{E}[X_1 X_2] - \mathrm{E}[X_1]\mathrm{E}[X_2]}{\sqrt{\mathrm{Var}(X_1)\mathrm{Var}(X_2)}} \\ &= \frac{p_{11} + 2p_{21} + 2p_{12} + 4p_{22} - 4p_1 p_2}{\sqrt{4p_1(1-p_1)p_2(1-p_2)}}, \end{aligned} \tag{2.2}$$

where we denote $p_{rs} = \mathrm{P}(X_1 = r, X_2 = s)$ for $r, s \in \{0, 1, 2\}$.

The blue triangles in Figure 2.1 represents the upper bound on correlation between two binomial random variables with two trials and success probability of the first random variable equal to 0.1 (left) or 0.3 (right) and the success probability of the

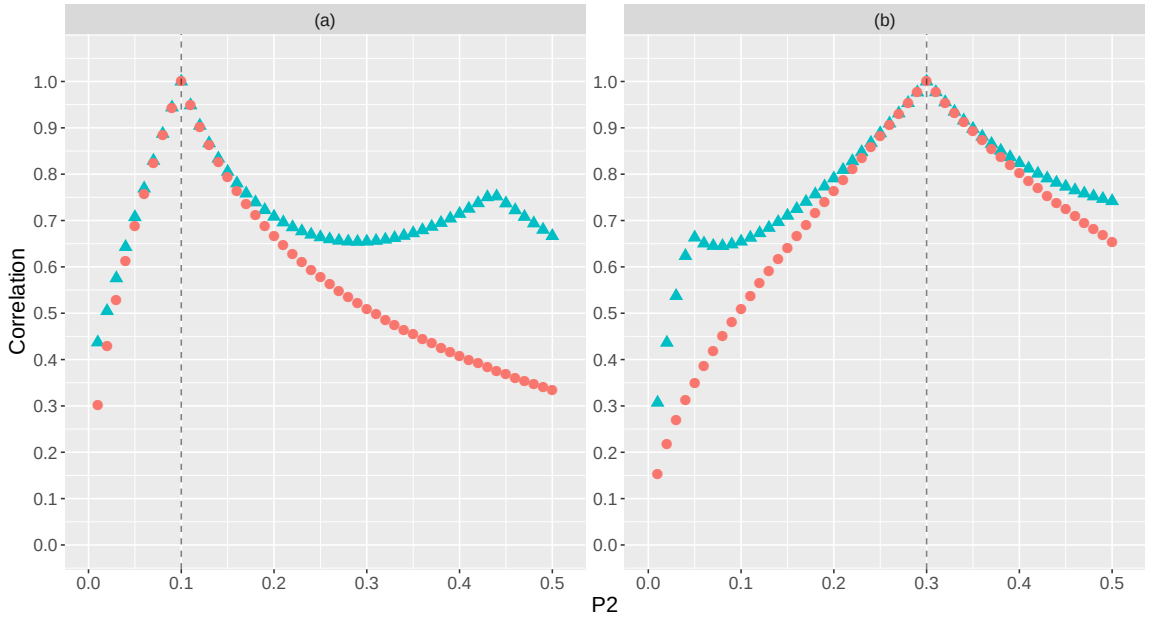

Figure 2.1: The upper bound on the correlation between (blue triangles) two binomial random variables or (red circles) two Bernoulli random variables as calculated using the Fréchet-Hoeffding bounds. The two binomial random variables have two trials and the success probability of one random variable is (a) 0.1 or (b) 0.3 and the success probability of the other is indicated on the x-axis. One of the Bernoulli random variables has success probability (a) 0.1 or (b) 0.3 and the other has success probability indicated on the x-axis. A dotted vertical line has been placed when the upper bound on correlation is one for either pair of random variables.

second random variable is indicated on the x-axis.

The upper bound on correlation between two binomial random variables only equals one when the MAFs are equal; as p_2 increases or decreases away from p_1, the upper bound on correlation decreases non-monotonically away from one. The cause of the non-monotonically decreasing upper bound on correlation when the MAFs are not equal has not been investigated in this thesis.

The red circles in Figure 2.1 represents the upper bound on correlation between two Bernoulli random variables. It was calculated similarily to the upper bound on

correlation between two binomial random variables except the marginal CDF involved is the Bernoulli CDF instead of the Bernoulli CDF. As seen in Figure 2.1, the (red circles) upper bound on correlation between a Bernoulli(p_1) random variable and a Bernoulli(p_2) random variable is smaller than the (blue triangles) upper bound on correlation between a binomial($2, p_1$) random variable and a binomial($2, p_2$) random variable. The upper bound on correlation between two Bernoulli random variables is monotonically decreasing away from one.

2.2 Multivariate Normal Based Methods

Let $\boldsymbol{X} = (X_1, \ldots, X_d)$ be a random vector with correlation matrix $\boldsymbol{\rho}_X$ where the components are marginally distributed as a binomial random variables with two trials and success probability equal to the MAF. Multivariate normal based methods start by generating from the multivariate normal distribution $\boldsymbol{Z} = (Z_1, \ldots, Z_d)$ with mean vector zero and correlation matrix $\boldsymbol{\rho}_Z$. Each component Z_i will be separately transformed to the target X_i. If X_i has CDF F_{X_i} for $i = 1, \ldots, d$, Φ is the univariate standard normal CDF and $F_{X_i}^{-1}(u) = \inf\{x : F_{X_i}(x) \geq u\}$ is the inverse CDF of X_i, then we have the following representation for $\boldsymbol{X}$:

$$\boldsymbol{X} = \left(F_{X_1}^{-1}[\Phi(Z_1)], \ldots, F_{X_d}^{-1}[\Phi(Z_d)]\right).$$

This method works for arbitrary marginal distributions, but we are only concerned with binomial marginal distributions. The transformation from Z_i to X_i for $i = 1, \ldots, d$ is explicitly shown for $X_i \sim \text{binomial}(2, p_i)$ in (1.2).

The correlation matrix for the resultant random vector is determined by the choice

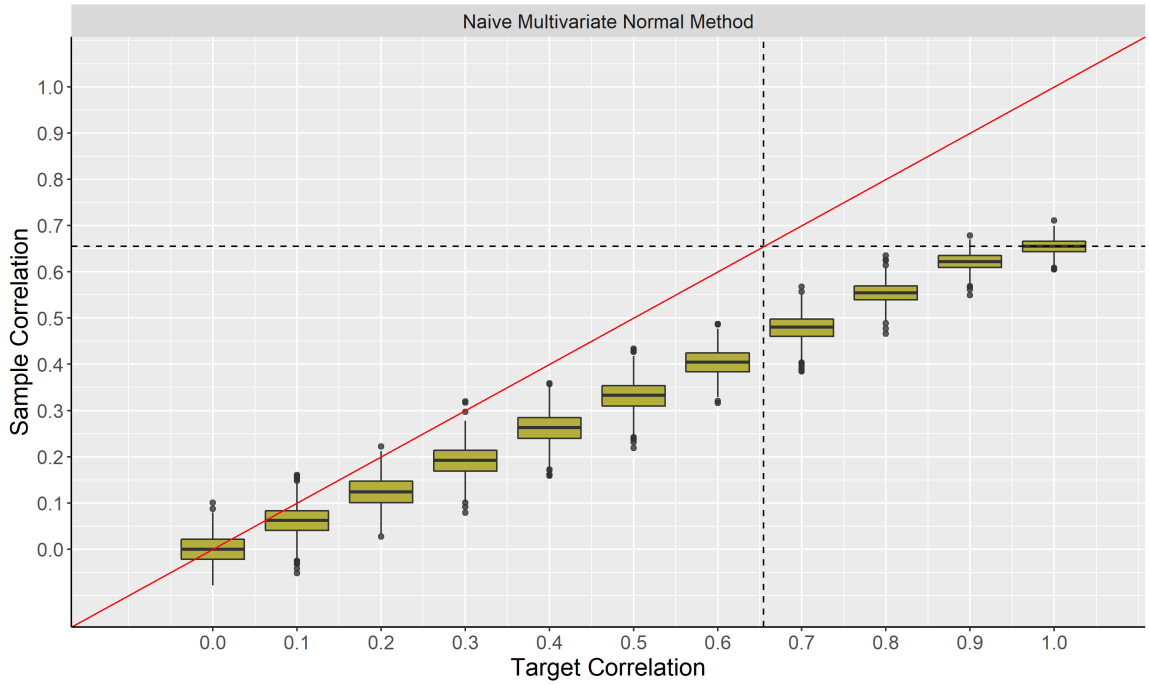

Figure 2.2: The sample and target correlation for samples generated by the naive multivariate normal method. Minor allele frequency was fixed at 0.1 for SNP1 and 0.3 for SNP2. Target correlation ranged from 0 to 1 in increments of 0.1. The upper bound on the correlation between two binomial random variables with two trials and success probability equal to 0.1 and 0.3 respectively is displayed as a dotted horizontal and dotted vertical line. Boxplots are used to display the 1000 sample correlations; each sample had sample size 1000. The line $y = x$ is in red.

of $\boldsymbol{\rho}_Z$. The naive multivariate normal based method is to set the correlation matrix $\boldsymbol{\rho}_Z$ to be equal to $\boldsymbol{\rho}_X$. This procedure does not produce samples that have the correct correlation. As seen in Figure 2.2, the sample correlations are consistently lower than the target correlation. This pattern always occurs except at a target correlation of zero or one and is due to how the support of each component is mapped from $\mathbb{R}$ to $\{0, 1, 2\}$.

As a result, $\boldsymbol{\rho}_Z$ must be chosen such that the resultant random vector has its correlation matrix equal to $\boldsymbol{\rho}_X$. The expression in (2.3) from Cario and Nelson (1997)

directly relates the correlation between components in $\boldsymbol{X}$ to correlation between components in $\boldsymbol{Z}$. Specifically, the correlation between X_i and X_j ($\boldsymbol{\rho}_{X_{ij}}$) will be a function of $\boldsymbol{\rho}_{Z_{ij}}$ only.

$$\begin{aligned} \mathrm{E}[X_i X_j] &= \mathrm{E}\left\{ F_{X_i}^{-1}[\Phi(Z_i)] \cdot F_{X_j}^{-1}[\Phi(Z_j)] \right\} \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \frac{F_{X_i}^{-1}[\Phi(z_i)] \cdot F_{X_j}^{-1}[\Phi(z_j)]}{2\pi\sqrt{1-\boldsymbol{\rho}_{Z_{ij}}^2}} \exp\left(-\frac{z_i^2 - 2z_i z_j \boldsymbol{\rho}_{Z_{ij}} + z_j^2}{2(1-\boldsymbol{\rho}_{Z_{ij}}^2)} \right) dz_i dz_j. \end{aligned} \tag{2.3}$$

There will be a total of $\binom{d}{2} = \frac{1}{2}d(d-1)$ equations that relate $\boldsymbol{\rho}_X$ to $\boldsymbol{\rho}_Z$; each representing a different pairwise correlation. There is no closed form solution given that integrating the bivariate normal CDF is involved. Cario and Nelson (1997) show that (2.3) is a continuous nondecreasing function for $\boldsymbol{\rho}_{Z_{ij}} \in [-1, 1]$. Furthermore, the authors state that any reasonable search algorithm should find the correct value for $\boldsymbol{\rho}_{Z_{ij}}$ given $\boldsymbol{\rho}_{X_{ij}}$.

When Cario and Nelson (1997) first developed this method, they used a bisection search to calculate the correct $\boldsymbol{\rho}_Z$. Ferrari and Barbiero (2012) used an iterative method to calculate the correct value where the next value of $\boldsymbol{\rho}_{Z_{ij}}$ is calculated by multiplying its current value by the ratio of $\boldsymbol{\rho}_{X_{ij}}$ over the pairwise correlation of the transformed random variable that the current $\boldsymbol{\rho}_{Z_{ij}}$ outputs. Madsen and Birkes (2013) use an alternate formulation of $\mathrm{E}[X_i X_j]$ and their method is implemented here.

The formulation for $\mathrm{E}[X_i X_j]$ that Madsen and Birkes use assumes that X_i and X_j are count-value random variables and starts from the easily verifiable result for $\mathrm{E}[X_i X_j]$ (Lo, 2019):

$$
\begin{aligned}
E[X_i X_j] &= \sum_{r=0}^{2}\sum_{s=0}^{2} \mathrm{P}(X_i > r, X_j > s) \\
&= \sum_{r=0}^{2}\sum_{s=0}^{2} \left[1 - \mathrm{P}(X_i < r) - \mathrm{P}(X_j < s) + \mathrm{P}(X_i < r, X_j < s)\right] \\
&= \sum_{r=0}^{2}\sum_{s=0}^{2} \left[1 - F_{X_i}(r) - F_{X_j}(s) + \mathrm{P}(X_i < r, X_j < s)\right] \\
&= \sum_{r=0}^{2}\sum_{s=0}^{2} \left[1 - F_{X_i}(r) - F_{X_j}(s) + \Phi_{\boldsymbol{\rho}_{Z_{ij}}}\left\{\Phi^{-1}[F_{X_i}(r)], \Phi^{-1}[F_{X_j}(s)]\right\}\right],
\end{aligned}
\tag{2.4}
$$

where $\Phi_{\boldsymbol{\rho}_{Z_{ij}}}$ is the bivariate normal CDF with mean zero and correlation equal to $\boldsymbol{\rho}_{Z_{ij}}$ and Φ^{-1} is the inverse of the univariate standard normal CDF. Equation (2.4) can be solved in R using the `optimize` function, one of many tools available in R, for the unknown $\boldsymbol{\rho}_{Z_{ij}}$ and subsequently used in the correlation matrix for the multivariate normal. The `optimize` function can be used to solve the equation by calculating the $\boldsymbol{\rho}_{Z_{ij}}$ that minimizes the squared difference between the value of $E[X_i X_j]$ and right hand side of (2.4). Once each entry in the correlation matrix $\boldsymbol{\rho}_Z$ is known, then observations can be generated from the multivariate normal and then transformed to have the correct target marginal distributions.

2.3 Bernoulli Based Method

Another way of generating observations that satisfy the constraints on $\boldsymbol{X}$ is to add observations from two independent and identically distributed random vectors $\boldsymbol{Y}_1 = (Y_{11}, \ldots, Y_{1d})$ and $\boldsymbol{Y}_2 = (Y_{21}, \ldots, Y_{2d})$ where the i^{th} component of $\boldsymbol{Y}_1$ and $\boldsymbol{Y}_2$ is distributed as a Bernoulli random variable with success probability equal to the

MAF of the i^{th} SNP and where $\boldsymbol{Y}_1$ and $\boldsymbol{Y}_2$ each have correlation matrix $\boldsymbol{\rho}_X$. Then, $\boldsymbol{X}$ is defined by $X_i = Y_{1i} + Y_{2i}$ for $i = 1, \ldots, d$.

The method by Kang and Jung (2001) can be used to calculate the PMF for $\boldsymbol{Y}$. Probabilities are assigned to every element in the support of $\boldsymbol{Y}$ based on the linear constraints imposed by the marginal distributions and correlation structure of $\boldsymbol{Y}$. Once we know the probability each element in the support of $\boldsymbol{Y}$, observations are generated based on a partition of the interval $[0, 1]$ and random observations generated from the standard uniform distribution. The interval $[0, 1]$ is partitioned into 2^d non-overlapping intervals that span $[0, 1]$; each interval is associated with one element in the support of $\boldsymbol{Y}$. Observations of $\boldsymbol{Y}$ are generated by mapping an observation from the standard uniform to elements in the support of $\boldsymbol{Y}$ according to the partition of $[0, 1]$ described earlier.

The linear constraints imposed on the probability of each element in the support of $\boldsymbol{Y}$ are from the marginal distributions of $\boldsymbol{Y}$ and correlation matrix $\boldsymbol{\rho}_X$. For $i = 1, \ldots, d$, the i^{th} component Y_i is distributed as a Bernoulli(p_i) random variable and imposes the following linear constraints in (2.5). One of the constraints in (2.5) is redundant since the probabilities of each element in the support of $\boldsymbol{Y}$ must sum to one.

$$\begin{aligned}
\mathrm{P}(Y_i = 0) &= \sum_{k_1=0}^{1} \cdots \sum_{k_{i-1}=0}^{1} \sum_{k_{i+1}=0}^{1} \cdots \sum_{k_d=0}^{1} \mathrm{P}[\boldsymbol{Y} = (k_1, \ldots, k_{i-1}, 0, k_{i+1}, \ldots, k_d)] = 1 - p_i \\
\mathrm{P}(Y_i = 1) &= \sum_{k_1=0}^{1} \cdots \sum_{k_{i-1}=0}^{1} \sum_{k_{i+1}=0}^{1} \cdots \sum_{k_d=0}^{1} \mathrm{P}[\boldsymbol{Y} = (k_1, \ldots, k_{i-1}, 1, k_{i+1}, \ldots, k_d)] = p_i
\end{aligned} \tag{2.5}$$

Simple algebra shows that the only unknown term for $\boldsymbol{\rho}_{Y_{ij}}$ is $E[Y_i Y_j]$ given the target marginal distributions and pairwise correlations. For $i = 1, \ldots, d$, $j = 1, \ldots, d$ and

$i \neq j$, we have the following expression for $E[Y_iY_j]$:

$$\begin{aligned} \boldsymbol{\rho}_{Y_{ij}} &= \frac{\mathrm{E}[Y_iY_j] - \mathrm{E}[Y_i]\mathrm{E}[Y_j]}{\sqrt{\mathrm{Var}(Y_i)\mathrm{Var}(Y_j)}} \\ \implies \mathrm{E}[Y_iY_j] &= \boldsymbol{\rho}_{Y_{ij}} \cdot \sqrt{\mathrm{Var}(Y_i)\mathrm{Var}(Y_j)} + \mathrm{E}[Y_i]\mathrm{E}[Y_j] \\ &= \boldsymbol{\rho}_{Y_{ij}} \cdot \sqrt{p_i(1-p_i)p_j(1-p_j)} + p_ip_j. \end{aligned} \tag{2.6}$$

In this case, the expression for $E[Y_iY_j]$ can be further written as follows:

$$\begin{aligned} \mathrm{E}[Y_iY_j] &= \sum_{k_i=0}^{1}\sum_{k_j=0}^{1} k_ik_j\mathrm{P}(Y_i = k_i, Y_j = k_j) \\ &= \sum_{k_i=0}^{1}\sum_{k_j=0}^{1} k_ik_j \cdot \\ &\quad \left(\sum_{k_1=0}^{1}\cdots\sum_{k_{i-1}=0}^{1}\sum_{k_{i+1}=0}^{1}\cdots\sum_{k_{j-1}=0}^{1}\sum_{k_{j+1}=0}^{1}\cdots\sum_{k_d=0}^{1}\mathrm{P}[\boldsymbol{Y} = (k_1,\ldots,k_i,\ldots,k_j,\ldots k_d)]\right) \\ &= \sum_{k_1=0}^{1}\cdots\sum_{k_i=0}^{1}\cdots\sum_{k_j=0}^{1}\cdots\sum_{k_d=0}^{1} k_ik_j\mathrm{P}[\boldsymbol{Y} = (k_1,\ldots,k_i,\ldots,k_j,\ldots k_d)]. \end{aligned} \tag{2.7}$$

The linear constraints derived from (2.5), (2.6) and (2.7) are all functions of probabilities for elements in the support for $\boldsymbol{Y}$. Specifically, they are summations of the probability of particular elements in the support of $\boldsymbol{Y}$. The probabilities of each element in the support of $\boldsymbol{Y}$ can be explicitly written as in (2.8) when $\boldsymbol{Y}$ has two

components:

$$\begin{aligned} q_{10} + q_{11} &= p_1 \\ q_{01} + q_{11} &= p_2 \\ q_{11} &= \rho_X \cdot \sqrt{p_1(1-p_2)p_2(1-p_2)} + p_1 p_2 \\ q_{00} + q_{01} + q_{10} + q_{11} &= 1, \end{aligned} \tag{2.8}$$

where we denote $q_{tu} = \mathrm{P}[\boldsymbol{Y} = (t, u)]$ for $t, u \in \{0, 1\}$.

If the probability of each element in the support of $\boldsymbol{Y}$ satisfies the constraints in (2.8), then the joint PMF of $\boldsymbol{Y}$ can be used to simulate a random vector with Bernoulli marginal distributions and a specified correlation structure. In the case of two components, the PMF for $\boldsymbol{Y}$ is unique as there are four constraints and four parameters. For $d \geq 3$ components, there are 2^d elements in the support of $\boldsymbol{Y}$, $d + 1$ linear constraints from the marginal distributions and $\binom{d}{2} = \frac{1}{2}d(d-1)$ linear constraints from the pairwise correlations. This is a system of linear equations with 2^d parameters and $\frac{1}{2}d^2 + \frac{1}{2}d + 1$ linear constraints. This is an underdetermined system since there are more parameters than linear constraints. Therefore, no unique solution exists and some parameters will be free to vary, meaning there is more than one joint PMF that satisfies the linear constraints. The Kang and Jung method used to calculate the joint PMF for $\boldsymbol{Y}$ requires additional arbitrary linear constraints for $d \geq 3$ as it requires a unique solution and none exist with only the constraints from the marginal distributions and correlation matrix. As a result, the Kang and Jung method will not work without additional information for $d \geq 3$.

Adding k copies of $\boldsymbol{Y}$ will result in a random vector with Binomial(k, p_i) for $i = 1, \ldots, d$ and correlation matrix $\boldsymbol{\rho}_X$. The resulting marginal distributions are all

binomials because, component wise, it is the same as adding independent and identically distributed Bernoulli random variables. For d correlated autosomal biallelic SNPs, we take k equal to two. Overall, observations are generated from $\boldsymbol{X}$ by calculating a PMF for $\boldsymbol{Y}$ and adding two independently generated observations from $\boldsymbol{Y}$.

2.4 Implementation and Discussion

Two methods were chosen for implementation: the Madsen and Bikes method and the Kang and Jung method. The number of SNPs, choice of MAFs and pairwise correlations were varied. Samples were generated for random vectors with two, three and five SNPs. Two SNPs represents the simplest scenario; three and five SNPs represents a scenario with multiple correlated SNPs where the number of different parameter combinations is still manageable. In the two SNP scenario, SNP1 had a MAF of 0.1 and SNP2 had three possible values for MAF: $0.01, 0.1$ and 0.3. Target correlation between the two SNPs had six possible values: $0, 0.2, 0.4, 0.6, 0.8, 1$. One thousand samples, each with sample size $n = 1000$, were generated for each combination of MAF and target correlation for the scenario with two SNPs. If the target correlation violated the upper bound on correlation, then samples are not generated in our implementation of the Kang and Jung method; samples are generated in our implementation of the Madsen and Birkes method due to error correcting code that effectively rounds the target correlation to the upper bound on correlation.

The error correcting code ensures that the Madsen and Birkes method calculates a positive definite matrix from the target correlation matrix. The Madsen and Birkes method uses the multivariate normal distribution which requires a positive definite

matrix. However, the correlation matrix used in the multivariate normal is calculated one entry at a time. As a result, there is no guarantee that the calculated correlation matrix will be positive definite. Madsen and Birkes (2013) suggest checking the calculated correlation matrix, and changing any small negative eigenvalues to be positive. In our implementation, the error correcting code first factored the calculated correlation matrix into the form $Q\Lambda Q^{-1}$ where Q is a matrix where each eigenvector is placed along a column and Λ is a diagonal matrix whose non-zero entries are the eigenvalues associated to the eigenvector with the same column index in Q. Then, a new matrix Λ_{adj} was constructed as copy of Λ where any negative eigenvalues were set to 10^{-5}. Finally, the error correcting code replaced the calculated correlation matrix with $Q\Lambda_{adj}Q^{-1}$. An error correcting code was not included in any other method due to time constraints.

In Figure 2.3, the median of the sample correlation is visually close to the target correlation and the IQR is approximately 0.1. This indicates that the majority of the sample correlations fall closely around the target correlation and is evidence that the sample correlations are on average around where they should be. As the target correlation approaches its upper bound, the IQR visibly decreases for samples generated by the Madsen and Birkes method or the Kang and Jung method.

In Figure 2.3, there are many sample correlations that are above the dotted horizontal line which represents the upper bound on correlation for the respective method. Sample correlations can exceed the upper bound on correlation since the empirical marginal distributions of the sample may not perfectly match the marginal binomial distributions specified. Therefore, the sample correlation is not expected to adhere exactly to the upper bound on the correlation between the two random variables.

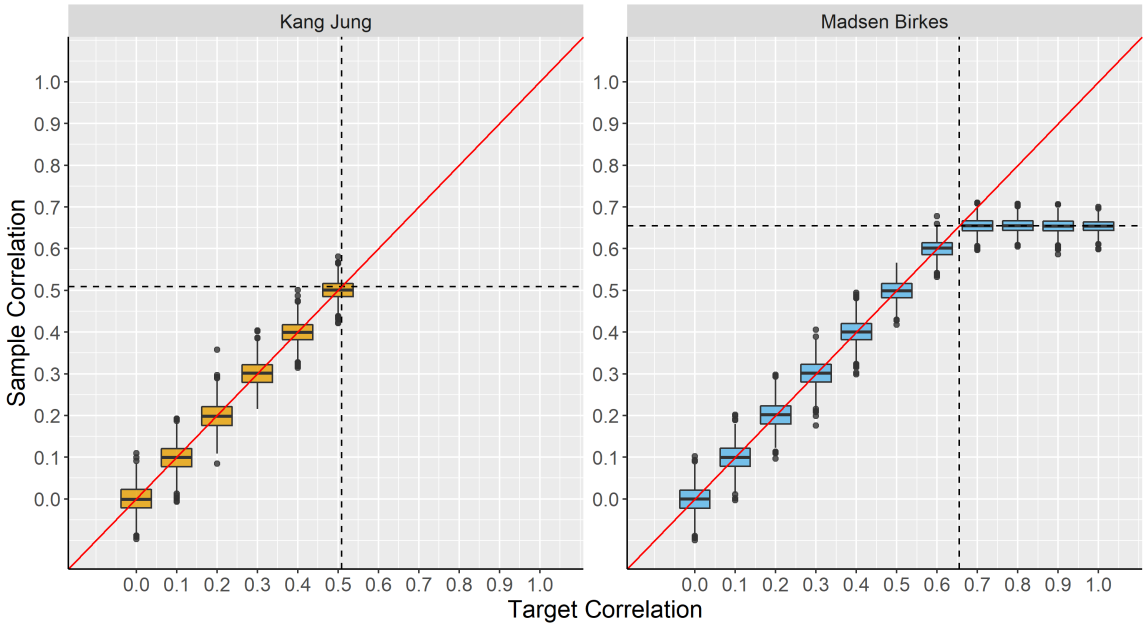

Figure 2.3: The sample and target correlation for samples generated by the Madsen and Birkes method and the Kang and Jung method. Minor allele frequency was fixed at 0.1 for SNP1 and 0.3 for SNP2. Target correlation ranged from 0 to 1 in increments of 0.1. The upper bound on the correlation between a Bernoulli(0.1) random variable and a Bernoulli(0.3) random variable is displayed as a dotted line in the graph on the left. The upper bound on the correlation between a binomial(2, 0.1) random variable and a binomial(2, 0.3) random variable is displayed as a dotted line in the graph on the right. Boxplots are used to display the 1000 sample correlations; each sample had sample size 1000. The red line is the linear curve with a slope of 1 that passes through the origin.

The Kang and Jung method does not produce a sample with target correlation of 0.6 or higher since the upper bound on the correlation between two Bernoulli random variables with success probability 0.1 and 0.3 is 0.5092. The Kang and Jung method, as implemented here, did not produce any samples if the bound on correlation is exceeded whereas the Madsen and Birkes method effectively rounded the target correlation to the upper bound if it was exceeded. Rounding target correlations to the upper bound can be implemented in the future for the Kang and Jung method.

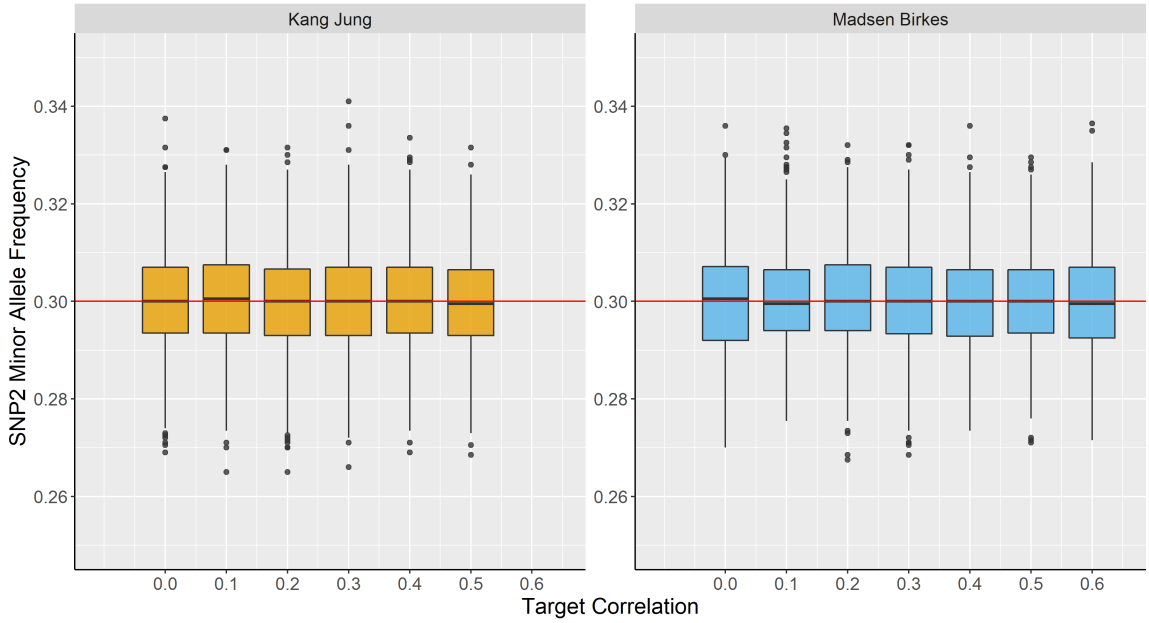

Figure 2.4: The sample MAF for SNP2 and target correlation for samples generated by the Madsen and Birkes method and the Kang and Jung method. Target MAF was fixed at 0.1 for SNP1 and 0.3 for SNP2. Target correlation ranged from 0 to 1 in increments of 0.1. Boxplots are used to display the 1000 sample MAFs; each sample had 1000 observations. The red line is the target MAF for SNP2, 0.3. The x-axis is truncated at 0.6 since the upper bound on correlation between a binomial(2, 0.1) random variable and a binomial(2, 0.6) random variable is 0.6547 when rounded to four decimal places.

In Figure 2.4, the boxplots for the sample MAF all included the target MAF and the IQR is less than 0.015 in all cases. Visually, the sample median and the target value (red line) are very close to each other. Similar results are also seen for SNP1. This is evidence that the samples generated have marginal distributions that are the target marginal distributions. Since every method displays this behavior, it is evidence that all of them have the correct marginal distributions on average.

Similar results are seen when the number of SNPs increase to three or five. The

boxplots of the sample MAF or correlation visually include the target MAF or correlation and the sample median is to the target value. One difference is that there are more bounds involved on the correlation between two components since many more pairs of components exist.

The Kang and Jung method does not extend well to multiple SNPs. A unique solution only exists in the case of two SNPs. For three or more SNPs, Kang and Jung add arbitrary linear constraints until an exact solution can be found. As the number of SNPs increases, the number of additional arbitrary constraints needed increases exponentially. Manually adding the required number of additional constraints is not feasible to do for an arbitrary number of d SNPs and would require additional coding to do automatically. The additional constraints manually added in our simulations for three or five SNPs are listed in Appendix A.1.

Overall, in the two SNP scenario, the Kang and Jung method and the Madsen and Birkes method produce samples whose sample correlation and sample MAF are, on average, close to the target correlation and target MAF. The Kang and Jung method does have a smaller upper bound on correlation compared to the Madsen and Birkes method and does not work without additional linear constraints for more than two SNPs. The Madsen and Birkes method can easily extend to more than two SNPs and has a higher upper bound on correlation than the Kang and Jung method. If the target correlation is below the upper bound, and only two SNPs are being simulated, then both methods work about the same. For more than two SNPs, the Madsen and Birkes method is the better option of the two.

Chapter 3

A Simulation Method Using Linear Optimization

In this chapter we propose a new method to simulate a collection of d correlated autosomal biallelic SNPs. Our method is similar to the Kang and Jung method from Chapter 2. One of the biggest differences is that our novel method uses the simplex method from the field of linear programming to solve a system of equations whereas the method by Kang and Jung (2001) uses an algebraic solver to solve the system of equations. Our novel method is also not limited to unique solutions and therefore does not require the additional arbitrary linear constraints that the method by Kang and Jung (2001) does when the number of SNPs exceeds two. Our novel method will be explained in Section 3.1. We examine the distributions of the sample MAF and sample pairwise correlation under different target MAF, number of SNPs, and target pairwise correlations in Section 3.2 and then end the chapter with a discussion in Section 3.3.

3.1 Our Novel Methods

Two novel methods are introduced in this section. In Sections 3.1.1 to 3.1.3, we explain how the novel binomial method calculates a possible joint PMF for $\boldsymbol{X}$. In Section 3.1.4 we explain how the novel Bernoulli method calculates a possible joint PMF for a Bernoulli random vector in order to generate observations that satisfy the constraints on $\boldsymbol{X}$. Both methods are based on the same premise of using techniques from linear programming to calculate a possible joint PMF for a random vector.

3.1.1 System of Equations

In order to calculate a possible joint PMF for $\boldsymbol{X}$, our novel method uses the system of equations derived from the specified marginal distributions and correlation matrix. In order to model a collection of d correlated autosomal biallelic SNPs, every component will have a binomial distribution. However, our method is not limited to binomial marginal distributions; its limited to whether the marginal distribution can be described by linear constraints on the joint PMF. The system of equations from $\boldsymbol{X}$ when $d = 2$ will be covered in detail and followed by the extension to $d \geq 3$.

In the case of $d = 2$, we have the following linear constraints from the specified binomial marginal distributions:

$$\begin{aligned} &\mathrm{P}(X_i = 0) = (1 - p_i)^2 && \text{for } i = 1, 2 \\ &\mathrm{P}(X_i = 1) = 2p_i(1 - p_i) && \text{for } i = 1, 2 \\ &\mathrm{P}(X_i = 2) = p_i^2 && \text{for } i = 1, 2 \end{aligned} \tag{3.1}$$

where p_1 and p_2 are the success probabilities for the binomial distributions for X_1

and X_2 respectively. One of the equations in (3.1) is redundant since the probability of all of the elements must sum to one and there are only three elements in the support of X_1 and X_2. There are a total of five linear constraints since X_1 and X_2 each contribute two and the probabilities must sum to one. If we choose to use the first two constraints in (3.1), then the non-redundant linear constraints can be written in terms of the probabilities of elements in the support of $\boldsymbol{X}$:

$$\begin{aligned}
(1-p_1)^2 &= p_{00} + p_{01} + p_{02} \\
2p_1(1-p_1) &= p_{20} + p_{21} + p_{22} \\
(1-p_2)^2 &= p_{00} + p_{10} + p_{20} \\
2p_2(1-p_2) &= p_{01} + p_{11} + p_{21},
\end{aligned} \tag{3.2}$$

where we denote $p_{rs} = \mathrm{P}[\boldsymbol{X} = (r,s)]$ for $r, s \in \{0,1,2\}$. The specified correlation between X_1 and X_2 also provides a linear constraint that can be written in terms of the p_{ij} as

$$\begin{aligned}
\rho_X &= \frac{\mathrm{Cov}(X_1, X_2)}{\sqrt{\mathrm{Var}[X_1]\mathrm{Var}[X_2]}} \\
\implies \rho_X &= \frac{\mathrm{E}[X_1X_2] - 4p_1p_2}{2\sqrt{p_1(1-p_1)p_2(1-p_2)}} \\
\implies 2\rho_X\sqrt{p_1(1-p_1)p_2(1-p_2)} + 4p_1p_2 &= \mathrm{E}[X_1X_2] \\
\implies 2\rho_X\sqrt{p_1(1-p_1)p_2(1-p_2)} + 4p_1p_2 &= p_{11} + 2p_{12} + 2p_{21} + 4p_{22},
\end{aligned} \tag{3.3}$$

where ρ_X is the correlation between X_1 and X_2. Overall, the six linear constraints from the specified binomial marginal distributions and the specified correlation are

as follows:

$$\begin{aligned}
(1-p_1)^2 &= p_{00} + p_{01} + p_{02} \\
2p_1(1-p_1) &= p_{20} + p_{21} + p_{22} \\
(1-p_2)^2 &= p_{00} + p_{10} + p_{20} \\
2p_2(1-p_2) &= p_{01} + p_{11} + p_{21} \\
2\rho_X\sqrt{p_1(1-p_1)p_2(1-p_2)} + 4p_1p_2 &= p_{11} + 2p_{12} + 2p_{21} + 4p_{22} \\
1 &= \sum_{r=0}^{2}\sum_{s=0}^{2} p_{rs},
\end{aligned} \tag{3.4}$$

where ρ_X, p_1 and p_2 are all inputs to the simulation method and therefore known quantities. Any assignment of probabilities to the elements in the support of $\boldsymbol{X}$, such that the equations in (3.4) are satisfied, is a possible joint PMF for the random vector $\boldsymbol{X}$. However, the system of equations is underdetermined since there are nine parameters and only six linear constraints. As a result, the system either has no solution or an infinite number of solutions. If there is no solution, then the specification of input parameters is infeasible. If there is an infinite number of solutions, then there is an uncountable number of joint PMFs which satisfies the constraints for $\boldsymbol{X}$. In our method, we propose using techniques from linear programming to solve an underdetermined system of equations in order to assign a probability to each element in the support of $\boldsymbol{X}$.

3.1.2 Linear Programming

The field of linear programming involves solving optimization problems as well as initial value problems. Finding a solution to the system of equations in (3.4) is an

initial value problem which is a common first step for many optimization problems. In fact, the initial value problem can be changed to a specific optimization problem, where all of the initial values are known by construction and the solution to the specific optimization problem provides the initial values for the original optimization problem. An optimization problem consists of a set of parameters (e.g. p_{00}, also known as decision variables), a system of equations (constraints) and an objective function to either maximize or minimize (Rader, 2010). Calculating a solution to the system of equations in (3.4) is an initial value problem which can be converted to an optimization problem by adding additional parameters, $a_1, \ldots, a_6$, and an objective function as shown in (3.5).

Objective Function

$$\text{minimize: } a_1 + a_2 + a_3 + a_4 + a_5 + a_6$$

Linear Constraints

$$\begin{aligned}
p_{00} + p_{01} + p_{02} + a_1 &= (1 - p_1)^2 \\
p_{10} + p_{11} + p_{12} + a_2 &= 2(1 - p_1)p_1 \\
p_{00} + p_{10} + p_{20} + a_3 &= (1 - p_2)^2 \\
p_{01} + p_{11} + p_{21} + a_4 &= 2(1 - p_2)p_2 \\
p_{11} + 2p_{12} + 2p_{21} + 4p_{22} + a_5 &= \rho_X \cdot 2\sqrt{p_1(1 - p_1)p_2(1 - p_2)} + 4p_1p_2 \\
\left[\sum_{r=0}^{2}\sum_{s=0}^{2} p_{rs}\right] + a_6 &= 1 \\
p_{00}, p_{01}, p_{02}, p_{10}, p_{11}, p_{12}, p_{10}, p_{21}, p_{22}, a_1, a_2, a_3, a_4, a_5 &\geq 0
\end{aligned} \tag{3.5}$$

The optimization problem is the minimization of the sum $\sum_{i=1}^{6} a_i$ subject to the

listed linear constraints in (3.5). If the minimized sum of $\sum_{i=1}^{6} a_i$ is zero, then all of the additional parameters, $a_1, \ldots, a_6$, must also be zero since they are all non-negative. As a result, the linear constraints in (3.5) reduce to the constraints in (3.4) provided by the target correlation matrix and marginal distributions. Therefore, a solution to the optimization problem in (3.5), where the minimized sum is zero, will satisfy the system of equations in (3.4).

The additional parameters, $a_1, \ldots, a_6$ are added in order to provide an initial solution to the optimization problem. The initial solution is required by most linear optimization problem solvers and is found by setting $p_{rs} = 0$ for $r, s \in \{0, 1, 2\}$ and a_i for $i = 1, \ldots, 6$, are set equal to the right hand side of their respective equation in (3.5). Minimization of the objective function, $\sum_{i=1}^{6} a_i$, was carried out using the R package lpSolve which is an interface that calls on functions written in C/C++ to solve linear optimization problems such as (3.5); it uses a revised simplex method to solve the minimization problem (Berkelaar et al., 2020). The simplex method is an iterative process where parameters are chosen and changed until some stopping criterion has been reached. Here, lpSolve chose parameters based on the Devex pricing algorithm from Berkelaar et al. (2020). In general, the Devex pricing algorithm chooses parameters based on an approximation to picking the steepest edge in the space of feasible solutions.

3.1.3 Generalizing to d SNPs

The system of equations in (3.4) can be generalized for an arbitrary number of components or SNPs. For d components, we assume the i^{th} marginal distribution contributes

the following non-redundant linear constraints:

$$\mathrm{P}(X_i = 0) = (1 - p_i)^2 \qquad \mathrm{P}(X_i = 1) = 2p_i(1 - p_i). \tag{3.6}$$

The linear constraints can be written as a sum involving the probabilities of elements in the support of $\boldsymbol{X}$:

$$\begin{aligned}(1 - p_i)^2 &= \sum_{k_1=0}^{2} \dots \sum_{k_{i-1}=0}^{2} \sum_{k_{i+1}=0}^{2} \dots \sum_{k_d=0}^{2} \mathrm{P}[\boldsymbol{X} = (k_1, \dots, k_{i-1}, 0, k_{i+1}, \dots, k_d)] \\ 2p_1(1 - p_i) &= \sum_{k_1=0}^{2} \dots \sum_{k_{i-1}=0}^{2} \sum_{k_{i+1}=0}^{2} \dots \sum_{k_d=0}^{2} \mathrm{P}[\boldsymbol{X} = (k_1, \dots, k_{i-1}, 1, k_{i+1}, \dots, k_d)].\end{aligned} \tag{3.7}$$

The linear constraint from the correlation between X_i and X_j where $i = 1, \dots, d-1$ and $j = i+1, \dots, d$ is:

$$\begin{aligned}\mathrm{Cor}(X_i, X_j) &= \frac{\mathrm{Cov}(X_i, X_j)}{\sqrt{\mathrm{Var}[X_i]\mathrm{Var}[X_j]}} \\ &= \frac{\mathrm{E}[X_i X_j] - 4p_i p_j}{2\sqrt{p_i(1 - p_i)p_j(1 - p_j)}},\end{aligned} \tag{3.8}$$

where $\mathrm{E}[X_i X_j]$ is defined in (2.7). The linear constraint from the summation of all probabilities is the following:

$$1 = \sum_{k_1=0}^{2} \dots \sum_{k_d=0}^{2} \mathrm{P}[\boldsymbol{X} = (k_1, \dots k_d)]. \tag{3.9}$$

The system of equations can be converted to a minimization problem in the same way as we did for 2 SNPs in Section 3.1.2. An additional parameter a_i is be added

to each linear constraint, every parameter must be non-negative and the optimization problem is the minimization of $\sum_{i=1}^{\frac{1}{2}d^2+\frac{3}{2}d+1} a_i$. Any solution to the optimization problem where the objective function is zero gives a possible joint PMF for $\boldsymbol{X}$.

3.1.4 Novel Bernoulli Method

Recall that the Kang and Jung method simulates observations that satisfy the constraints on $\boldsymbol{X}$ by first calculating a possible joint PMF for random vector, $\boldsymbol{Y}$, with correlation matrix $\boldsymbol{\rho}_X$ and where the i^{th} component of $\boldsymbol{Y}$ is distributed as a Bernoulli random variable with success probability equal to the MAF of the i^{th} SNP. In order to generate one observation that satisfies the constraints on $\boldsymbol{X}$, two independent observations are generated from $\boldsymbol{Y}$ and added together. The algebraic solver used in the Kang and Jung method can be replaced by the simplex method. This would enable the method to extend to more than two SNPs without requiring manual addition of arbitrary linear constraints. In the case of two SNPs, there is a unique solution for the random variable with Bernoulli marginal distributions. In this case, both methods were tested and we verified that they result in the same joint PMF for $\boldsymbol{Y}$. In the case of more than two SNPs, the methods differ as the Kang and Jung method requires additional linear constraints whereas our novel Bernoulli method does not.

3.2 Sample Correlation and Sample MAF

Using our novel binomial method, samples with two, three and five SNPs were generated. The novel Bernoulli method was not used since it would have the same results as the Kang and Jung method covered in Chapter 2 since the novel Bernoulli method

calculated the same possible joint PMF for $\boldsymbol{X}$ as the Kang and Jung method does for $d = 2$. One thousand samples were generated for each unique combination of input parameters - number of SNPs, MAFs and target correlation matrix; each sample had $n = 1000$ observations. In the case of two SNPs, the the input parameters tested are listed in (3.10) where p_i is the MAF of the i^{th} SNP and $\boldsymbol{\rho}_{X_{ij}}$ is the target correlation between the i^{th} and j^{th} SNP. The input parameters for three and five SNPs are listed in (3.11) and (3.12) respectively.

2 SNPs

$$\begin{aligned} p_1 &= 0.1 \\ p_2 &\in (0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5) \\ \rho_{X_{12}} &\in \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\} \end{aligned} \tag{3.10}$$

3 SNPs

$$\begin{aligned} (p_1, p_2, p_3) &\in \{(0.1, 0.1, 0.1), (0.1, 0.15, 0.05)\} \\ \boldsymbol{\rho}_{X_{12}} &\in \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\} \\ \boldsymbol{\rho}_{X_{13}}, \boldsymbol{\rho}_{X_{23}} &\in \{0, 0.2, 0.8\} \end{aligned} \tag{3.11}$$

5 SNPs

$$\begin{aligned} (p_1, p_2, p_3, p_4, p_5) &\in \{(0.1, 0.1, 0.1, 0.1, 0.1), (0.1, 0.15, 0.05, 0.3, 0.025)\} \\ \boldsymbol{\rho}_{X_{12}} &\in \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\} \\ \boldsymbol{\rho}_{X_{kl}} &\in \{0, 0.2, 0.8\} \text{ for } k = 1, \ldots, 4, \;\; l = 3, 4, 5, \;\; k < l. \end{aligned} \tag{3.12}$$

Only combinations of parameters that gave a positive definite correlation matrix and did not violate the correlation bounds mentioned in Chapter 2 were used in the simulations.

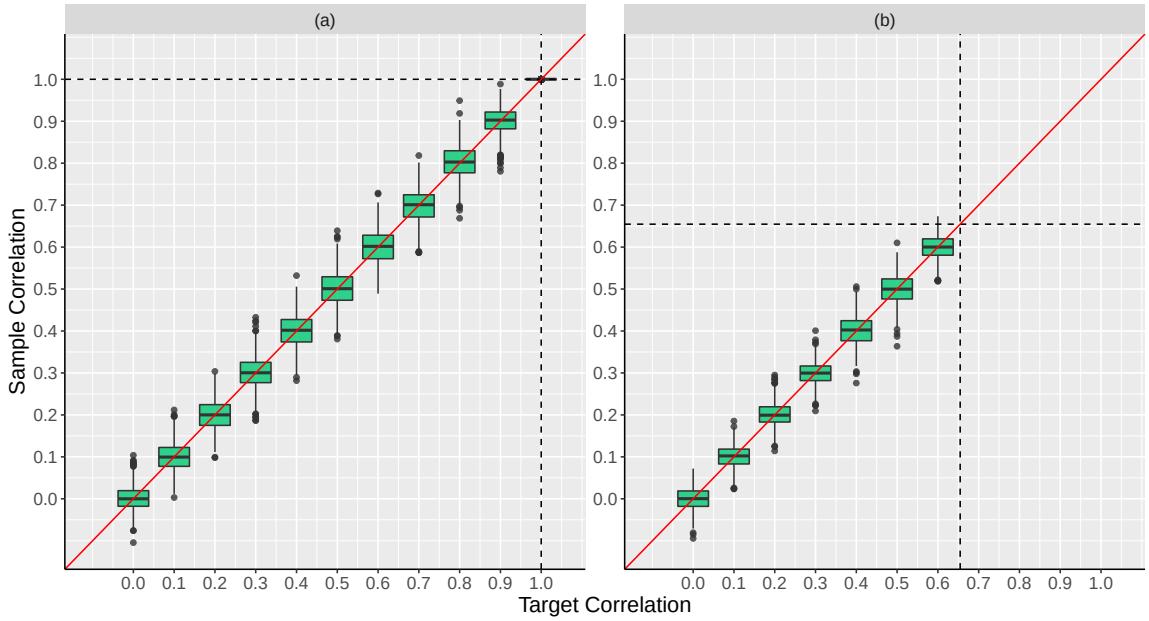

Figure 3.1: The sample correlation is plotted against the target correlation for samples generated using our novel binomial method. The target correlation was changed in increments of 0.1 and ranged from (a) 0 to 1 or from (b) 0 to 0.6. The MAFs are either (a) 0.1 and 0.1 or (b) 0.1 and 0.3. Each boxplot consists of 1000 sample correlations; each sample correlation comes from a sample of size 1000. The $y = x$ line is in red; the dotted vertical and horizontal lines occur at the upper bound on correlation between two random variables that are both binomially distributed with two trials and success probabilities equal to the two MAFs of the respective graph.

Figure 3.1 shows the sample correlation between the two SNPs for a subset of the data. Panel (a) shows the sample correlation between two SNPs with MAFs of 0.1 and 0.1; panel (b) shows the sample correlation between two SNPs with MAFs of 0.1 and 0.3. The MAF values were chosen to show a case where the upper bound on correlation is one and a case where it is not one. Visually, the median sample correlation of each boxplot is approximately equal to the target correlation for all different target correlations and MAFs shown. The whiskers on the boxplot indicate that sample correlations in the tail end of the distribution can be approximately 0.1

Target Correlation	Target MAF	Median	IQR	Q1	Q3
0.0	0.1	0.1000	0.0085	0.0873	0.1127
0.4	0.1	0.1000	0.0090	0.0865	0.1135
0.8	0.1	0.0995	0.0090	0.0860	0.1130
1.0	0.1	0.1000	0.0085	0.0873	0.1127
0.0	0.3	0.2995	0.0140	0.2785	0.3205
0.4	0.3	0.3007	0.0140	0.2797	0.3218

Table 3.1: Median, IQR, Q1, and Q2 of the Sample MAF for SNP2 in case of 2 SNPs total. SNP1 has a MAF of 0.1, the median, IQR, Q1 and Q3 are calculated on 1000 samples where each sample has 1000 observations. Median, IQR, Q1 and Q3 have been rounded to 4 decimal places. Samples were generated by our novel binomial method.

above and below the target value.

When the target correlation is one, the sample correlation will also be one. A target correlation of one is only possible when the MAFs are the same. In fact, a target correlation of one implies the joint PMF must satisfy $\mathrm{P}(X_1 = X_2) = 1$ and there is a unique joint PMF which satisfies this constraint and has the correct marginal distributions (which must also be identical for a correlation of one to be possible). The joint PMF that is calculated by our method will only have non-zero probabilities for elements (0,0), (1,1) and (2,2) in the support of $\boldsymbol{X}$. If some other element in the support of $\boldsymbol{X}$ has a non-zero probability, then the correlation calculated will not be one, thereby violating one of the linear constraints in this scenario.

If the MAFs and target correlations do not violate any of the upper bounds on correlation or the positive definite correlation matrix, then the subset of data shown in Figure 3.1 is indicative of the rest of the samples generated described in the beginning of the results section. The MAF, number of SNPs and target correlation does not drastically affect how close the median is to the target value. There are outliers that

are 0.1 above and below the target value.

Table 3.1 shows that the median of the sample MAFs are within 0.0015 of the target MAFs and the IQR is between 0.0025 and 0.016. While Table 3.1 only shows a subset of the total number of input parameter combinations tested, the range of median and IQR hold for all data generated. All of these values indicate that the sample MAF median is close to the target MAF and is evidence that the samples generated by our method have the correct marginal distribution.

3.3 Discussion

Currently, there is a practical limitation to our method; as the number of SNPs increase, the number of linear constraints grows quadratically and the number of parameters currently stored grows exponentially. Each new component is another marginal distribution that contributes two new linear constraints for a total of $2d$ constraints from the marginal distributions. Each new component also adds to the number of pairwise correlations. There are $\binom{d}{2} = \frac{1}{2}d(d-1)$ pairs when order does not matter and therefore $\frac{1}{2}d(d-1)$ linear constraints from the pairwise target correlations. Altogether, the number of linear constraints grow at a rate of $\frac{1}{2}d^2 + \frac{3}{2}d$, which is a quadratic. The number of elements in the support grows exponentially as each component contributes three values so the number of different combinations of elements from each component grows at a rate of 3^d. In our current implementation, each parameter is stored so the matrices involved in the optimization problem take an extraordinary amount of computer space. This prevents our current implementation from simulating a large number of SNPs. For example, there is no way to represent

200 SNPs with the current algorithm since the matrices involved would have dimensions on the order of 3^{200} and this impossible to directly represent. If each entry in the matrix required one bit for storage (smallest unit of storage in a computer), then the matrix would need approximately $2.6 \cdot 10^{95}$ bits or approximately $3.3 \cdot 10^{82}$ terabytes for storage which is an infeasible number.

One way to improve our implementation would be to only store the values in the matrix that are non-zero. Since the simplex method gives a solution that is a vertex of a feasible space of solutions, the number of non-zero parameters will be the same as the number of linear constraints. As a result, the number of non-zero parameters will grow at a quadratic rate instead of exponential. This optimization was not done due to time constraints.

In our current implementation, the novel Bernoulli method can simulate more SNPs than our novel binomial method. This is because the current number of parameters stored by our novel Binomial method grows at a rate proportional to 3^d whereas the number of parameters stored by our novel Bernoulli grows at a rate of 2^d. If only the non-zero parameters are stored, then the number of parameters stored would also grow at a rate proportional to d^2. Thus, our novel Bernoulli method can currently simulate a larger number of SNPs than our novel binomial method but improvements to the algorithm can increase the number of SNPs that can be simulated by both methods to about the same level.

The choice of heuristic used could impact the solution to the optimization problem. For example, the heuristic used in the linear solver could be changed to a different one such as Dantzig's. Changing the heuristic used changes which parameters the linear solver adjusts when it is iterating over the value of the objective function and the

(A)

		$\boldsymbol{X}_2$		
		0	1	2
$\boldsymbol{X}_1$	0	0.5184	0.1152	0.0064
	1	0.2592	0.0576	0.0032
	2	0.0324	0.0072	0.0004

(B)

		$\boldsymbol{X}_2$		
		0	1	2
$\boldsymbol{X}_1$	0	0.5300	0.2400	0.0400
	1	0.1000	0.0800	0.0000
	2	0.0100	0.0000	0.0000

Table 3.2: The (A) independence PMF for $\boldsymbol{X}$ for $d = 2$, $p_1 = 0.1$, $p_2 = 0.2$ and $\rho_x = 0$ and the (B) joint PMF for $\boldsymbol{X}$ calculated by our novel binomial method. The element of X_1 is indicated by the rows and element of X_2 is indicated by the columns.

parameter to change. Assuming a solution exists, then there is an infinite number of solutions that satisfy the system of equations. Therefore, if the parameters are chosen in a different way, then the solution to the system of equations may be different.

It is important to note that setting all pairwise linear correlations to zero does not produce the independence PMF. The independence PMF is the joint PMF where each component of a random vector is independent of all other components. As a result, the correlation between components in the independence PMF is zero. In Table 3.2 we see two possible joint PMFs of $\boldsymbol{X}$ when there are two SNPs, the target correlation is zero and the MAFs are 0.1 and 0.2. Clearly, the independence PMF (panel A) is different than the joint PMF calculated by our novel binomial method (panel B) but both have correlation equal to zero. Additionally, in Table 3.2, we see that for panel B, $\mathrm{P}[X_1 = 0|X_2 = 2] = 1$ but $\mathrm{P}[X_1 = 0] = 0.81$, which clearly shows that X_1 and X_2 are not independent components in panel B. The joint PMF for $\boldsymbol{X}$

from our novel Bernoulli method is equal to the independence PMF for $\boldsymbol{X}$. If the independence PMF is desired, each component should be simulated separately.

Overall, our novel binomial method simulates observations from a distribution that satisfies the constraints placed on $\boldsymbol{X}$. The sample correlations and MAF are all close to the target values. Furthermore, by construction, the linear constraints from marginal distributions and correlations make the calculated joint PMF a possible joint PMF for $\boldsymbol{X}$. Simulating observations once a PMF is known can be done following the same procedure in Section 2.3.

Chapter 4

Comparisons between Simulation Methods

This chapter compares three simulation methods to each other: the Madsen and Birkes method and the two versions of our novel method (the Bernoulli version that uses Bernoulli marginal distributions and the binomial version that uses binomial marginal distributions). The Kang and Jung method will not be considered as it does not extend to random vectors with more than two components without additional arbitrary linear constraints. These methods are compared in terms of the resulting observed empirical distribution, the estimated bias of the pairwise correlations, how many confidence intervals for the Fisher transformed sample correlation contain the target value and their computational speed.

4.1 Empirical Distributions

The empirical distribution of the random vectors generated using the three aforementioned methods will be used to get an idea about how different the samples from each method are. In Table 4.1, we see the three empirical distributions from a sample of ten million generated by our three methods where the MAFs were set to 0.1 and 0.2 and target correlation was set to 0.1. Table 4.1 shows that elements (1,2), (2,1) and (2,2) did not appear in the sample from our novel binomial method whereas all elements in the support of $\boldsymbol{X}$ appeared in the samples from the other two methods. Furthermore, Table 4.2 shows that four elements did not appear in the sample generated by our novel Bernoulli method when there were three SNPs with MAFs set to 0.1, 0.2, and 0.15 with all pairwise correlations equal to 0.1. Despite the large sample size, some elements in the support of $\boldsymbol{X}$ did not appear in the sample generated by our novel Bernoulli method for Table 4.2, indicating that some elements had an extremely small or zero probability to occur. In the case of the novel binomial method, the elements that do not appear in the sample generated for Table 4.1 or Table 4.2 do indeed have zero probability to occur if we look at the calculated joint PMF from our novel binomial method.

Our novel binomial method will have elements in the support of $\boldsymbol{X}$ with zero probability to occur because the joint PMF for $\boldsymbol{X}$ it calculates is the solution to a system of linear equations. In Chapter 3, we showed that the system of linear equations can be solved as an optimization problem using the simplex method in R where the resulting solution is a possible joint PMF for $\boldsymbol{X}$ (Berkelaar et al., 2020). The simplex method calculates solutions to the system of equations that are vertices of the feasible space of solutions. Those vertices will have some parameters equal to

x	Novel Binomial	Novel Bernoulli	Madsen Birkes
$(0,0)$	0.5539	0.5359	0.5356
$(0,1)$	0.2162	0.2460	0.2462
$(0,2)$	0.0401	0.0282	0.0279
$(1,0)$	0.0759	0.0996	0.0998
$(1,1)$	0.1039	0.0696	0.0694
$(1,2)$	0.0000	0.0108	0.0111
$(2,0)$	0.0100	0.0046	0.0046
$(2,1)$	0.0000	0.0043	0.0044
$(2,2)$	0.0000	0.0010	0.0009

Table 4.1: The empirical distribution of our generated random vectors for $d = 2,\ p_1 = 0.1,\ p_2 = 0.2$, target correlation equal to 0.1 and sample size of ten million. Our novel binomial method, our novel Bernoulli method and the Madsen and Birkes method each generated a single sample. Values are rounded to four decimal places.

zero since the non-negativity constraints on the parameters are required in addition to the constraints from the marginal distributions and pairwise correlations to have a closed space of feasible solutions.

As seen in Table 4.2, four elements in the support of $\boldsymbol{X}$ do not appear in the sample of ten million generated by our novel Bernoulli method. Our novel Bernoulli method calculates a joint PMF but it is of an intermediate random vector, $\boldsymbol{Y}$, that is ultimately used to generate observations that satisfy the constraints on $\boldsymbol{X}$. Elements in the support of $\boldsymbol{Y}$ will have zero probability to occur for the same reason why elements in the support of $\boldsymbol{X}$ have zero probability to occur in our novel binomial method; both joint PMFs calculated are vertices of a feasible space of solutions. In the joint PMF for $\boldsymbol{Y}$ calculated by our novel Bernoulli method, $\mathrm{P}[\boldsymbol{Y} = (1,1,1)] = 0$. As a result, $\boldsymbol{X} \in \{(1,2,2),(2,1,2),(2,2,1),(2,2,2)\}$ all have zero probability to occur as all of them would require $\boldsymbol{Y} = (1,1,1)$. Recall that our novel Bernoulli method generates observations that satisfy the constraints on $\boldsymbol{X}$ by adding two independent and

x	Novel Binomial	Novel Bernoulli	Madsen Birkes
$(0,0,0)$	0.3301	0.4254	0.4118
$(0,0,1)$	0.1828	0.1041	0.1155
$(0,0,2)$	0.0000	0.0064	0.0081
$(0,1,0)$	0.2697	0.1612	0.1722
$(0,1,1)$	0.0000	0.0776	0.0680
$(0,1,2)$	0.0183	0.0071	0.0062
$(0,2,0)$	0.0000	0.0153	0.0178
$(0,2,1)$	0.0090	0.0110	0.0092
$(0,2,2)$	0.0000	0.0019	0.0011
$(1,0,0)$	0.1228	0.0552	0.0674
$(1,0,1)$	0.0000	0.0402	0.0294
$(1,0,2)$	0.0042	0.0041	0.0029
$(1,1,0)$	0.0000	0.0523	0.0420
$(1,1,1)$	0.0221	0.0152	0.0244
$(1,1,2)$	0.0000	0.0023	0.0030
$(1,2,0)$	0.0000	0.0079	0.0059
$(1,2,1)$	0.0310	0.0028	0.0044
$(1,2,2)$	0.0000	0.0000	0.0007
$(2,0,0)$	0.0000	0.0018	0.0028
$(2,0,1)$	0.0000	0.0022	0.0017
$(2,0,2)$	0.0000	0.0007	0.0002
$(2,1,0)$	0.0000	0.0027	0.0023
$(2,1,1)$	0.0100	0.0016	0.0018
$(2,1,2)$	0.0000	0.0000	0.0003
$(2,2,0)$	0.0000	0.0010	0.0004
$(2,2,1)$	0.0000	0.0000	0.0004
$(2,2,2)$	0.0000	0.0000	0.0001

Table 4.2: The empirical distribution of our generated random vectors for $d = 3$, $p_1 = 0.1$, $p_2 = 0.2$, $p_3 = 0.15$, all target correlations equal to 0.1 and sample size equal to ten million from our novel binomial method, our novel Bernoulli method and the Madsen and Birkes method. Values are rounded to four decimal places and no values were rounded to zero.

identically distributed observations from $\boldsymbol{Y}$. Therefore, the only way to get an element such as $\boldsymbol{X} = (2, 2, 2)$ would be from $\boldsymbol{Y} = (1, 1, 1)$ which has a probability of zero to occur according to the joint PMF calculated by our novel Bernoulli method. Table 4.2 is consistent with $\mathrm{P}[\boldsymbol{Y} = (1, 1, 1)] = 0$ as $\boldsymbol{X} \in \{(1, 2, 2), (2, 1, 2), (2, 2, 1), (2, 2, 2)\}$ do not appear in the sample generated by our novel Bernoulli method.

The possible joint PMF for $\boldsymbol{X}$ from the Madsen and Birkes method will have non-zero probabilities for every element in the support of $\boldsymbol{X}$ because the method entails mapping observations generated from a multivariate normal random variable, $\boldsymbol{Z}$, to the support of $\boldsymbol{X}$. Recall from Chapter 2 that observations from $\boldsymbol{Z}$ are mapped to the support of $\boldsymbol{X}$ based on cut-offs on the support of marginal distribution of $\boldsymbol{Z}$ determined by the target binomial marginal distribution of $\boldsymbol{X}$. The cut-offs divide the support of the marginal normal distribution into three intervals; each interval is mapped to the support of the target binomial distribution $\{0, 1, 2\}$. Each observation from $\boldsymbol{Z}$ is changed coordinate by coordinate based on the cut-offs of each marginal distribution; the end result is an observation that satisfies the constraints on $\boldsymbol{X}$. The three intervals that are used to divide the support of the marginal normal distribution all have positive length and the cut-offs of each component are independent of each other, therefore every element in the support of $\boldsymbol{X}$ has positive probability.

4.2 Estimated Bias for Correlation

The bias for sample correlation was estimated to understand if the sample correlation was under or overestimating the target correlation. The bias, based on 1000 samples of size 1000, was calculated as the difference between the average sample correlation and the target correlation. Bias estimates were calculated from a scenario with two

		Novel Binomial	Novel Bernoulli	Madsen Birkes
2 SNP	Min	-0.00521	-0.00234	-0.00324
2 SNP	Max	0.00316	0.00185	0.00169
3 SNP	Min	-0.00119	-0.00231	-0.00278
3 SNP	Max	0.00266	0.00222	0.02847

Table 4.3: The minimum and maximum estimated bias for sample correlation across all of the unique input parameters tested for 2 SNPs and 3 SNPs are listed in this table. Values have been rounded to four decimal places. The estimated bias was calculated as the difference between the average sample correlation and the target correlation; 1000 samples of size 1000 were used. The estimated bias was calculated for a scenario with 2 SNPs and a scenario with 3 SNPs; in the two SNP simulation, the MAFs and correlation used is the factorial combination of the following values: the MAF for SNP1 was 0.1, the MAF of SNP2 was one of $(0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5)$, and the target correlation was between 0 and 1 in increments of 0.1. In the three SNP simulation, the input parameters were the factorial combination of the following: the three SNPs had a MAF of $0.1, 0.1$, and 0.1 or a MAF of $0.1, 0.15$ and 0.05, target correlation between SNP1 and SNP2 was varied between 0 and 1 in increments of 0.2, target correlation between SNP1 and SNP3 was one of $(0, 0.2, 0.8)$ and target correlation between SNP2 and SNP3 was one of $(0, 0.2, 0.8)$. Any input parameters that violated the upper bound on correlation were discarded.

SNPs and scenario with three SNPs. In the two SNP simulation, the MAF for SNP1 was 0.1, the MAF of SNP2 was one of $(0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5)$, and the target correlation was between 0 and 1 in increments of 0.1. In the three SNP simulation, the three SNPs all had a MAF of 0.1 or a MAF of $0.1, 0.15$ and 0.05, target correlation between SNP1 and SNP2 was varied between 0 and 1 in increments of 0.2, target correlation between SNP1 and SNP3 was one of $(0, 0.2, 0.8)$ and target correlation between SNP2 and SNP3 was one of $(0, 0.2, 0.8)$. Combinations of input parameters that violate the upper bound on target correlation given the MAF were not considered.

The range of the estimated bias across all of the input parameters is displayed in Table 4.3. Out of the 160 different combination of input parameters, two sets of parameters resulted in an estimated bias that was an order of magnitude higher

than all the others. This indicates that there may be a certain specification of input parameters where a significant bias is expected. As a result, these two estimated biases should be further investigated to determine what led to them being larger than the other 158 estimated biases. Due to time constraints, this investigation was not carried out.

4.3 Coverage Probabilities

A confidence interval for correlation was constructed for each sample by first transforming the sample correlation (r) via $\frac{1}{2}\ln\left(\frac{1+r}{1-r}\right) = \text{arctanh}(r)$. Fisher (1915) showed that applying this transformation to the sample correlation for a bivariate normal random variable resulted in a random variable that is asymptotically distributed as a normal random variable with mean $\frac{1}{2}\ln\left(\frac{1+\rho}{1-\rho}\right)$ and variance $\frac{1}{n-3}$ where n is the number of observations in the sample and ρ is the population correlation coefficient. Confidence intervals can therefore be constructed using the asymptotic distribution. The coverage probability calculated here is not the usual one where it is the number of confidence intervals that contain the true value. Instead, coverage probability is calculated as the number of confidence intervals that contain the target value. If the simulation methods are working as intended, then by construction, the true value would be equal to the target value. Also, while our data is not from a bivariate normal distribution, Hawkins (1989) showed that the Fisher transformation can be used in non-bivariate normal settings as the mean of the asymptotic distribution is arctanh(r) and the variance depends on the joint distribution. For simplicity, we will assume the variance for the asymptotic distribution of a Fisher transformed sample correlation in our non-bivariate normal setting to be $\frac{1}{n-3}$.

Target Correlation	Novel Binomial	Novel Bernoulli	Madsen Birkes
0	86.70 (1.073)	94.70 (0.7085)	94.30 (0.7332)
0.2	82.90 (1.191)	92.20 (0.8480)	92.80 (0.8174)
0.4	83.20 (1.182)	94.10 (0.7451)	91.70 (0.8724)
0.6	91.90 (0.8627)	*	94.80 (0.7021)

Table 4.4: The number of confidence intervals constructed from the Fisher transformed sample correlation that contain the target correlation value is displayed as a percentage with standard errors in brackets. A thousand confidence intervals were calculated for samples generated by our novel binomial method, by our novel Bernoulli method and by the Madsen and Birkes method. Each sample had 1000 observations and was a simulation of three SNPs with MAFs equal to 0.1, 0.3 and 0.1, target correlation between SNP1 and SNP2 equal to 0.2, target correlation between SNP2 and SNP3 equal to 0 and target correlation between SNP1 and SNP2 indicated in the first column of the table. (*) is an empty entry as the input parameters violate the upper bound on correlation between SNP1 and SNP2 for the novel Bernoulli method.

The coverage probability for correlation was calculated in order to compare the methods to each other. While the coverage is not expected to reach the nominal value, we do compare the coverage probabilities relative to the other methods. Coverage probability was calculated for two SNPs and three SNPs; in the three SNP scenario MAFs were fixed at 0.1, 0.3 and 0.1, correlation between SNP1 and SNP3 was 0.2, correlation between SNP2 and SNP3 was 0 and the correlation between SNP1 and SNP2 ranged from 0 to 0.6 in increments of 0.2. In the two SNP scenario, MAFs were fixed at 0.1 and 0.3 and the target correlation ranged from 0 to 0.6 in increments of 0.2. Table 4.4 displays the coverage probability as a percentage for the three SNP scenario; standard errors are listed in brackets.

As seen in Table 4.4, the coverage is lower than the nominal 95% level in most cases. The coverage for our novel binomial method is lower for all correlations as shown in Table 4.4; this pattern is also present in the generated two SNP data. One potential reason why our novel binomial method has a lower coverage than the other

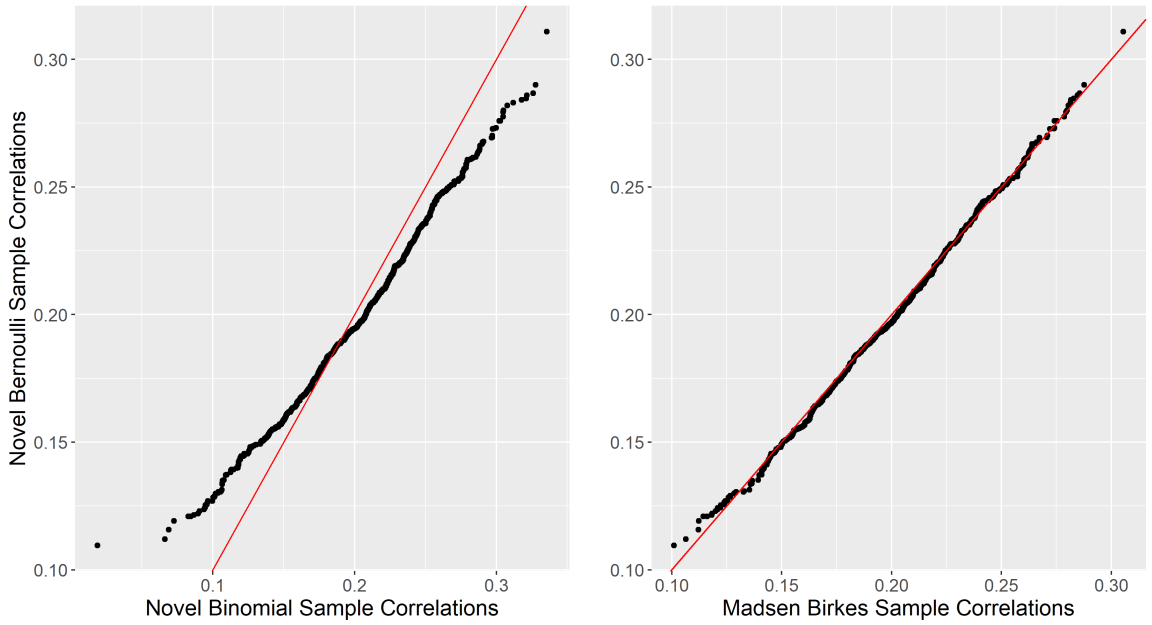

Figure 4.1: The QQ plot comparing the distributions of sample correlation. A thousand sample correlations were generated for the following input parameters: three SNPs, MAFs were 0.1, 0.3 and 0.1, target correlation between SNP1 and SNP2 and between SNP1 and SNP3 was 0.2, target correlation between SNP2 and SNP3 was 0 and sample size was 1000. The graph on the left compares the sample correlations from our novel binomial method to our novel Bernoulli method. The graph on the right compares the sample correlations from the Madsen and Birkes method to our novel Bernoulli method.

two methods may be due to the asymptotic distribution for the transformed sample correlation. The theoretical asymptotic variance was simplified to be $\frac{1}{n-3}$; if the simplification is a poor assumption, then the variance used in the confidence interval construction is inaccurate. Therefore, the constructed confidence intervals would be incorrect which impacts the number of confidence intervals that contain the target value.

Table 4.5 shows that the estimated variance for our novel binomial method is larger than the simplified theoretical asymptotic variance and the estimated variance

(a)

	Theoretical Value	Novel Binomial	Novel Bernoulli	Madsen Birkes
Mean	0.2027	0.2053	0.2015	0.2027
Variance	0.001003	0.002247	0.001217	0.001193

(b)

	Theoretical Value	Novel Binomial	Novel Bernoulli	Madsen Birkes
Mean	0.2027	0.2026	0.2026	0.2024
Variance	0.0001003	0.0001977	0.0001180	0.0001136

Table 4.5: The expected and estimated value (from each method) for the Fisher transformed sample correlation are displayed in the table. The theoretical values for mean and variance were calculated as $\frac{1}{2}\ln\left(\frac{1+\rho}{1-\rho}\right)$ and $\frac{1}{n-3}$ respectively where $\rho = 0.2$ and (a) $n = 1000$ or (b) $n = 10000$. n is the sample size and ρ is the target correlation between SNP1 and SNP2. The estimated mean and variance were calculated as the sample mean and sample variance from 1000 sample correlations. Each sample correlation came from a sample with three SNPs where the MAFs were 0.1, 0.3 and 0.1; the target correlation between SNP1 and SNP2 and between SNP 1 and SNP3 was 0.2, the target correlation between SNP2 and SNP3 was 0.

for other two methods. The other two methods are also much closer to the simplified theoretical asymptotic variance. Furthermore, the QQ plot on the left in Figure 4.1 shows evidence that the sample correlations from our novel binomial method and the sample correlations from our novel Bernoulli method are distributed differently. In contrast, the graph on the right in Figure 4.1 shows evidence that the sample correlations from our novel Bernoulli method and from Madsen and Birkes method are similarly distributed. Lastly, if we compare the boxplots of the sample correlation in Figure 4.2, we can see that the IQR from our novel binomial method is larger than the other two methods. Altogether, this is evidence that the sample correlations from our novel binomial method are distributed differently than the other two methods for our input parameters.

Additionally, Table 4.5 indicates that a small sample size was not the issue for

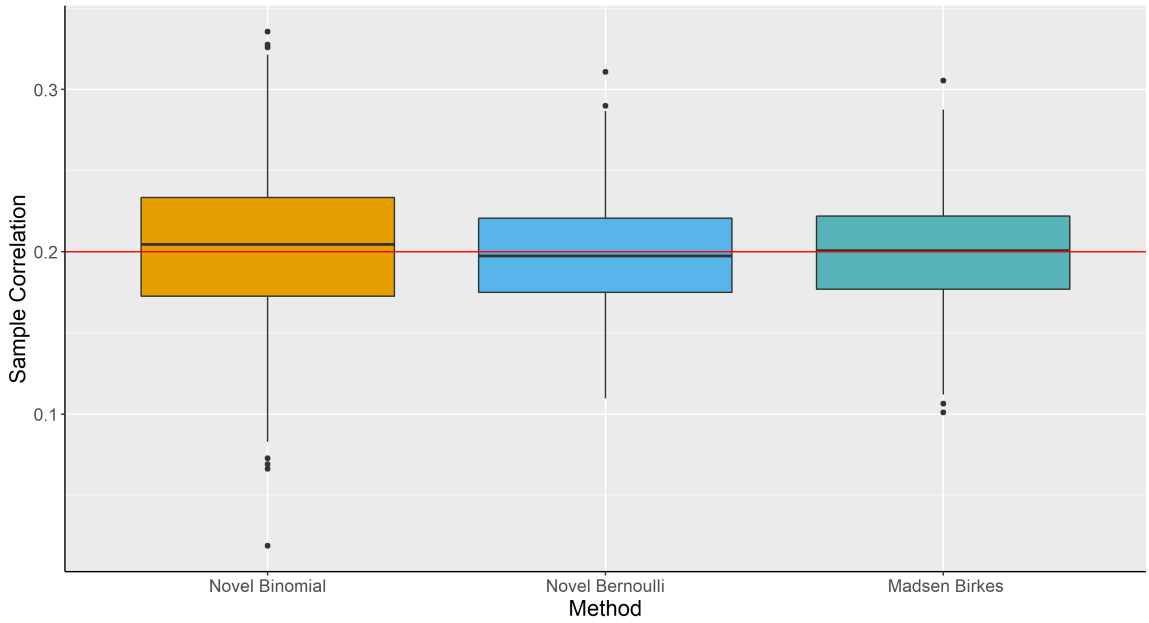

Figure 4.2: The boxplot for sample correlation for each method. A thousand sample correlations were generated for the following input parameters: three SNPs, MAFs were 0.1, 0.3 and 0.1, target correlation between SNP1 and SNP2 and between SNP1 and SNP3 was 0.2, target correlation between SNP2 and SNP3 was 0 and sample size was 1000.

the estimated variance from our novel binomial method being larger than the other two methods. Table 4.5 shows the estimated mean and variance of the transformed sample correlation for samples with (a) 1000 observations or (b) 10000 observations for three SNPs with MAFs of 0.1, 0.3 and 0.1, and the target correlations between SNP1 and SNP2 is 0.2, between SNP1 and SNP3 is 0.2 and between SNP2 and SNP3 is 0. For both sample sizes, the estimated variance from our novel binomial method with either sample size is larger than the theoretical value, indicating that a small sample size was not the issue.

Overall, it appears that the simplified theoretical asymptotic variance is a poor assumption for our novel binomial method. While the simplification works well for

our novel Bernoulli method and the Madsen and Birkes method, it is not the true variance. The true theoretical asymptotic variance is a function of the joint distribution (Hawkins, 1989). Since the joint distribution is known for our novel methods, the variance of the asymptotic distribution can be calculated; this was not done here due to time constraints.

4.4 Timing Measurements

We measured the computational time for each method using the `system.time` function in R version 4.0.2 (2020-06-22). Two simulations were carried out - the first simulation explored the effect of increasing sample size by fixing MAFs to be 0.1 and 0.3, the correlation to be 0.2 and varying the sample size between 1000, 10000 and 1000000. The second simulation explored the effect of increasing the number of SNPs by fixing MAF to be 0.1, all pairwise correlations to be 0.1, the sample size to be 1000 and varying the number of SNPs to be 2, 5 and 8. Simulations were carried out on a Windows 10 computer with 16 GBs of RAM and an i7-7700K processor with 8 CPUs at 4.20 GHz. Measurements are an average of 100 repetitions.

As the number of SNPs or the sample size increases, the processing time increases. In both cases, the Madsen and Birkes method consistently takes the longest compared to the other two methods. The other two methods are at least three times faster. The novel binomial method is faster than the novel Bernoulli method when sample size increases and slower when the number of SNPs increase.

Increasing the number of SNPs increases the number of linear equations involved in each method. The number of linear equations increase quadratically as the majority of the equations come from the pairwise target correlations which grow at a rate of

Total SNPs	Novel Binomial	Novel Bernoulli	Madsen Birkes
2	0.0023	0.0017	0.0178
5	0.0068	0.0023	0.1822
8	0.2601	0.0125	0.5019

Table 4.6: The average amount of time (seconds) needed to generate a sample of size 1000 with varying number of SNPs by our novel binomial method, our novel Bernoulli method and the Madsen and Birkes method is displayed in the table. The methods were timed in R version 4.0.2 (2020-06-22) using `system.time`. Measurements are an average of 100 simulations. The MAFs were all 0.1 and all pairwise correlations were 0.1. Simulations were carried out on a Windows 10 computer with 16 GBs of RAM and an i7-7700K processor with 8 CPUs at 4.20 GHz.

$\binom{d}{2} = \frac{1}{2}d(d-1)$ where d is the number of SNPs. The speed at which the equations can be solved is likely to be the main issue when the number of SNPs increase and sample size is held constant. The Madsen and Birkes, as implemented here, uses the `optimize` function in R to solve the linear equations for an approximate solution. Our two methods uses the lpSolve package from Berkelaar et al. (2020) to solve an optimization problem. The difference in time is likely due to how quickly each package or function is able to solve their respective set of linear equations.

As the sample size increases from 10000 to 1000000, the time taken by the novel Bernoulli method drastically increases by two orders of magnitude. In contrast, the take taken by the novel binomial method only increases by one order of magnitude. One potential cause for the difference between both methods is the novel Bernoulli method has to generate and add two observations from an intermediate random vector, $\boldsymbol{Y}$, for an observation that satisfies the constraints on $\boldsymbol{X}$. In contrast, the novel binomial method directly generates an observation that satisfies the constraints on $\boldsymbol{X}$ by using a similar method to how the novel Bernoulli method simulates an observation from $\boldsymbol{Y}$.

Sample Size	Novel Binomial	Novel Bernoulli	Madsen Birkes
1000	0.0019	0.0026	0.0198
10000	0.0032	0.0029	0.0211
1000000	0.0281	0.2517	0.4511

Table 4.7: The average amount of time (seconds) needed to generate a sample of varying size by our novel binomial method, our novel Bernoulli method and the Madsen and Birkes method is displayed in the table. The methods were timed in R version 4.0.2 (2020-06-22) using `system.time`. Measurements are an average of 100 simulations. There were two SNPs with both MAFs equal to 0.1 and all pairwise correlations were 0.1. Simulations were carried out on a Windows 10 computer with 16 GBs of RAM and an i7-7700K processor with 8 CPUs at 4.20 GHz.

The changing sample size indicates that the implementation of the Madsen and Birkes method could be further optimized in terms of transforming the values. Since only sample size is changing, the difference between the methods is how the implementation handles the transformation from the multivariate normal (Madsen and Birkes method) or the transformation from the standard uniform (our novel method) to the desired final distribution. In our implementation, the Madsen and Birkes method transforms each coordinate of an observation separately, so it is subject to a minimum of d comparisons per observation generated. If the minimum number of comparisons for each observation were to be lower than d, then our implementation cannot be used.

4.5 Discussion

Overall, the joint PMFs for $\boldsymbol{X}$ from the Madsen and Birkes method, our novel Bernoulli method and our novel binomial method are all different. The bias estimate for sample correlation indicates that there may be specifications of input parameters where it is positively biased. The coverage probabilities are all under the nominal

95% with our method with binomial marginal distributions being the lowest. The Madsen and Birkes has comparable coverage to our novel Bernoulli method but has the slowest processing time of all three methods. As a result, our novel Bernoulli method works the best.

Unfortunately, our novel Bernoulli method also has the more restrictive set of input parameters compared to the other two methods as it cannot simulate as many SNPs as the Madsen and Birkes method and it has a smaller upper bound on pairwise correlations than either of the other two methods. Additionally, our novel binomial method currently cannot simulate as many SNPs as the Madsen and Birkes method or our novel Bernoulli method. The limitation on the number of SNPs is imposed by the matrices stored in each method. In our current implementation, the matrix in the Madsen and Birkes method has dimensions that grow at a rate of d^2 whereas our novel Bernoulli method has dimensions that grow at a rate of 2^d and our novel binomial method has dimensions that grow at a rate of 3^d. The amount memory required as d increases imposes practical constraints on the number of SNPs a method can simulate. As discussed in Chapter 3, a future optimization that only stores the non-zero values in the matrix would change the rate of growth for the matrix dimension for our novel binomial and novel Bernoulli method to be quadratic instead of exponential. Lastly, the upper bound on pairwise correlation for our novel Bernoulli method is based on two Bernoulli random variables whereas the upper bound on pairwise correlation is based on binomial random variables for the other two methods. As a result, the upper bound on the pairwise correlation for our novel Bernoulli method is lower than the other two. Nonetheless, if the set of input parameters are within the restrictions, then our novel Bernoulli method works the best.

The results from the timing measurements are likely to hold for all MAF so this was not varied in the set of input parameters tested. Changing the MAF is not expected to change the results since it does not change the number of linear equations involved - it only changes the constants used in them. The step in the algorithm that takes the longest is hypothesized to be solving the linear equations.

The variance of the transformed sample correlation is not consistently approximately equal to the the simplified theoretical asymptotic variance whereas the variance of the transformed sample correlation from our novel Bernoulli method and from the Madsen and Birkes method are approximately equal. When the variance was larger than the theoretical asymptotic variance, the coverage was lower compared to the nominal value and the coverage of the other two methods. Overall, this indicates that the simplification of the theoretical asymptotic variance works poorly for our novel binomial method. As a result, the true theoretical asymptotic variance should be calculated according to Hawkins (1989).

The variance of the transformed sample correlation should be calculated for our novel Bernoulli method in the case of five SNPs since the joint PMF for the intermediate random vector may have more elements with zero probability to occur. If the joint PMF for the intermediate random vector has a similar number elements with zero probability to occur as the joint PMF for $\boldsymbol{X}$ from our novel binomial method in the three SNP case, then it may show the same issues with simplified theoretical asymptotic variance.

Chapter 5

Discussion and Future Work

In this thesis we described a novel method to simulate a collection of d correlated autosomal biallelic SNPs which can be represented as the random vector $\boldsymbol{X} = (X_1, \ldots, X_d)$ where the i^{th} component is binomially distributed with two trials and success probability equal to the MAF of the i^{th} SNP. The pairwise correlations between components of $\boldsymbol{X}$ can be specified by a $d \times d$ symmetric positive definite correlation matrix having all diagonal entries equal to one ($\boldsymbol{\rho}_X$). Simulating correlated SNPs is important to validate and develop statistical analysis techniques used in applications like GWAS.

The novel method we developed in this thesis has two versions: the novel binomial method and the novel Bernoulli method. The novel binomial method sets up a system of equations from the distribution of each component of $\boldsymbol{X}$ and from $\boldsymbol{\rho}_X$ such that the solution to the system of equations is a possible joint PMF for $\boldsymbol{X}$. Additionally, $\boldsymbol{X}$ does not have a unique joint PMF since the system of equations has more elements than the number of non-redundant linear constraints and is therefore underdetermined. The simplex method from Berkelaar et al. (2020) was used to solve the system of equations to calculate a possible joint PMF for $\boldsymbol{X}$. The joint PMF was

then used to generate observations that satisfy the constraints on $\boldsymbol{X}$.

The novel Bernoulli method is similar to the novel binomial method except the linear constraints come from an intermediate random vector, $\boldsymbol{Y}$, with correlation matrix $\boldsymbol{\rho}_X$ and where the i^{th} component is Bernoulli distributed with success probability equal to the MAF of the i^{th} SNP. Adding two independent copies of the intermediate random vector results in $\boldsymbol{X}$. As a result, observations that satisfy the constraints on $\boldsymbol{X}$ are generated by adding two independently generated observations that satisfy the constraints on $\boldsymbol{Y}$.

Two existing methods to generate from $\boldsymbol{X}$ were also implemented and discussed: the Madsen and Birkes method, Kang and Jung method. The Kang and Jung method requires additional arbitrary linear constraints when the number SNPs exceeds two. The Madsen and Birkes method improves upon the naive multivariate normal method and does not share its issue with sample and input correlations. Therefore Madsen and Birkes method is the main method used for comparisons against our novel methods.

Overall, our novel Bernoulli method works better than the Madsen and Birkes method when the number of SNPs is low and the correlation between two components is lower than the upper bound between two Bernoulli random variables with the same success probability as the components. Our novel Bernoulli method works better as it is faster than the Madsen and Birkes method when the number of SNPs is low, has comparable coverage probability and comparable bias estimates for sample correlation. When the number of SNPs grows large - about 30 SNPs on a Windows 10 computer with 16 GBs of RAM on an i7-7700k processor with 8 CPUs a 4.20 GHz - our novel Bernoulli method cannot be run as the amount of memory required grows exponentially. When the number of SNPs is large or the correlations is near

its upper bound, the Madsen and Birkes method is the best choice compared to our novel Bernoulli method. The Madsen and Birkes method is also preferred over our novel binomial method as the coverage probability for sample correlation is lower for our novel binomial method compared to the Madsen and Birkes method.

Future work could involve continuing the investigation into why the estimated variance of the transformed sample correlation for our novel binomial method is different than that of our novel Bernoulli method, the Madsen and Birkes method or the expected value. This would start by calculating the true theoretical asymptotic variance for all of the methods according to Hawkins (1989). If the joint distribution is the underlying issue, then a potential avenue would be to decrease the number of zeros in the joint PMF resulting from our methods. This would make the PMF more similar to the underlying one from the Madsen and Birkes method which has an estimated variance closer to the simplified theoretical asymptotic variance. This can be achieved by calculating a solution to the system of equations that is an interior point instead of a vertex in the solution space. This could be done by looking into other linear solvers or by averaging different solutions to the system of equations for $\boldsymbol{X}$. Different solutions to the system of equations can be obtained starting from the solution to the system of equations our novel method calculates. A new optimization problem can be set up with the same linear constraints our novel method uses and with some arbitrary objective function. The solution our novel method calculates would be set as an initial solution to the new optimization problem. When the linear solver repeatedly changes parameters in order to find an optimal solution, each specification of parameters the linear solver goes though satisfies all of the linear constraints. Therefore, different solutions to the system of equations can be obtained by

saving each specification of parameters the linear solver goes through to calculate an optimal solution.

The efficiency of the implementation of our novel method could also be improved. The matrices involved grow exponentially which prevents the method from simulating a large number of SNPs due to limited memory on a computer. Saving only the non-zero parameters would reduce the number of saved parameters to be on the scale of a quadratic instead of a exponential. Additionally, our novel method defaults to no output if the specified correlations are beyond their upper bound. Instead, our novel method could send a warning message to the user, round the specified correlation to the upper bound and then generate observations.

Our novel method could also incorporate additional linear constraints relevant to the overall problem at hand. This could be done to reflect the problem at hand and ensure that certain combination of genotypes has a positive probability to occur. For example, if a researcher knows that $\boldsymbol{X} = (2, 2, 1, 2, 1)$ has a specific observed frequency, v, from practice, then they could specify $\mathrm{P}[\boldsymbol{X} = (2, 2, 1, 2, 1)] = v$ as an additional linear constraint. However, introducing additional linear constraints could reduce the feasible space of solutions to be empty.

Lastly, our novel method could be used for different discrete distributions. Our novel method is able to simulate for a random vector whose marginal distributions can be described with linear constraints using the probability of each element in its support. Here, the marginal distributions are binomial and a correlation matrix is also specified. However, the method would work as long as a system of equations can be set up for the desired random vector.

Appendix A

Appendix

A.1 Additional Constraints

A.1.1 Three SNPs

The single additional linear constant used in the Kang and Jung method when simulating three SNPs is the following:

$$q_{11} = 0,$$

where we denote $q_{tu} = \mathrm{P}[\boldsymbol{Y} = (t, u)]$ for $t, u \in \{0, 1\}$.

A.1.2 Five SNPs

The 16 additional linear constants used in the Kang and Jung method when simulating five SNPs are the following:

$$\begin{aligned}
&q_{00001} = 0 \quad && q_{00100} = 0 \quad && q_{00101} = 0 \\
&q_{00111} = 0 \quad && q_{01101} = 0 \quad && q_{01110} = 0 \\
&q_{01111} = 0 \quad && q_{10011} = 0 \quad && q_{10101} = 0 \\
&q_{10110} = 0 \quad && q_{10111} = 0 \quad && q_{11000} = 0 \\
&q_{11001} = 0 \quad && q_{11010} = 0 \quad && q_{11100} = 0 \\
&q_{11101} = 0,
\end{aligned}$$

where we denote $q_{tuvwx} = \mathrm{P}[\boldsymbol{Y} = (t, u, v, w, x)]$ for $t, u, v, w, x \in \{0, 1\}$.

A.2 R Code

A.2.1 Calculate Upper Bound on Correlation

Main code:

```
# Calculates the Frechet-Hoeffding bounds on correlation for two binomial r.v.
# px = success rate for first binomial r.v.
# nx = number of trials for first binomial r.v.
# py = success rate for second binomial r.v.
# ny = number of trials for second binomial r.v.
# Outputs the lower bounds, upper bounds and the pmf's that lead to each bound
```

```
CalcFrecHoefBoundsBinBin <- function(px = 0.1, nx = as.integer(2),
                                     py = 0.1, ny = as.integer(2)){

  # Check basic conditions
  if (!is.double(px) | px < 0 | 1 < px) {
    stop("Error In initial values, px;
         should be double between 0 and 1 (inclusive)")
  }
  if (!is.double(py) | py < 0 | 1 < py) {
    stop("Error In initial values, py;
         should be double between 0 and 1 (inclusive)")
  }
  if (!is.integer(nx) | nx < 0 ) {
    stop("Error In initial values, nx;
         should be non-zero integer")
  }
  if (!is.integer(ny) | ny < 0 ) {
    stop("Error In initial values, ny;
         should be non-zero integer")
  }

  # Support of X and Y
  ysup <- c(0:ny)
  xsup <- c(0:nx)
```

```
# Matrices to store pmf/cdf values
lowcdfmat <- matrix(0,
                    nrow = length(ysup),
                    ncol = length(xsup))
lowpmfmat <- matrix(0,
                    nrow = length(ysup),
                    ncol = length(xsup))
highcdfmat <- matrix(0,
                       nrow = length(ysup),
                       ncol = length(xsup))
highpmfmat <- matrix(0,
                       nrow = length(ysup),
                       ncol = length(xsup))

# Calculate pmf/cdf
for (yrow in ysup) {
  for (xcol in  xsup) {
   xval <- xcol
   yval <- yrow

   xind <- xcol + 1;
   yind <- yrow + 1
```

```
# Calculate cdf/pmf for lower bound
# Definition of cdf for lower bound
 lowcdfmat[yind, xind] <-
   max(
     pbinom(xval, nx, px) +
       pbinom(yval, ny, py) -
       1,
     0)

 # f(x,y) = F(X <= x, Y <= y) -
 #   F(X <= x - 1, Y <= y) -
 #   F(X <= x , Y <= y - 1) +
 #   F(X <= x - 1, Y <= y - 1)
 lowcdftop <- ifelse(yind > 1,
                     lowcdfmat[yind - 1, xind],
                     0)
 lowcdfleft <- ifelse(xind > 1,
                      lowcdfmat[yind, xind - 1],
                      0)
 lowcdftopleft <- ifelse(yind > 1 & xind > 1,
                         lowcdfmat[yind - 1, xind - 1],
                         0)

 lowpmfmat[yind, xind] <-
```

```
  lowcdfmat[yind, xind] -
  lowcdfleft -
  lowcdftop +
  lowcdftopleft

# Calculate pmf/cdf of upper bound
# definition of cdf for upper bound
highcdfmat[yind, xind] <-
  min(
    pbinom(xval, nx, px),
    pbinom(yval, ny, py)
    )

# f(x,y) = F(X <= x, Y <= y) -
#   F(X <= x - 1, Y <= y) -
#   F(X <= x , Y <= y - 1) +
#   F(X <= x - 1, Y <= y - 1)
highcdftop <- ifelse(yind > 1,
                     highcdfmat[yind - 1, xind],
                     0)
highcdfleft <- ifelse(xind > 1,
                      highcdfmat[yind, xind - 1],
                      0)
highcdftopleft <- ifelse(yind > 1 & xind > 1,
```

```
                              highcdfmat[yind - 1, xind - 1],
                              0)

    highpmfmat[yind, xind] <-
      highcdfmat[yind, xind] -
      highcdfleft -
      highcdftop +
      highcdftopleft

  }
}

# Calculate the correlation
# E[X], Var[X]; E[Y], Var[Y]
eX <- nx*px; varX <- nx*px*(1-px)
eY <- ny*py; varY <- ny*py*(1-py)

# E[XY]
xyvalmat <- ysup %*% t(xsup)
loweXY <- sum(lowpmfmat * xyvalmat)
higheXY <- sum(highpmfmat * xyvalmat)

# Correlation
upperCor <- (higheXY - eX*eY)/sqrt(varX*varY)
```

```
  lowerCor <- (loweXY - eX*eY)/sqrt(varX*varY)

  # Return value
  ret <- list(
    LowerBound = lowerCor,
    UpperBound = upperCor,
    pmfLower = lowpmfmat,
    pmfUpper = highpmfmat
  )

  return(ret)
}
```

A.2.2 Novel Binomial Method

Main code:

```
# Generates from correlated binomials according to the input parameters
# Uses an optimization method to get an appropriate pmf to generate
# multivariate binomial values directly

# Outputs a matrix of observations
GenerateAnySnpOptParamBinomial <- function(
  pAllele = c(0.1, 0.1, 0.1),
  matTargetCor = diag(3),
  nObs = 500
```

```
){
  # pAllele = The minor allele frequency of each SNP in order
  # matTargetCor = The target correlation matrix
  # nObs = the number of observations to generate

  # Generate integer values from the pmf generated by the method
  pmf <- calcOptPmfAnySnpBinomial(pAllele, matTargetCor)

  if (isTRUE(all.equal(sum(pmf), 0))) {
    return(matrix(NA, ncol = length(pAllele), nrow = nObs))
  }


  genIntVals <- genBag(nObs, pmf)

  # Constructs the Map between the integer values and the binomial values
  # for each SNP.
  # Each column is a SNP, each row is a  value,
  # each entry is the appropriate
  # binomial value for the integer value.
  # Note that the first column of the map matrix corresponds to the last snp,
  # The second column of the map matrix is the second last snp...
  # ....
  # The last column of the map matrix is the first snp
```

```
# Snp naming convention is

# Example of naming convention for 3 SNPs together
# 111, 112, 113, 121, 122, 123, 131, 132, 133,
# 211, 212, 213, 221, 222, 223, 231, 232, 233
# 311, 312, 313, 321, 322, 323, 331, 332, 333

# Corresponding values are
# 1, 2, 3, 4, 5, 6, 7, 8, 9,
# 10, 11, 12, 13, 14, 15, 16, 17, 18
# 19, 20, 21, 22, 23, 24, 25, 26, 27
numSnp <- length(pAllele)
mapIntToBin <- sapply(c(1:numSnp), function(snpInd){
  inner <- 3 ^ (snpInd - 1)
  outer <- 3 ^ (numSnp - snpInd)
  eexp <- snpInd - 1
  toAdd <- rep(
    c(rep(0, inner),
      rep(1, inner),
      rep(2, inner)),
    outer
  )

  return(toAdd)
```

```
  })

  # Reverse Ordering so SNP1 corresponds to column 1
  mapIntToBinOrdered <- mapIntToBin[,ncol(mapIntToBin):1]

  # use a simple apply to get the binomial values using the mapping
  multiBinomialVals <- apply(mapIntToBinOrdered, 2, function(colMap){
    colMap[genIntVals]
  })

  return(multiBinomialVals)
}
```

Calculating the PMF for the target random variable:

```
# Sets up the equations associated with the marginal binomial distributions
# Outputs a matrix
calcMatMarginalBinomial <- function(numSnp = 3,
                                    pAlleles = c(0.1, 0.1, 0.1)) {
    # Calculates the constraint matrix associated to the marginal equations
    # numSnp = the number of snps
    # pAlleles = the minor allele frequency for each snp
    # Returns a typical constraint matrix where the last column
    # is the RHS of all the equations and the LHS
    # denotes the coefficient on each variable
    # The parameters are the the probability of each element in the support of the
```

```
# target random variable.

# First, construct the Left hand side (LHS) of the equations
snpMatLHS <- matrix(double(), 0, (3 ^ numSnp))
snpVecRHS <- vector("double", 0)

for (indSnp in c(1:numSnp)) {
  # For each snp,we separately work on the values they take on
  # Note that we let the SNP value take on (1,2,3) instead of (0,1,2)
  # so we don't need to deal with some index issues

  # construct the three rows associated to a particular SNP
  snpRows <- t(sapply(c(1:3), function(valSnp) {
    # Example of naming convention for 3 SNPs together
    # 111, 112, 113, 121, 122, 123, 131, 132, 133,
    # 211, 212, 213, 221, 222, 223, 231, 232, 233
    # 311, 312, 313, 321, 322, 323, 331, 332, 333

    # If we're interested in finding the SNPs that represent the marginals
    # i.e. if we want the marginal for SNP1 = 1,
    # then we want all of the above naming convention that
    # have a 1 in the first place (1??)
    # i.e. if we want the marginal for SNP2 = 3,
    # then we want all of the above naming convention that
```

```
# have a 3 in the second place (?3?)

# We can automate the the way we find the marginals.
# Note that the SNPs are ordered lowest to highest
# (i.e. 111, 112, ..., 332, 333)

# There will be a specific number of blocks
# This number is related to the index of the SNP we're looking at
numBlocks <- 3 ^ (indSnp - 1)

# we will have a slew/block of 1's
blockOneWidth <- 3 ^ (numSnp - indSnp)

# The space between the first block of 1's is
# dependent on the value/snp of interest
blockStartZeros <- (valSnp - 1) * blockOneWidth

# The block of 1's will always be followed by a block of 0's
blockEndZeros <- (3 - valSnp) * blockOneWidth

ValBlock <- c(rep(0, blockStartZeros),
              rep(1, blockOneWidth),
              rep(0, blockEndZeros))
ValRow <- rep(ValBlock, numBlocks)
```

```
        return(ValRow)
      }))

      # Append the 3 rows for a particular SNP to the matrix of constraints
      snpMatLHS <- rbind(snpMatLHS, snpRows)

      # Append the RHS of what the above 3 rows should equal
      pSnp <- pAlleles[indSnp]
      snpVecRHS <- c(snpVecRHS, #Previous values
                     (1 - pSnp) ^ 2, #1
                     2 * (1 - pSnp) * pSnp, #2
                     pSnp ^ 2) #3
    }

    # Combine into a matrix and return
    snpMatConstraints <- as.matrix(cbind(snpMatLHS, snpVecRHS))
    return(snpMatConstraints)
  }


# Sets up the equations associated to the target correlation constraints
# Outputs a matrix
calcMatTargetCorBinomial <- function(numSnp = 3,
```

```
                            matTargetCor = diag(3),
                            pAlleles = c(0.1, 0.1, 0.1)) {
  # Calculates the constraint matrix associated to the
  # Target Correlation equations
  # numSnp = the number of snps
  # matTargetCor = the target correlation matrix
  # pAlleles = the alleles associated to each snp
  # Returns a typical constraint matrix where the last
  # column is the RHS of all the equations and the LHS
  # denotes the coefficient on each variable
  # The variables are the frequencies of the multivariate SNPs.

  # The snp values that are used in target correlation calculation
  valSnpXYmat <- matrix(c(2, 2, 3, 3,
                          2, 3, 2, 3), 2, 4, T)

  # The coefficients to be placed on the variables
  valXYTargetCor <- c(1, 2, 2, 4)

  # We will be appending to construct the matrix
  snpLHSMatTarCor <- matrix(double(), 0, (3 ^ numSnp))
  snpRHSVecTarCor <- vector("double", 0)

  for (indSnpX in c(1:numSnp)) {
```

```
for (indSnpY in c((indSnpX):numSnp)) {
  if (isTRUE(all.equal(indSnpX, indSnpY)))
    next #Skip if we look a SNP paired with itself
  # Looking at all pairs of SNPs
  # SnpX is the on a lower index (more left) than SnpY

  # For each pair of SNPs
  # we will assemble the row for each value used in the target correlation
  # separately


  # Construct a matrix where each row indicates where to place
  # The appropriate value a single pair of values
  # involved in the target correlation
  # i.e. the snp values of 2-2, 2-3, 3-2 and 3-3 are
  # all used in the target correlation
  # The contribution of each pair is calculated separately and
  # then added to get a separate row
  separateConstraint <- t(sapply(c(1:4), function(indXY) {
    valSnpX <- valSnpXYmat[1, indXY]
    valSnpY <- valSnpXYmat[2, indXY]
    valPlace <- valXYTargetCor[indXY]

    bigBlockNum <- 3 ^ (indSnpX - 1)
```

```
bigBlockWidth <- 3 ^ (numSnp - indSnpX)

bigBlockStartZeros <- (valSnpX - 1) * bigBlockWidth

# Constructing smaller block inside big block
smlBlockNum <- 3 ^ (indSnpY - indSnpX - 1)
smlBlockoneWidth <- 3 ^ (numSnp - indSnpY)

smlBlockStartZeros <- (valSnpY - 1) * smlBlockoneWidth
smlBlockOnes <- smlBlockoneWidth
smlBlockEndZeros <- (3 - valSnpY) * smlBlockoneWidth

smlblock <- c(
  rep(0, smlBlockStartZeros),
  rep(valPlace, smlBlockOnes),
  rep(0, smlBlockEndZeros)
)
smlBlockRow <- rep(smlblock, smlBlockNum)

bigBlockOnes <- smlBlockRow

bigBlockEndZeros <- (3 - valSnpX) * bigBlockWidth

# Constructing final big block
```

```
  bigBlock <- c(rep(0, bigBlockStartZeros),
                rep(smlBlockRow, 1),
                rep(0, bigBlockEndZeros))

  bigBlockRow <- rep(bigBlock, bigBlockNum)

  return(bigBlockRow)
}))

# collapse the above matrix into a row vector to be
# used in linear optimization
singleRowConstraint <- apply(separateConstraint, 2, sum)
snpLHSMatTarCor <- rbind(snpLHSMatTarCor, singleRowConstraint)


# RHS of target correlation is targetCor * sqrt(VarX*VarY) + eX*EY
pX <- pAlleles[indSnpX]
pY <- pAlleles[indSnpY]
nXY <- 2
rhseXYTarCor <-
  matTargetCor[indSnpX, indSnpY] *
  sqrt(nXY * pX * (1 - pX) * nXY * pY * (1 - pY)) +
  (nXY * pX * nXY * pY)
```

```
      snpRHSVecTarCor <-
        c(snpRHSVecTarCor, rhseXYTarCor)
    }
  }

  snpMatTarCor <- as.matrix(cbind(snpLHSMatTarCor, snpRHSVecTarCor))
  # snpMatTarCor
  return(snpMatTarCor)
}


# Combines the marginal and target correlation constraints into a single matrix
# Outputs a matrix
calcMatPmfConstaintBinomial <-function(numSnp = 3,
                               matTargetCor = diag(3),
                               pAlleles = c(0.1, 0.1, 0.1)) {
  # Calculates the constraint matrix associated to
  # the Target Correlation equations
  # numSnp = the number of snps
  # matTargetCor = the target correlation matrix
  # pAlleles = the alleles associated to each snp
  # Returns A typical constraint matrix where the
  # last column is the RHS of all the equations and the LHS
  # denotes the coefficient on each variable
```

```
  # The variables are the frequencies of the multivariate SNPs.

  matMarg <- calcMatMarginalBinomial(numSnp, pAlleles)
  matTarC <- calcMatTargetCorBinomial(numSnp, matTargetCor, pAlleles)

  snpMat <- as.matrix(rbind(matMarg, matTarC))
  return(snpMat)
}

# Calculates the pmf based on optimizing the set of constraints
# as given by the marginals and target correlation
# Outputs a vector that contains the probability of each element
calcOptPmfAnySnpBinomial <- function(pAllele, matTarCor){
  require(lpSolve)
  # pAllele = vector in minor allele frequencies
  # matTarCor = correlation matrix

  numSnp <- length(pAllele)
  oriMatSnp <- calcMatPmfConstaintBinomial(length(pAllele), matTarCor, pAllele)

  # Need to separate matrix to LHS and RHS and add the unity constraint
  LHSOriMatSnp <- oriMatSnp[,-ncol(oriMatSnp)]
  RHSOriMatSnp <- as.vector(oriMatSnp[,ncol(oriMatSnp)])
```

```
# Add unity constraint to the above.
# RHS only needs a 1 added
uniRHSOriMatSnp <- c(RHSOriMatSnp, 1)

# LHS needs an extra row and column, last diagonal entry needs to be 1
uniLHSOriMatSnp1 <- rbind(LHSOriMatSnp, 0)
uniLHSOriMatSnp2 <- cbind(uniLHSOriMatSnp1, 0)
uniLHSOriMatSnp2[nrow(uniLHSOriMatSnp2), ncol(uniLHSOriMatSnp2)] <- 1
uniLHSOriMatSnp <- uniLHSOriMatSnp2

# Objective function
# All of the variables added should equal the unity (last) variable
obj <- c(
  rep(1,ncol(LHSOriMatSnp))
  ,-1)

constranints_direction  <- rep("=", length(uniRHSOriMatSnp))

optimum <-  lp(direction = "max",
               objective.in = obj,
               const.mat = uniLHSOriMatSnp,
               const.dir = constranints_direction,
               const.rhs = uniRHSOriMatSnp,
               all.int = F)
```

```
  sol <- optimum$solution

  # Remove last entry since that's the total sum
  # Remaining vector contains the entries in the PMF of the target
  # random variable
  pmfSolve <- sol[-length(sol)]

  # Name the vector according to the naming convention
  names(pmfSolve) <- namingForBinomialPmf(numSnp)
  return(pmfSolve)
}
```

Naming convention used:

```
# Gets the name for each entry in the PMF of correlated binomials
# Outputs a vector of names (string)
namingForBinomialPmf <- function(numSnp){
  # numSNP = number of SNPs
  # Naming convention follows a specific pattern
  # they are the ordered (lowest to highest) of what
  # value (1,2,3) each snp (indicated by index) takes on
  # For example, if snp1 is 1, snp2 is 3 and snp3 is 2,
  # then the corresponding name is 132.

  # Construct the names from right to left.
  # The pattern of 1,2,3 is a repetitive one.
```

```
  nmedoub <- rep(0, 3^numSnp)
  for (snpInd in c(1:numSnp)) {
    # snpInd <- 1
    inner <- 3 ^ (snpInd - 1)
    outer <- 3 ^ (numSnp - snpInd)
    eexp <- snpInd - 1

    toAdd <- (10^eexp)*rep(
      c(rep(1, inner),
        rep(2, inner),
        rep(3, inner)),
      outer
    )
    toAdd
    nmedoub <- nmedoub + toAdd
  }

  nmestri <- paste("x", nmedoub, sep = "")
  nmestri
}
```

Generating integers:

```
# Generates integer given the probability for each integer
# nObs = number of observations to generate
# pGeno = vector of probability weights
```

```
# Outputs a vector of integers
genBag <- function(nObs, pGeno){
  if (!all.equal(sum(pGeno), 1)) print("Given probabilities don't add to 1 ")

  # Generate and return values
  genBagVals <- sample.int(length(pGeno), nObs , replace=TRUE, prob=pGeno)
  return(genBagVals)
}
```

A.2.3 Novel Bernoulli Method

Main code:

```
# Generates correlated Bernoulli's according to the input parameters
# Uses the optimization method to get an appropriate
# Independently generates twice the number of multivariate bernoulli values
# such that they can be added component wise to get
# binomial marginal distributions

# Outputs a matrix of observations
GenerateAnySnpOptParamBernoulli <- function(
  pAllele = c(0.1, 0.1, 0.1),
  matTargetCor = diag(3),
  nObs = 500
){
  # pAllele = The minor allele frequency of each SNP in order
```

```
# matTargetCor = The target correlation matrix
# nObs = the number of observations to generate

# Generate integer values from the pmf generated by the method
# We'll generate all the needed bernoulli values at once,
# separate them at the end and add them to get binomial values
pmf <- calcOptPmfAnySnpBernoulli(pAllele, matTargetCor)

if (isTRUE(all.equal(sum(pmf), 0))) {
  return(matrix(NA, ncol = length(pAllele), nrow = nObs))
}

genIntVals <- genBag(2*nObs, pmf)

# Constructs the Map between the integer values and the binomial values
# for each SNP.
# Each column is a SNP, each row is an
# integer value, each entry is the appropriate binomial value for
# the integer value.
# Note that the first column of the map matrix corresponds to the last snp,
# The second column of the map matrix is the second last snp...
# ....
# The last column of the map matrix is the first snp
# Snp naming convention is
```

```
# Example of naming convention for 3 SNPs together
# 111, 112, 121, 122, 211, 212, 221, 222

# Corresponding integer values are
# 1, 2, 3, 4, 5, 6, 7, 8
numSnp <- length(pAllele)
mapIntToBern <- sapply(c(1:numSnp), function(snpInd){
  inner <- 2 ^ (snpInd - 1)
  outer <- 2 ^ (numSnp - snpInd)
  eexp <- snpInd - 1
  toAdd <- rep(
    c(rep(0, inner),
      rep(1, inner)),
    outer
  )

  return(toAdd)
})

# Reverse Ordering so SNP1 corresponds to column 1
mapIntToBernOrdered <- mapIntToBern[,ncol(mapIntToBern):1]

# use a simple apply to get the bernoulli values using the mapping
```

```
  multiBernVals <- apply(mapIntToBernOrdered, 2, function(colMultiMap){
    colMultiMap[genIntVals]
  })

  # Add the halves of the bernoulli values together to get binomial values
  multiBinVals <- apply(multiBernVals, 2, function(colBernVal){
    (colBernVal[1:nObs] + colBernVal[(nObs + 1): length(colBernVal)])
  })

  return(multiBinVals)
}
```

Calculating the PMF involved:

```
# Based on Adding two bernoulli's
# Sets up the equations associated with the marginal constraints
# Outputs a matrix
calcMatMarginalBernoulli <- function(numSnp = 3,
                                     pAlleles = c(0.1, 0.1, 0.1)) {
  # Calculates the constraint matrix associated to the marginal equations
  # numSnp = the number of snps
  # pAlleles = the minor allele frequency for each snp
  # Returns a typical constraint matrix where the
  # last column is the RHS of all the equations and the LHS
  # denotes the coefficient on each variable
  # The variables are the frequencies of the multivariate SNPs.
```

```
# See below for the naming convention.

# First, construct the Left hand side (LHS) of the equations
snpMatLHS <- matrix(double(), 0, (2 ^ numSnp))
snpVecRHS <- vector("double", 0)

for (indSnp in c(1:numSnp)) {
  # For each snp,we separately work on the values they take on
  # Note that we let the SNP value take on (1,2,3) instead of (0,1,2)
  # so we don't need to deal with some index issues

  # construct the three rows associated to a particular SNP
  snpRows <- t(sapply(c(1:2), function(valSnp) {
    # Example of naming convention for 3 SNPs together
    # 111, 112, 121, 122, 211, 212, 221, 222

    # If we're interested in finding the SNPs that represent the marginals
    # i.e. if we want the marginal for SNP1 = 0,
    # then we want all of the above naming convention that have a 1
    # in the first place (1)
    # i.e. if we want the marginal for SNP2 = 2,
    # then we want all of the above naming convention that
    # have a 3 in the second place
```

```
# We can automate the the way we find the marginals.
# Note that the SNPs are ordered lowest to highest
# (i.e. 111, 112, ..., 221, 222)

# There will be a specific number of blocks
# This number is related to the index of the SNP we're looking at
numBlocks <- 2 ^ (indSnp - 1)

# we will have a slew/block of 1's
blockOneWidth <- 2 ^ (numSnp - indSnp)

# The space between the first block of 1's is dependent
# on the value/snp of interest
blockStartZeros <- (valSnp - 1) * blockOneWidth

# The block of 1's will always be followed by a block of 0's
blockEndZeros <- (2 - valSnp) * blockOneWidth

ValBlock <- c(rep(0, blockStartZeros),
              rep(1, blockOneWidth),
              rep(0, blockEndZeros))
ValRow <- rep(ValBlock, numBlocks)

return(ValRow)
```

```
    }))

    # Append the 2 rows for a particular SNP to the matrix of constraints
    snpMatLHS <- rbind(snpMatLHS, snpRows)

    # Append the RHS of what the above 2 rows should equal
    pSnp <- pAlleles[indSnp]
    snpVecRHS <- c(snpVecRHS, #Previous values
                   (1 - pSnp), #1
                   pSnp) #2
  }

  # Combine into a matrix and return
  snpMatConstraints <- as.matrix(cbind(snpMatLHS, snpVecRHS))
  return(snpMatConstraints)
}


# Sets up the equations associated to the target correlation constraints
# Outputs a matrix
calcMatTargetCorBernoulli <- function(numSnp = 3,
                                      matTargetCor = diag(3),
                                      pAlleles = c(0.1, 0.1, 0.1)) {
  # Calculates the constraint matrix associated to the
```

```
# Target Correlation equations
# numSnp = the number of snps
# matTargetCor = the target correlation matrix
# pAlleles = the alleles associated to each snp
# Returns a typical constraint matrix where the last
# column is the RHS of all the equations and the LHS
# denotes the coefficient on each variable
# The variables are the frequencies of the multivariate
# SNPs. See below for the naming convention.

# The snp values that are used in target correlation calculation
valSnpXYmat <- matrix(c(2,
                        2), 2, 1, T)

# The coefficients to be placed on the variables
valXYTargetCor <- c(1)

# We will be appending to construct the matrix
snpLHSMatTarCor <- matrix(double(), 0, (2 ^ numSnp))
snpRHSVecTarCor <- vector("double", 0)

for (indSnpX in c(1:numSnp)) {
  for (indSnpY in c((indSnpX):numSnp)) {
    if (isTRUE(all.equal(indSnpX, indSnpY)))
```

```
  next #Skip if we look a SNP paired with itself
# Looking at all pairs of SNPs
# SnpX is the on a lower index (more left) than SnpY


# Construct a matrix where each row indicates where to place
# The appropriate value a single pair of values involved
# in the target correlation
# i.e. the snp values of 2-2, 2-3, 3-2 and 3-3 are
# all used in the target correlation
# The contribution of each pair is calculated separately
# and then added to get a separate row
separateConstraint <- t(sapply(c(1), function(indXY) {
  valSnpX <- valSnpXYmat[1, indXY]
  valSnpY <- valSnpXYmat[2, indXY]
  valPlace <- valXYTargetCor[indXY]

  bigBlockNum <- 2 ^ (indSnpX - 1)
  bigBlockWidth <- 2 ^ (numSnp - indSnpX)

  bigBlockStartZeros <- (valSnpX - 1) * bigBlockWidth

  # Constructing smaller block inside big block
  smlBlockNum <- 2 ^ (indSnpY - indSnpX - 1)
```

```
smlBlockoneWidth <- 2 ^ (numSnp - indSnpY)

smlBlockStartZeros <- (valSnpY - 1) * smlBlockoneWidth
smlBlockOnes <- smlBlockoneWidth
smlBlockEndZeros <- (2 - valSnpY) * smlBlockoneWidth

smlblock <- c(
  rep(0, smlBlockStartZeros),
  rep(valPlace, smlBlockOnes),
  rep(0, smlBlockEndZeros)
)
smlBlockRow <- rep(smlblock, smlBlockNum)

bigBlockOnes <- smlBlockRow

bigBlockEndZeros <- (2 - valSnpX) * bigBlockWidth

# Constructing final big block
bigBlock <- c(rep(0, bigBlockStartZeros),
              rep(smlBlockRow, 1),
              rep(0, bigBlockEndZeros))

bigBlockRow <- rep(bigBlock, bigBlockNum)
```

```
      return(bigBlockRow)
    }))

    # collapse the above matrix into a row vector to be
    # used in linear optimization
    singleRowConstraint <- apply(separateConstraint, 2, sum)
    snpLHSMatTarCor <- rbind(snpLHSMatTarCor, singleRowConstraint)


    # RHS of target correlation is targetCor * sqrt(VarX*VarY) + eX*EY
    pX <- pAlleles[indSnpX]
    pY <- pAlleles[indSnpY]
    nXY <- 1
    rhseXYTarCor <-
      matTargetCor[indSnpX, indSnpY] *
      sqrt(nXY * pX * (1 - pX) * nXY * pY * (1 - pY)) +
      (nXY * pX * nXY * pY)

    snpRHSVecTarCor <-
      c(snpRHSVecTarCor, rhseXYTarCor)
  }
}

snpMatTarCor <- as.matrix(cbind(snpLHSMatTarCor, snpRHSVecTarCor))
```

```
  return(snpMatTarCor)
}


# Combines the marginal and target correlation constraints into a single matrix
# Outputs a matrix
calcMatPmfConstaintBernoulli <- function(numSnp = 3,
                                         matTargetCor = diag(3),
                                         pAlleles = c(0.1, 0.1, 0.1)) {
  # Calculates the constraint matrix associated to the
  # Target Correlation equations
  # numSnp = the number of snps
  # matTargetCor = the target correlation matrix
  # pAlleles = the alleles associated to each snp
  # Returns A typical constraint matrix where the last column
  # is the RHS of all the equations and the LHS
  # denotes the coefficient on each variable
  # The variables are the frequencies of the multivariate SNPs.


  matMarg <- calcMatMarginalBernoulli(numSnp, pAlleles)
  matTarC <- calcMatTargetCorBernoulli(numSnp, matTargetCor, pAlleles)

  snpMat <- as.matrix(rbind(matMarg, matTarC))
```

```
  return(snpMat)
}



# Calculates the pmf based on optimizing the set of constraints as
# given by the marginals and target correlation
# Outputs a vector containing the probability of each element
calcOptPmfAnySnpBernoulli <- function(pAllele, matTarCor){
  require(lpSolve)
  # pAllele = vector of minor allele frequencies
  # matTarCor = correlation matrix

  numSnp <- length(pAllele)
  oriMatSnp <- calcMatPmfConstaintBernoulli(length(pAllele), matTarCor, pAllele)

  # Need to separate matrix to LHS and RHS and add the unity constraint

  LHSOriMatSnp <- oriMatSnp[,-ncol(oriMatSnp)]
  RHSOriMatSnp <- as.vector(oriMatSnp[,ncol(oriMatSnp)])

  # Add unity constraint to the above.
  # RHS only needs a 1 added
  uniRHSOriMatSnp <- c(RHSOriMatSnp, 1)
```

```
# LHS needs an extra row and column, last diagonal entry needs to be 1
uniLHSOriMatSnp1 <- rbind(LHSOriMatSnp, 0)
uniLHSOriMatSnp2 <- cbind(uniLHSOriMatSnp1, 0)
uniLHSOriMatSnp2[nrow(uniLHSOriMatSnp2), ncol(uniLHSOriMatSnp2)] <- 1
uniLHSOriMatSnp <- uniLHSOriMatSnp2

# Objective function
# All of the variables added should equal the unity (last) variable
obj <- c(
  rep(1,ncol(LHSOriMatSnp))
  ,-1)

constranints_direction  <- rep("=", length(uniRHSOriMatSnp))

optimum <-  lp(direction = "max",
               objective.in = obj,
               const.mat = uniLHSOriMatSnp,
               const.dir = constranints_direction,
               const.rhs = uniRHSOriMatSnp,
               all.int = F)
sol <- optimum$solution

# Remove last entry since that's the total sum
pmfSolve <- sol[-length(sol)]
```

```
  # Name the vector according to naming convention
  names(pmfSolve) <- namingForBernoullipmf(numSnp)
  return(pmfSolve)
}
```

Naming convention used:

```
# Gets the name for each entry in the PMF of correlated Bernoullis
namingForBernoullipmf <- function(numSnp){
  # Naming convention follows a specific pattern
  # they are the ordered (lowest to highest) of what value (1,2)
  # each snp (indicated by index) takes on
  # For example, if snp1 is 1, snp2 is 1 and snp3 is 2,
  # then the corresponding name is 112.

  # Construct the names from right to left.
  # The pattern of 1,2 is a repetitive one.
  nmedoub <- rep(0, 2^numSnp)
  for (snpInd in c(1:numSnp)) {
    # snpInd <- 1
    inner <- 2 ^ (snpInd - 1)
    outer <- 2 ^ (numSnp - snpInd)
    eexp <- snpInd - 1

    toAdd <- (10^eexp)*rep(
```

```
      c(rep(1, inner),
        rep(2, inner)),
      outer
    )
    toAdd
    nmedoub <- nmedoub + toAdd
  }

  nmestri <- paste("x", nmedoub, sep = "")
  nmestri
}
```

Generating integers:

```
# Generates integer given the probability for each integer
# nObs = number of observations to generate
# pGeno = vector of probability weights
# Outputs a vector of integers
genBag <- function(nObs, pGeno){
  if (!all.equal(sum(pGeno), 1)) print("Given probabilities don't add to 1 ")

  # Generate and return values
  genBagVals <- sample.int(length(pGeno), nObs , replace=TRUE, prob=pGeno)
  return(genBagVals)
}
```

A.2.4 Kang and Jung Method

Main code:

```
# Simulates a number of observations from
# two correlated binomials


# Outputs a matrix of observations
Generate2SnpBernoulli <- function(p1, p2, rho, nObs){
  # p1 minor allele frequency of snp 1
  # p2 minor allele frequency of snp 2
  # rho correlation between SNP1 and SNP2
  # nObs number of observations to generate



  # Compute the four probabilities for the joint distribution.
  a0 <-  rho * sqrt(p1*p2*(1-p1)*(1-p2)) + (1-p1)*(1-p2)
  prob <-
    c(
      `(0,0)` = a0,
      `(1,0)` = 1 - p2 - a0,
      `(0,1)` = 1 - p1 - a0,
      `(1,1)` = a0 + p1 + p2 - 1
    )


  # Checks for valid probabilities, skips over if invalid
```

```
  # Negative probabilities due to choice of p1, p2 and
  # correlation coefficient can cause this
  if (min(prob) < 0) {
    return(as.matrix(cbind(NA, NA)))
  }

  nbin <- 2

  # Sample integers with specific weights
  u <- sample.int(4, nBinObs * nbin, replace=TRUE, prob=prob)

  # Split into bernoulli representation
  y <- floor((u-1)/2)
  x <- 1 - u %% 2

  # Sum up "columns" to get binomial
  xx <- colSums(matrix(x, nrow=nbin)) # Sum in groups of `n`
  yy <- colSums(matrix(y, nrow=nbin)) # Sum in groups of `n`

  return(as.matrix(cbind(xx,yy)))
}
```

A.2.5 Madsen and Birkes Method

Main Code:

```
# Simulate observations from correlated binomials using
# the Madsen Birkes method

# Returns the generated observations in a matrix as well as the
# correlations used in the multivariate matrix as additional columns
# in the matrix
GenerateSnpMadsenBirkes <- function(pAlleleMinor, targetBinCorMat,
                                    nBinObs, useEigAdjust = F){
  # pAlleleMinor vector of minor allele frequencies
  # targetBinCorMat Matrix of desired correlations
  # nBinObs Number of observations to generate

  require(gdata)
  require(matrixcalc)
  require(MASS)
  nBin <- rep(2, length(pAlleleMinor))

  # Calculate correlation matrix for multivariate normal
  sigmaZ <- CalculateSigmaZ(nBin, pAlleleMinor, targetBinCorMat)

  # Ensures the correlation matrix is positive definite if desired
  if (useEigAdjust) {
    sigmaZ <- CalcAdjustMadsenInverseEigen(sigmaZ)
  }
```

```
if (is.symmetric.matrix(sigmaZ)){
  if (!is.positive.definite(sigmaZ)){
    llen <- length(pAlleleMinor) + ncol(sigmaZ)
    retVal <- matrix(NA, ncol = llen)
    return(retVal)
  }
}

normMeans <- nBin * 0 # mean is just 0

# Simulate from multivariate normal
generatedNormalObs <- mvrnorm(nBinObs, normMeans, sigmaZ)

# Convert values from multivariate normal to binomial
transformedNormToBinMat <- as.matrix(
  TransformNormToBin(
    pAlleleMinor = pAlleleMinor,
    normalGeneratedMat = generatedNormalObs)
)

# Return value
retVal <- cbind(transformedNormToBinMat,
                matrix(upperTriangle(sigmaZ), byrow = T,
```

```
                          ncol = length(upperTriangle(sigmaZ)),
                          nrow = nBinObs))
  return(as.matrix(retVal))
}
```

Sets up the calculation for each entry in the correlation matrix for the multivariate normal:

```
# Calculates the correlation matrix used for the multivariate normal
# given the target correlation matrix for the correlated binomial random vector


# Outputs a matrix
CalculateSigmaZ <- function(nBin, pAlleleMinor, corDesire) {
  # nBin = Should be a vector of just 2's. The number of "trials"
  # pAlleleMinor = vector of minor allele frequencies
  # corDesire = correlation matrix for the target r.v.


  # Calculate correlation matrix for normals
  corLen <- length(corDesire[,1])
  sigmaZ <- diag(length(corDesire[, 1]))


  # Note: Can speed up by running loop in parallel
  # Calculates the correlation matrix
  for (i in c(1:corLen)) {
    for (j in c(1:corLen)) {
```

```
# Initialize values needed for calculating delta
corD <- corDesire[i, j]
pI <- pAlleleMinor[i]
pJ <- pAlleleMinor[j]
nI <- nBin[i]
nJ <- nBin[j]
sigmaI2 <- pI * nI * (1 - pI)
sigmaJ2 <- pJ * nJ * (1 - pJ)
uI <- nI * pI
uJ <- nJ * pJ

# Calculate the matrix entry by entry
# The matrix is symmetric and main diagonal is all 1's
if (i == j) {
  # diagonal elements of correlation matrix
  sigmaZ[i,j] <- 1
} else if (i < j) {
  # Compute the entry in the upper triangle part
  sigmaZ[i,j] <- solveSigmaEqnForNormToBin(corD, sigmaI2, sigmaJ2,
                                           uI, uJ, nI, nJ, pI, pJ)
} else if (i > j) {
  # Copy the upper triangle part for the lower triangle
  sigmaZ[i,j] <- sigmaZ[j,i]
}
```

```
    }
  }

  return(sigmaZ)
}
```

Calculates the entry in the correlation matrix of the multivariate normal:

```
# Numerically solves the what the entry in the correlation matrix
# of the multivariate normal should be to get the desired correlation
# after transformation to correlated binomial.

# Uses optimize to solve the equation.
# Equation is from Madsen and Birkes (2013)
# Outputs a number value
solveSigmaEqnForNormToBin <- function(corDesire, sigmaI2, sigmaJ2,
                                      uI, uJ, nI, nJ, pI, pJ){
  # corDesire is the correlation desired between two binomial r.v.
  # sigmaI2, uI, nI, pI are specified parameters for the first binomial r.v.
  # sigmaJ2, uJ, nJ, pJ are specified parameters for the second binomial r.v.
  require(mvtnorm)

  # Initialize the values we will be using
  sigmaI <- sqrt(sigmaI2)
  sigmaJ <- sqrt(sigmaJ2)
```

```
# Support of distributions. 2 Binomials
rVals <- c(0,1,2,0,1,2,0,1,2)
sVals <- c(0,0,0,1,1,1,2,2,2)

# Cdf of R and S values.
FiRVec <- pbinom(rVals, nI, pI)
FjSVec <- pbinom(sVals, nJ, pJ)

# Inverse of Standard Normal of FiR and FiS
NInvFiRVec <- qnorm(FiRVec)
NInvFjSVec <- qnorm(FjSVec)

# Function we will be optimizing to find delta
# delta is what the value should be in the correlation matrix for
# the multivariate normal
fx.opt <- function(deltaInit){
  delta <- deltaInit[1]

  # Function to create bivariate normal functions
  # with the delta for optimization
  fx.BivarNorm <- function(nFiR, nFiS) {
    sigDelta <- matrix(c(1,delta,delta,1), nrow = 2, ncol = 2)
    pmvnorm(upper = c(nFiR, nFiS), mean = c(0,0), sigma = sigDelta)
```

```
    }

    # Constructs vector so we can sum the bivariate normal values up
    BiNormVec <- vector(mode = "double", length = length(rVals))
    for (i in c(1:9)) {
      BiNormVec[i] <- fx.BivarNorm(NInvFiRVec[i], NInvFjSVec[i])
    }

    # Assembling the equation
    eqSumTermVec <- 1 - FiRVec - FjSVec + BiNormVec

    rightEqTerm <- 1/(sigmaI*sigmaJ) * (sum(eqSumTermVec) - uI*uJ)
    leftTerm <- corDesire

    # What is being optimizd
    (rightEqTerm - leftTerm)^2
  }

  # Delta is between -1 to 1
  opt <- optimize(fx.opt, interval = c(-1,1))
  estimatedDelta <- opt$minimum
  return(estimatedDelta)
}
```

Maps values from normal to binomial:

```
# Maps values from the normal distribution to the binomial distribution
# based on cut-offs of the multivariate normal
# Outputs a matrix of values
TransformNormToBin <- function(pAlleleMinor, normalGeneratedMat) {
  # pAlleleMinor = vector of minor allele frequencies
  # normalGeneratedMat = matrix of values from multivariate
  # normal distribution

  transBinMat <- sapply(c(1:length(pAlleleMinor)),
                        function(indexBin) {
                          x <- normalGeneratedMat[, indexBin]
                          minAllele <-
                            pAlleleMinor[indexBin]

                          # Transform normal marginal to standard uniform
                          unifVal <-
                            pnorm(x, mean = 0, sd = 1)

                          # Map uniform to binomial using the cutoffs
                          binValue <-
                            TransformUnifToBin(minAllele, unifVal)

                          return(binValue)
                        })
```

```
  return(data.frame(transBinMat))
}

# Transforms standard uniform values to binomial(2,p)
# given that p is the minor allele frequency
TransformUnifToBin <- function(minAllele, unifVal) {
  # pAlleleMinor = minor Allele Frequency
  # unifMat = Vector of uniform values

  unifVal <- as.double(unifVal)
  minAllele <- as.double(minAllele)

  # Transform to specified marginal binomial
  # Cutoffs for mapping uniform to binomial
  binCutHomoMinor <- minAllele ^ 2
  binCutHomoMajor <- (1 - minAllele) ^ 2
  binCutHetero <- 2 * minAllele * (1 - minAllele)

  # Map uniform to binomial using the cutoffs
  binValue <-
    ifelse(unifVal < binCutHomoMinor,
           2,
           ifelse(unifVal < binCutHetero + binCutHomoMinor,
```

```
                  1,
                  0))

  return(binValue)
}
```

Adjusting for non-positive definite correlation matrix:

```
# Takes a matrix and adjusts it to be positive definite
# by setting any non-positive eigen values to a small positive number
# Outputs a matrix
CalcAdjustMadsenInverseEigen <- function(A, tol = 1e-5){
  # A is the matrix to adjust
  # tol is the small positive to set negative eigen values to be

  eigens <- eigen(A)
  eigVal <- eigens$values
  eigVec <- eigens$vectors

  # A = Q %*% Lambda %*% Q^-1
  Q <- eigVec
  Qinv <- solve(Q)

  # Adjust the lamda by setting any negatives to some small
  # positive number
```

```
  eigValAdj <- ifelse(eigVal <= 0, tol, eigVal)
  lamadj <- diag(eigValAdj,
              ncol = length(eigVal), nrow = length(eigVal))

  # Construct adjusted matrix
  Aadj <- Q %*% lamadj %*% Qinv

  # Return matrix as correlation matrix
  return(cov2cor(Aadj))
}
```

Bibliography

Berkelaar, M. et al. (2020). *lpSolve: Interface to 'Lp solve' v. 5.5 to Solve Linear/Integer Programs*. R package version 5.6.15.

Cario, M. C. and Nelson, B. L. (1997). Modeling and generating random vectors with arbitrary marginal distributions and correlation matrix. Technical report, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston.

Carvajal-Rodriguez, A. (2010). Simulation of Genes and Genomes Forward in Time. *Current Genomics*, **11**(1), 58–61.

Demirtas, H. and Hedeker, D. (2011). A practical way for computing approximate lower and upper correlation bounds. *American Statistician*, **65**(2), 104–109.

Ferrari, P. A. and Barbiero, A. (2012). Simulating Ordinal Data. *Multivariate Behavioral Research*, **47**(4), 566–589.

Fisher, R. A. (1915). Frequency Distribution of the Values of the Correlation Coefficient in Samples from an Indefinitely Large Population. *Biometrika*, **10**(4), 507.

Franks, P. W., Pearson, E., and Florez, J. C. (2013). Gene-environment and gene-treatment interactions in type 2 diabetes: progress, pitfalls, and prospects. *Diabetes Care*, **36**(5), 1413–1421.

Frechet, M. (1951). Sur les tableaux de corrélation dont les marges sont données. *Annales de l'Université de Lyon Section A*, **14**, 53–77.

Hawkins, D. L. (1989). Using u statistics to derive the asymptotic distribution of fisher's z statistic. *American Statistician*, **43**(4), 235–237.

Hoeffding, W. (1994). Scale—invariant correlation theory. In F. N.I. and P. Sen, editors, *The Collected Works of Wassily Hoeffding.*, chapter 10, pages 57–107. Springer-Verlag, New York.

Kang, S.-H. and Jung, S.-H. (2001). Generating Correlated Binary Variables with Complete Specification of the Joint Distribution. *Biometrical Journal*, **43**(3), 263–269.

Korf, B. R. (2004). Basic genetics. *Primary Care: Clinics in Office Practice*, **31**(3), 461–478.

Lo, A. (2019). Demystifying the integrated tail probability expectation formula. *The American Statistician*, **73**(4), 367–374.

Madsen, L. and Birkes, D. (2013). Simulating dependent discrete data. *Journal of Statistical Computation and Simulation*, **83**(4), 677–691.

Mayo, O. (2008). A century of Hardy-Weinberg equilibrium. *Twin Research and Human Genetics*, **11**(3), 249–256.

Montana, G. (2005). HapSim: A simulation tool for generating haplotype data with pre-specified allele frequencies and LD coefficients. *Bioinformatics*, **21**(23), 4309–4311.

Morris, T. P., White, I. R., and Crowther, M. J. (2019). Using simulation studies to evaluate statistical methods. *Statistics in Medicine*, **38**(11), 2074–2102.

Myers, R. H. (2004). Huntington's Disease Genetics. *NeuroRx*, **1**(2), 255–262.

Pagon, R. A., Hanson, N. B., Neufeld-Kaiser, W., and Covington, M. L. (2001). Genetic testing. *Western Journal of Medicine*, **174**(5), 344–347.

Rader, D. J. J. (2010). *Deterministic Operations Research - Models and Methods in Linear Optimization*. John Wiley & Sons Inc., Hoboken, New Jersey.

Visscher, P. M., Brown, M. A., McCarthy, M. I., and Yang, J. (2012). Five years of GWAS discovery. *American Journal of Human Genetics*, **90**(1), 7–24.