Utah State University
DigitalCommons@USU

All Graduate Theses and Dissertations | Graduate Studies

5-2008

Small Sample Methods for the Analysis of Clustered Binary Data

Lawrence J. Cook
Utah State University

Follow this and additional works at: http://digitalcommons.usu.edu/etd

Part of the Statistics and Probability Commons

Recommended Citation
Cook, Lawrence J., "Small Sample Methods for the Analysis of Clustered Binary Data" (2008). *All Graduate Theses and Dissertations.* Paper 1.

This Dissertation is brought to you for free and open access by the Graduate Studies at DigitalCommons@USU. It has been accepted for inclusion in All Graduate Theses and Dissertations by an authorized administrator of DigitalCommons@USU. For more information, please contact dylan.burns@usu.edu.

UtahStateUniversity
MERRILL-CAZIER LIBRARY

SMALL SAMPLE METHODS FOR THE ANALYSIS OF CLUSTERED BINARY DATA

by

Lawrence J. Cook

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

of

DOCTOR OF PHILOSOPHY

in

Statistics

Approved:

Dr. Christopher C. Corcoran Major Professor	Dr. D. Richard Cutler Committee Member
Dr. John R. Stevens Committee Member	Dr. David E. Brown Committee Member
Dr. JoAnn T. Tschanz Committee Member	Dr. Byron R. Burnham Dean of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2008

Copyright © Lawrence J. Cook 2008

All Rights Reserved

ABSTRACT

Small Sample Methods for the Analysis of Clustered Binary Data

by

Lawrence J. Cook, Doctor of Philosophy

Utah State University, 2008

Major Professor: Dr. Christopher C. Corcoran
Department: Mathematics and Statistics

There are several solutions for analysis of clustered binary data. However, the two most common tools in use today, generalized estimating equations and random effects or mixed models, rely heavily on asymptotic theory. However, in many situations, such as small or sparse samples, asymptotic assumptions may not be met. For this reason we explore the utility of the quadratic exponential model and conditional analysis to estimate the effect size of a trend parameter in small sample and sparse data settings. Further we explore the computational efficiency of two methods for conducting conditional analysis, the network algorithm and Markov chain Monte Carlo. Our findings indicate that conditional estimates do indeed outperform their unconditional maximum likelihood counterparts. The network algorithm remains the fastest tool for generating the required conditional distribution. However, for large samples, the Markov chain Monte Carlo approach accurately estimates the conditional distribution and is more efficient than the network algorithm.

(213 pages)

This thesis is dedicated to my family. Mom and dad thank you for your unending support and unconditional love, even when I didn't deserve it. Ainsley and Zoe, you are the light of my life. You continuously amaze and impress me. I hope you someday realize how proud I am of both of you. Kei, you are my best friend and partner forever. I couldn't have made it without you.

ACKNOWLEDGMENTS

I would like to thank my committee members for their help and suggestions on this project. I would like to especially thank Dr. Chris Corcoran, my major professor. Thank you for your introduction to this problem and your guidance throughout the process and many hours of reading. Thank you Dr. Tschanz for providing data sets, analyses, and support to help me throughout my time at Utah State University. I would also like to thank Drs. Brown, Cutler, and Stevens for serving on my committee.

I am grateful for my many colleagues at the Intermountain Injury Control Research Center who supported me throughout this process. Lenora Olson deserves special mention for making it possible for me to return to school and her never-ending encouragement and support. Special thanks to Stacey Knight and Amy Donaldson for always providing a positive word and for always being there when I needed their help.

I would like to thank the faculty, staff, and students in the Department of Mathematics and Statistics for making the past five years an enjoyable experience. I would like to thank Cindy Moulton for helping me navigate all the processes and procedures.

Thank you to my family, Kei, Ainsley, and Zoe. You are my inspiration and I love you all very deeply.

Lawrence J Cook

CONTENTS

Page

ABSTRACT iii

ACKNOWLEDGMENTS v

LIST OF TABLES ix

LIST OF FIGURES xi

1 INTRODUCTION 1
- 1.1 Examples 1
  - 1.1.1 Example 1: Fetal Outcomes 2
  - 1.1.2 Example 2: Airbags 2
  - 1.1.3 Example 3: Malformation in Mice Fetuses 3
  - 1.1.4 Example 4: Corneal Grafts 3
- 1.2 Analyzing Correlated Categorical Outcomes 5
  - 1.2.1 Generalized Linear Models for Independent Data 6
  - 1.2.2 Population Average 8
  - 1.2.3 Cluster Specific Methods 10
  - 1.2.4 Conditional Models 12
  - 1.2.5 Comparison of Approaches 14
- 1.3 Analysis of Examples 16
  - 1.3.1 Example 1: Fetal Outcomes 16
  - 1.3.2 Example 2: Airbags 20
  - 1.3.3 Example 3: Malformation in Mice Fetuses 22
  - 1.3.4 Example 4: Corneal Graphs 23

2 ESTIMATION APPROACHES FOR SMALL OR SPARSE SAMPLES . . 25
- 2.1 Exact Logistic Regression for Independent Binomials 26
- 2.2 Conditional Exact Estimation 27
- 2.3 An Exact Framework for Clustered Binomials 30
  - 2.3.1 Conditional Likelihood of the QEM 31
  - 2.3.2 Conditional Estimation 32

3 COMPUTATIONAL METHODS FOR CONDITIONAL ESTIMATION . . 35
- 3.1 The Network Algorithm 35
- 3.2 Markov Chain Monte Carlo 43
  - 3.2.1 Introduction 43

3.2.2 Random Walk on Contingency Tables with Fixed Row and Column Sums . . . 46
3.2.3 Conditional Likelihood and a . . . 47
3.2.4 Random Walk on the Quadratic Exponential Model . . . 49
3.2.5 Parameter Estimation . . . 50
3.3 Analysis of Examples . . . 51
3.3.1 Toxicology Data . . . 51
3.3.2 Corneal Grafts . . . 55

4 SMALL SAMPLE PROPERTIES OF PARAMETER ESTIMATES . . . 59
4.1 Simulation Settings . . . 59
4.2 Parameter Estimate Properties . . . 60
4.2.1 $x \in \{0, 1\}$ . . . 61
4.2.2 $x \in \{0, 1, 2\}$ . . . 63
4.2.3 $x \in \{0, 1, 2, 3\}$ . . . 65
4.3 Confidence Intervals . . . 67
4.3.1 Confidence Interval Coverage . . . 67
4.3.2 Confidence Interval Length . . . 68
4.3.3 Discussion . . . 71

5 ACCURACY AND COMPUTATIONAL EFFICIENCY OF MCMC . . . 74
5.1 Simulation Settings . . . 74
5.2 MCMC Distribution Accuracy . . . 75
5.3 Computational Efficiency . . . 77
5.4 Computational Burden of Rejection Sampling . . . 81
5.5 Discussion . . . 81

6 SUMMARY . . . 84

REFERENCES . . . 87

APPENDICES . . . 93
APPENDIX A PROGRAMS FOR PARAMETER ESTIMATION FROM THE NETWORK ALGORITHM . . . 94
A.1 RunAllNoRecurse.txt - R program for calculating network algorithm . 95
A.2 clustexampmod.c - C program for gaining basic information prior to network calculation [8] . . . 110
A.3 trend.c - C program for calculating network algorithm [8] . . . 114
A.4 simc.txt - R program for generating random samples from a quadratic exponential distribution and reading the results of the network algorithm for parameter estimation . . . 143
A.5 MUEFalse.txt - R program for calculating median unbiased estimate . 150
A.6 CIFalse.txt - R program for calculating $(1 - \alpha)\%$ confidence intervals 157
APPENDIX B PROGRAMS FOR MARKOV CHAIN MONTE CARLO ESTIMATION . . . 165

B.1 readsim.cpp - C program for MCMC estimation of the distribution of the trend parameter 166
B.2 parameterestimate.cpp - C program for calculating MCMC parameter estimates 183

VITA 198

LIST OF TABLES

Table		Page
1.1	Fetal Outcomes by Crash Status	2
1.2	Injury Status for Front Seat Passengers by Presence of an Airbag	3
1.3	Malformations in Mice Fetuses	4
1.4	Rejected Implants in Children with CHED	5
1.5	Parameter Estimates from Maternal Crash Data	17
1.6	Odds Ratios for a Woman Not in a Crash Compared to a Woman Having a Crash Experiencing an Additional Low Weight Birth Given a Specified Number of Births and Previous Low Weight Births	19
1.7	Parameter Estimates from the Airbag Crash Data	21
1.8	Parameter Estimates from the Mice Malformation Data	22
1.9	Parameter Estimates from the Corneal Graft Data	23
3.1	Example Sample of Clustered Data	36
3.2	Example Sample of Clustered Data	43
3.3	Probability of Obtaining Each Table in the Reference Set from Table 3.1	45
3.4	MCMC Relative Frequency Obtained for Each of the Tables in the Reference Set from Table 3.1	46
3.5	Parameter Estimates from the Mice Malformation Data	52
3.6	Results of 20 MCMC Runs for the Data in Table 3.1	54
3.7	P-values for Several Choices of β for the Toxicology Data in Table 1.3	55
3.8	Exact Permutation Distribution for $t = \sum z$	56
3.9	Parameter Estimates from the Corneal Graft Data	56

3.10 Results of 20 MCMC Runs for the Data in Table 1.4 58

3.11 P-values for Several Choices of β for the corneal grafts data in Table 1.4 58

5.1 Proportion of 95%Confidence Intervals Not Containing the True p-value 77

5.2 Summary Statistics for Run Times (in Seconds) of the Network Algorithm 78

5.3 Summary Statistics for Run Times (in Seconds) of the MCMC Method 78

LIST OF FIGURES

Figure Page

1.1 Fifty Samples of the Estimated Probability of a Low Weight Birth from the Estimates in Table 1.5. 18

3.1 Network Representation of $\Gamma(s_1)$ for the Data in Table 3.1. 41

3.2 Network Representation of $\Gamma(s_1, s_2)$ for the Data in Table 3.1. 42

3.3 Exact and Estimated MCMC Conditional Distribution of T. 53

4.1 Parameter Estimates of β and Percent of Nonestimable Samples When $x \in \{0, 1\}$. 62

4.2 Parameter Estimates of β and Percent of Nonestimable Samples When $x \in \{0, 1, 2\}$. 64

4.3 Parameter Estimates of β and Percent of Nonestimable Samples When $x \in \{0, 1, 2, 3\}$. 66

4.4 Histograms of the Exact Confidence Interval Coverage. 69

4.5 Histograms of the MLE Confidence Interval Coverage. 70

4.6 Neyman Shortness Distance for Exact and MLE Confidence Intervals. 72

5.1 QQ Plot of P-values from Goodness-of-Fit Tests vs. U(0,1) by Number of Clusters. 76

5.2 Box Plot of Log Base 10 Run Times for Network and MCMC Distribution Calculations. 80

5.3 Scatter Plot of log(Sample Space Ratio) by log(Run Time). 82

CHAPTER 1

INTRODUCTION

Correlated binary outcomes arise frequently in applied research. In ophthalmologic studies, measurements are often taken simultaneously on both eyes of an individual. Similar environmental and genetic exposures make it likely that measurements on the two eyes will be more similar compared to measurements taken on eyes from separate individuals. In the study of genetically transmitted diseases, measurements taken on siblings can be considered a cluster and are likely to have correlated outcomes. In longitudinal studies, subjects are tracked over time with measurements taken periodically. It is highly likely that observations taken from the same subject will be more alike than observations taken from different subjects.

Ignoring dependency between correlated (or clustered) observations can lead one to underestimate standard errors and overstate significance. This phenomenon is often referred to as over-dispersion: subjects in a cluster exhibit less variation than expected, while the variation between clusters is larger than expected. This effect has been demonstrated both mathematically and through examples [21, 28, 30, 53].

1.1 Examples

The following four examples illustrate common occurrences with clustered data. These examples motivate our subsequent discussion of current methods for analyzing correlated binary data. We analyze the four examples using these methods, and discuss how each approach affects the results and interpretations.

Table 1.1: Fetal Outcomes by Crash Status

	Number Low Birth Weight									
	No Crash					Crash				
Number Born	0	1	2	3	4	0	1	2	3	4
Twins	1,131	732	1,458	-	-	21	26	41	-	-
Triplets	0	2	3	102	-	0	0	0	3	-
Quads	0	0	0	0	4	-	-	-	-	-

1.1.1 Example 1: Fetal Outcomes

Does experiencing a motor vehicle crash during pregnancy have a negative impact on birth outcomes? Studies have previously shown that women involved in a crash during pregnancy were at risk of delivering low-weight infants [25]. However, due to correlations among multiple births, all twins and triplets were excluded from the original analysis. The data for multiple births are presented in Table 1.1. Table 1.1 shows there is a slight trend for women in a crash being more likely to have low birth weight children, with 45% of twins not in crashes having normal weights and only 39% of twins in crashes having normal birth weights. However, analysis is complicated by the correlation between children born to the same mother.

1.1.2 Example 2: Airbags

Does the combination of an airbag and seatbelt provide more protection in a head-on collision than seatbelts alone? Table 1.2 contains injury information (hospitalized or killed vs. not hospitalized or killed) for front seat adult crash participants involved in a head-on or a single-vehicle, fixed-object crash in Utah during 2004, stratified by whether or not the vehicle was equipped with an airbag. Table 1.2 shows there is a slight increase in the percent of occupants injured or killed for vehicles without airbags (17% vs. 15%). However, analysis of these data should account for

Table 1.2: Injury Status for Front Seat Passengers by Presence of an Airbag

	# Passengers	Not Injured	Hosp. or Killed	Total
Airbag Not Present	1	1,234	250	1,484
	2	655	89	744
	3	30	0	30
	4	4	0	4
Airbag Present	1	4,611	704	5,315
	2	3,196	446	3,462
	3	42	3	45
	4	16	0	16
Total		9,788	1,492	11,280

correlation between passengers. Occupants of the same vehicle are subjected to many of the same forces (e.g. speed and direction of impact) throughout the crash.

1.1.3 Example 3: Malformation in Mice Fetuses

The data in Table 1.3 are from Bradstreet and Liss [5]. One hundred female mice were randomized to either control or one of three dose levels (8, 80, or 800 mg/kg) of a potentially harmful drug, then mated with untreated male mice. Twenty-five mice were assigned to each category; however not all were impregnated. At day seventeen of gestation, the animals were sacrificed and their offspring observed for malformations. Table 1.3 displays, for each mouse, the number of malformations and offspring. Summary statistics for this table show that the response rate does increase with dosage, 1.4% to 3.0%. As in the first example, litter mates are likely to exhibit correlated outcomes. However, this data set poses an additional complication. The occurrence of malformations appears to be rare. The large number of litters with no malformations may cause large sample assumptions to fail.

1.1.4 Example 4: Corneal Grafts

Table 1.4 shows the results of corneal grafts for nine children categorized into

Table 1.3: Malformations in Mice Fetuses

Dose (mg/kg)			
0	8	80	800
0/7	0/10	0/11	0/11
0/8	3/10	0/11	0/11
0/9	0/11	0/11	0/11
0/10	0/11	0/11	2/11
0/10	0/12	0/12	0/12
0/11	0/12	0/12	1/12
0/11	0/12	0/12	0/13
0/11	0/12	1/12	0/13
1/11	0/12	2/12	0/13
0/12	0/12	0/13	0/13
0/12	0/13	0/13	1/13
1/12	0/13	0/13	2/13
0/13	0/13	1/13	0/14
0/13	1/13	0/14	0/14
0/13	0/14	0/14	1/14
0/13	0/14	1/14	1/14
1/13	0/14	0/15	0/15
0/14	1/14	1/15	1/15
0/14	1/14	2/15	0/16
0/14	0/15	0/16	0/16
0/14	0/15	0/16	1/16
1/14	0/15	0/16	0/17
0/16	0/15	0/16	0/17
0/16	0/15		0/17
	0/17		

Table 1.4: Rejected Implants in Children with CHED

Age at Diagnosis (years)	
≤ 3	≥ 3
0/2	0/2
0/2	1/2
0/2	1/2
0/2	1/1
	1/1

two age groups. The data were collected as part of a study to assess the impact of potential risk factors on the success of corneal implants to correct the loss of visual acuity resulting from congenital hereditary endothelial dystrophy (CHED). Seven of the children received implants in both eyes, while two received implants in only one eye [48]. Not only are outcomes of eyes from the same individual likely to be correlated, this data set has additional features that complicate the analysis. The first complication is sample size. Many methods rely on asymptotic theory for inference. The second complication with these data is the fact that all rejections come from the older group, which prevents the use of conventional methods to conduct inference for an age effect.

1.2 Analyzing Correlated Categorical Outcomes

A number of approaches have been developed to accommodate over-dispersion caused by clustering. However, unlike continuous, normally distributed outcomes where the choice of modeling approach only changes the correlation structure, with binomial data the modeling approach changes the interpretation of the parameters. The most common approaches have interpretations which can be termed as "population averaged," "subject-specific," and "conditional."

For the remainder of this discussion we will assume that Y is a binary outcome. Further, we will assume that observations have been taken on N clusters or individuals

and that each cluster or individual has $n_i, i = 1, \ldots, N$, observations. The total number of observations is $\sum_{i=1}^{N} n_i$ or $N * n$ if $n_1 = \cdots = n_N = n$. The data point Y_{ij} refers to the jth observation taken from cluster or subject i, and y_{ij} refers to a specific instance of the variable Y_{ij}.

1.2.1 Generalized Linear Models for Independent Data

Before discussing current methods for the analysis of clustered binary data, we briefly review generalized linear models (GLM) and how logistic regression arises within the GLM framework. For a detailed discussion see McCullagh and Nelder [34] and Agresti [1, chapters 4 and 5].

GLMs refer to a broad class of regression problems arising from the exponential family. In the context of GLMs, one must be able to write the probability mass or density function for y_i as

$$f(y; \theta_i, \phi) = \exp\big([y_i\theta_i - b(\theta_i)]/a(\phi) + c(y_i, \phi)\big). \tag{1.1}$$

Formulating the distribution in this fashion has many useful properties. The most useful perhaps is that the mean and variance are specified; i.e.

$$\mu_i = E(Y_i) = b'(\theta_i) \tag{1.2}$$

$$\mathrm{var}(Y_i) = b''(\theta_i)a(\phi). \tag{1.3}$$

The analysis of GLMs usually proceeds by modeling θ, the canonical parameter, as a linear function of model parameters, i.e. $\theta_i = \alpha + \beta x_i$.

Letting $Y_i \sim \mathrm{B}(n_i, \pi_i)$, Y_i can be formulated as a GLM,

$$
\begin{aligned}
P[Y_i = y_i] &= \binom{n_i}{y_i} \pi^{y_i} (1-\pi_i)^{n_i - y_i} \\
&= \exp\left[\frac{y_i\theta_i - \log[1+\exp(\theta_i)]}{1/n_i} + \log\binom{n_i}{y_i}\right] \\
&\text{where } \theta_i = \log\frac{\pi_i}{1-\pi_i}.
\end{aligned} \tag{1.4}
$$

In Equation (1.4) θ is the log odds ratio. Formulating the model in this fashion has come to be known as "logistic regression." In addition to being the canonical parameter for a binomial GLM, modeling the log odds has another advantage over modeling the probability, π_i, directly. Namely, the odds ratio is unconstrained on the log scale whereas probabilities are constrained to the interval [0,1].

Maximum likelihood methods can be used to find model parameters. For GLMs the likelihood equations are,

$$\sum \frac{\partial \mu_i}{\partial \beta} \frac{y_i - \mu_i}{\mathrm{var}(Y_i)} = 0. \tag{1.5}$$

In the context of the logistic regression model above the likelihood equations become

$$\frac{\partial \pi_i}{\partial \beta} \frac{y_i - \pi_i}{\pi_i(1-\pi_i)} = 0. \tag{1.6}$$

Equations (1.5) and (1.6) can be solved using methods such as Newton-Raphson or Fisher Scoring.

Model parameters in logistic regression also have a useful interpretation for researchers. For the model $\theta_i = \alpha + \beta x_i$, $\exp(\beta)$ can be interpreted as the multiplicative change in odds for a positive outcome in y_i for a unit change in x_i.

1.2.2 Population Average

Population-averaged or marginal models are useful when the goal of the research is to answer questions about differences in the population, such as the average difference between the control and treatment groups in a clinical trial. The most common method of analyzing data to obtain population-averaged estimates is generalized estimating equations (GEE). GEEs model the population average, $\mathrm{E}[Y_{ij}] = \mu_{ij}$. For this reason, parameters from this model estimate differences between groups across clusters, and not effects within clusters.

Liang and Zeger were the first to use GEEs to analyze correlated binary outcomes [31]. This method is an extension of the quasi-likelihood techniques for dependent data [51]. Rather than being concerned with the likelihood as in Section 1.2.1, the GEE approach relaxes some of the requirements of GLMs. The GEE model makes three basic assumptions:

1. The marginal expectation of the response, $\mathrm{E}[Y_{ij}] = \mu_{ij}$, depends on explanatory variables, $\mathbf{x_{ij}}$, by a link function, $h(\mu_{ij}) = \mathbf{x_{ij}}'\beta$, where h is known. The logit link is the common choice for binary outcomes.

2. The marginal variance depends on the marginal mean according to a known variance function, $\mathrm{Var}(Y_{ij}) = v(\mu_{ij})\phi$. ϕ is a scale parameter and can be estimated.

3. The correlation between two observations in the same cluster, Y_{ij} and Y_{ik} is a function of the marginal means and possibly of additional parameters α, e.g. $\mathrm{Corr}(Y_{ij}, Y_{ik}) = \rho(\mu_{ij}, \mu_{ik}; \alpha)$ where ρ is known.

Estimation for GEEs proceeds by making slight modifications to the likelihood equations in Equation (1.5). The variance portion of Equation (1.5) is replaced by a

working correlation matrix. Using matrix notation, the likelihood can be written as

$$\sum_{i=1}^{N} D_i^T A_i^{-1}(y_i - \pi_i) = 0, \tag{1.7}$$

where D_i is a matrix of partial derivatives, $\frac{\partial \pi_i}{\partial \beta}$, A_i is an n_i x n_i matrix representing the variance of $\mathbf{y_i}$. GEEs replace the variance matrix, A_i, with working correlation matrix, V_i,

$$\sum_{i=1}^{N} D_i^T V_i^{-1}(y_i - \pi_i) = 0,$$

where

$$V_i = A_i^{1/2} R_i(\alpha) A_i^{1/2}/\phi.$$

Liang and Zeger show that estimates obtained using this model are asymptotically consistent and follow a normal distribution assuming the working correlation is correct [31]. One can replace V_i by a robust variance estimate which will produce consistent estimates even when the correlation, $R(\alpha)$, is misspecified. This is known as the "sandwich estimator"

$$\text{cov}(\hat{\beta}) = \left(\sum \hat{D}^T \hat{V}^{-1} \hat{D}\right)^{-1} \left[\sum \hat{D}^T \hat{V}^{-1} S \hat{V}^{-1} \hat{D}\right] \left(\sum \hat{D}^T \hat{V}^{-1} \hat{D}\right)^{-1} \tag{1.8}$$

where

$$S = (y_i - \hat{\pi}_i)^T (y_i - \hat{\pi}_i).$$

[47, 52].

An interesting feature of GEEs is that the model parameters are not necessarily tied to the parameters of the correlation structure. The researcher is free to model each separately. It is possible to change either the model or the correlation structure without changing the other.

GEEs are flexible in that many different correlation structures can be accommodated [31]. Some of the common correlation structures implemented in commercial packages include independence, exchangeability, autoregressive, and unstructured. An independent correlation assumes that there is no correlation between observations in the same cluster, which reduces to ordinary logistic regression. Exchangeable correlation assumes that correlation between any two observations in the same cluster is constant across clusters. Autoregressive correlation allows for correlation between successive observations to be constant within and across clusters but to decrease with distance. Such a pattern is common for repeated measures. Finally, unstructured correlation puts no assumptions on the correlation structure and estimates each component separately.

One difficulty in formulating GEE for binomial data is that the correlations are constrained by the marginal probabilities in complicated ways [16]. Additionally, correlations are constrained to the interval [-1,1]. For this reason Liang, Zeger, and Qaqish [32] and Lipsitz, Laird, and Harrington [33] suggest modeling the dependency in terms of odds ratios. The benefits of this approach are that odds ratios are not constrained to [-1,1], and are easier to interpret than correlations.

1.2.3 Cluster Specific Methods

Rather than modeling the marginal response, cluster-specific methods treat the probability distribution of Y_{ij} as function of covariates, $\mathbf{X_{ij}}$ and parameters α_i specific to each cluster. Common approaches include random effects models [3, 49] and conditional likelihood [6, 7]. Parameters from these models are most useful when the goal of the research is to make inferences about variables on the individual or cluster level.

The most common method for gaining cluster, or subject-specific, estimates is

random or mixed effects models. Such an approach assumes that heterogeneity between clusters arises due to some unobservable variable which can be represented by a probability distribution. Conditional on a realization from such a distribution, observations in a cluster are mutually independent. The simplest random effects model, known as the "random intercepts model," allows each cluster to have its own intercept,

$$\text{logit}\big[\text{P}(Y_{ij} = 1|U_i)\big] = (\beta_0 + U_i) + \beta_1 x_{ij}. \tag{1.9}$$

This model assumes that differences between clusters arise from each having a different baseline probability, $\frac{\exp(\beta_0+U_i)}{1+\exp(\beta_0+U_i)}$. It also assumes that once this baseline probability is known, the odds ratio of a positive outcome given the covariate x is constant across clusters, $\exp(\beta_1)$. This result is best demonstrated in Diggle [16, Chapter 7]. More complicated mixed models can be constructed by allowing covariates to have random effects as well.

In addition to conditional independence within a cluster given U_i, random effects models make two other main assumptions. Given U_i, the responses $Y_{i1}, \ldots, Y_{in_i}$ follow a GLM with density $f(y_{i,j}|U_i)$ of form Equation (1.1), with conditional mean, $u_{ij} = \text{E}(Y_{ij}|U_i) = b'(\theta_i)$, and conditional variance, $\text{Var}(Y_{ij}) = v_{ij} = b''(\theta_i)a(\phi)$. The final assumption is that the random effects, $U_i, i = 1, \ldots, N$, are mutually independent with a common underlying multivariate distribution, $f(U_i; G)$.

Calculation of parameter estimates typically involves maximum likelihood methods. The likelihood function for the unknown parameter, δ, which includes both β and the elements of G is

$$L(\delta; y) = \prod_{i=1}^{N} \int \prod_{j=1}^{n_i} f(y_{ij}|U_i, \beta) f(U_i, G) dU_i. \tag{1.10}$$

Numerical integration methods can be used to evaluate Equation (1.10). However, this can be computationally intensive. Many approaches have been introduced to speed up computations, including REML [3], Gibbs sampling [27, 55], and Gaussian quadrature [43, 44].

While this model does not impose a prespecified assumption about the distribution of the random effects, U_i, the most common random effects model for binomial outcomes is the logistic-normal model, where one assumes $U_i \sim \mathrm{N}(0, \sigma^2)$. In this model, σ^2 represents the amount of heterogeneity between the clusters. Large values of σ^2 represent a greater degree of heterogeneity between clusters and small values of σ^2 correspond to samples with clusters that are more homogeneous. By setting $\sigma^2 = 0$, the logistic-normal model reduces to ordinary logistic regression.

1.2.4 Conditional Models

Conditional models are a third approach to modeling correlated binary outcomes. Rather than modeling the odds ratios, however, the response-conditional approach models the probability of the sample. This approach is an extension of a model proposed by Cox [10] and formalized as a loglinear model by Bishop, Fienberg, and Holland [4]. The joint distribution for a single cluster $\mathbf{Y}$ is defined as

$$P[\mathbf{Y} = \mathbf{y}] = c(\theta) \exp\Big(\sum_{j=1}^{n} \theta_j^{(1)} y_j + \sum_{j1<j2} \theta_{j1j2}^{(2)} y_{j1} y_{j2} + \cdots + \theta_{1\ldots n} y_1 \ldots y_n \Big). \tag{1.11}$$

The function c(θ) is a constant that normalizes the density to sum to one. This model is conditional in the sense that its parameters have interpretations in terms of conditional probabilities. One unique characteristic of the conditional model is that it has an exponential family form. Note that if all second and higher order terms are set to zero, this reduces to the ordinary logistic regression model of Section 1.2.1

A much studied form of Equation (1.11) is the quadratic exponential model (QEM) [12, 41, 42, 57]. The QEM is derived by fixing all third and higher order correlations to be zero, yielding a joint distribution of the form

$$P[\mathbf{Y} = \mathbf{y}] = c(\theta) \exp\Big(\sum_{j=1}^{n} \theta_j^{(1)} y_j + \sum_{j<k} \theta_{jk}^{(2)} y_j y_k \Big).$$

For this model, the log odds of a response between observations Y_{ij} and Y_{ik}, given that all other observations in the cluster are zero, is computed as

$$\log\left(\frac{P[Y_j = 1 | Y_k = y_k, Y_l = 0, l \neq j, k]}{P[Y_j = 0 | Y_k = y_k, Y_l = 0, l \neq j, k]} \right) = \theta_j^{(1)} + \theta_{jk}^{(2)} y_k.$$

Thus, $\theta_j^{(1)}$ is the log odds for $Y_i = 1$ given that the remaining responses in the cluster are all zero. Similarly, $\theta_{jk}^{(2)}$ is the log odds describing the association between Y_i and Y_j given that all the other responses are fixed and set to zero. As demonstrated by Molenberghs and Ryan [42], assuming exchangeable correlation and rewriting the distribution in terms of the number of "successes," Z_i, in cluster i, this distribution can be expressed as

$$f_Z(Z_i; \Theta_i, n_i) = \exp\{\theta \; z_i^{(1)} + \delta z_i^{(2)} - A(\Theta_i)\}, \tag{1.12}$$

where $z_i^{(1)} = z_i = \sum y_i$ and $z_i^{(2)} = -z_i(n_i - z_i)$ [42]. In this model $\delta = 0$ corresponds to independence between observations in the same cluster, $\delta > 0$ corresponds to over-dispersion, and $\delta < 0$ corresponds to under-dispersion.

Under this formulation, the conditional logit of an additional success given $z - 1$ successes is

$$\text{logit}\big(P[y_{ik} = 1 | z_i - 1, n_i]\big) = \theta_i + \delta(2z_i - n_i - 1). \tag{1.13}$$

Equation (1.13) demonstrates that the conditional logit of an additional success is $\theta_i + \delta(2z_i - n_i - 1)$; thus when $z_i = \frac{n_i - 1}{2}$, θ_i is the conditional logit of an additional success when about half the cluster are successes. The log odds ratio for responses between two members of the cluster can also be shown to be 2δ.

Parameters in the QEM can be found by using maximum likelihood; however, this can be computationally inefficient because the constant term $\mathrm{c}(\theta)$ needs to be recalculated at each step [46].

1.2.5 Comparison of Approaches

These three approaches lead to different substantive interpretations with respect to regression parameters. GEEs have the advantage of estimating the marginal mean, allowing parameters to have the analogous interpretations as those obtained from logistic regression. A limitation with GEEs, is the focus on the marginal means does not allow estimation of intra-cluster effects. Additionally, Diggle [16] argues that when there is a great amount of heterogeneity between clusters, a cluster specific model is more appropriate. Further, the reliance of GEEs on quasi-likelihood methods eliminates many useful tools for analyzing GLMs, such as likelihood ratio and goodness-of-fit statistics.

Mixed effects models, on the other hand, do rely on likelihood-based methods, thus bringing the full complement of GLM tools to bear. However, parameters have interpretations on the cluster or subject level. Furthermore, one must make an additional assumption regarding the specific distribution of the random effects.

Neuhaus [43], Neuhaus and Kalbfleisch [44], and Zeger, Liang, and Albert [56] have explored the relationship between parameter estimates from the different models: population average (PA), cluster specific (CS), and conditional (COND). Under the assumption that the random effects take on a $\mathrm{N}(0, \sigma^2)$ distribution, Zeger, Liang, and

Albert [56] show that

$$\beta^{PA} = [(16\sqrt{3})/(15\pi)^2\sigma^2 + 1]^{1/2}\beta^{CS} \approx (0.346\sigma^2 + 1)^{-1/2}\beta^{CS}.$$

Neuhaus [43] and Neuhaus and Kalbfleisch [44] proved the more general result that regardless of the distribution of the random effects, the following relationships hold,

1. $|\beta^{COND}| \leq |\beta^{PA}| \leq |\beta^{CS}|$.
2. the difference between β^{CS} and β^{PA} increase with σ^2.
3. $|\beta^{COND}| = |\beta^{CS}| = |\beta^{PA}|$ when there is no inter-cluster correlation or when there is no effect, i.e $|\beta^{COND}| = |\beta^{CS}| = |\beta^{PA}| = 0$.

Given these relations and that it is possible to gain estimates for parameters that don't vary within clusters, it may be tempting to take advantage of the likelihood approach of mixed models. However, Diggle [16] and Neuhaus [43] argue that parameters estimated by the cluster specific means cannot be interpreted in terms of marginal means, since the model was not designed to estimate them as such.

Finally, it should be noted that conditional models have received criticism from several authors [16, 43, 44]. Criticisms fall into three main categories. Interpretation of the model parameters depends on the observed responses in each cluster [16, 43, 44]. Interpretation of the model parameters depends on the cluster size [16, 43, 44]. Interpretation of an individual parameter depends on the values of the other parameters [43, 44].

Zhao and Prentice address the first two points by marginalizing the parameters from the QEM and using GEEs [57]. Fitzmaurice and Laird use likelihood methods and a mixed models approach for solving QEMs [17]. However, Neuhaus still argues against conditional models due to the third criticism above [44, 43]. Despite Neuaus's

criticisms, Molenberghs and Ryan believe that conditional models offer certain advantages over GEEs and mixed models [42], namely:

1. efficiency, [2, 41]
2. the ability to model the association structure, e.g. it is the only method which provides the ability to calculate the probability that at least one member of the cluster has a positive outcome, [42]
3. the model's ability to test for no association [2, 41, 42]
4. the ability to facilitate exact analysis for small sample sizes [9].

1.3 Analysis of Examples

In this section, we analyze each example from Section 1.1 using GEE, random effects, and QEM. For comparison, we also use ordinary logistic regression. GEE, random effects, and logistic regression estimates were respectively calculated using PROC GENMOD, PROC NLMIXED, and PROC LOGISTIC in SAS. QEM fits were obtained using the software package R. For each example, we compare and contrast results between these four methods.

1.3.1 Example 1: Fetal Outcomes

Table 1.1 shows there is a slight trend for women in a crash being more likely to have low birth weight children. Results summarized in Table 1.5 demonstrate the consequences of ignoring clustering. While the point estimates from logistic regression and GEE are nearly the same, the standard errors are quite different. This is due to the correlation between children from the same mother. Table 1.5 shows the estimated correlation between children of the same mother is 0.57, indicating a strong positive relationship. Thus children from the same mother tend to have low birth weights

Table 1.5: Parameter Estimates from Maternal Crash Data

	Logistic Regression		GEE		Random Effects		QEM	
Parameter	Estimate	SD	Estimate	SD	Estimate	SD	Estimate	SD
Intercept	0.28	0.02	0.26	0.03	0.53	0.06	0.18	0.02
Crash	0.26	0.15	0.26	0.18	0.50	0.37	0.16	0.12
Dispersion	NA	NA	0.57	-	7.00	0.53	1.30	0.04

together or normal birth weights together. Focusing on interpreting the parameters from the GEE, population-averaged conclusions can be made. The intercept implies that, on average, a child has a baseline probability for being low birth weight of $\frac{\exp(0.26)}{1+\exp(0.26)} = 0.56$. However, children who experience a crash in-utero are $\exp(0.26) = 1.3$ times more likely to be low weight compared to children who do not experience a crash in-utero.

While the GEE leads to population-averaged interpretation, the random effects model does not. With this model we are assuming that each woman has an individual probability of delivering a low weight child. This probability, however, depends on some unobservable variable, U. We have imposed the N(0,σ^2) distribution on U. Table 1.5 shows that the estimate of σ^2 to be nearly 7. Since we know that about 95% of the observations fall within $\pm 2\sqrt{7} = 5.2$, there is great heterogeneity between women. Some women will have almost no chance of delivering a low weight child, while other women will almost certainly deliver low weight children. While there is correlation between children born to the same women, the random effects model assumes that once the value of $U = u$ is known for a particular woman, the birth weight of her children are independent. Women with $U = 0$ have a baseline probability of $\frac{\exp(0.53)}{1+\exp(0.53)} = 0.63$. The model we have chosen assumes that experiencing a crash affects all women the same way. The value of the parameter for having a crash can be interpreted as the odds ratio comparing children of women experiencing a crash

to women not experiencing a crash, given that both women have the same value of U. For instance, given a specific value $U = u$, children born to women experiencing a crash are $\exp(0.50) = 1.64$ times more likely have low birth weight compared to children born to women who did not experience a crash. This relationship can be seen in Figure 1.1. Figure 1.1 was generated by taking a sample of size fifty from a

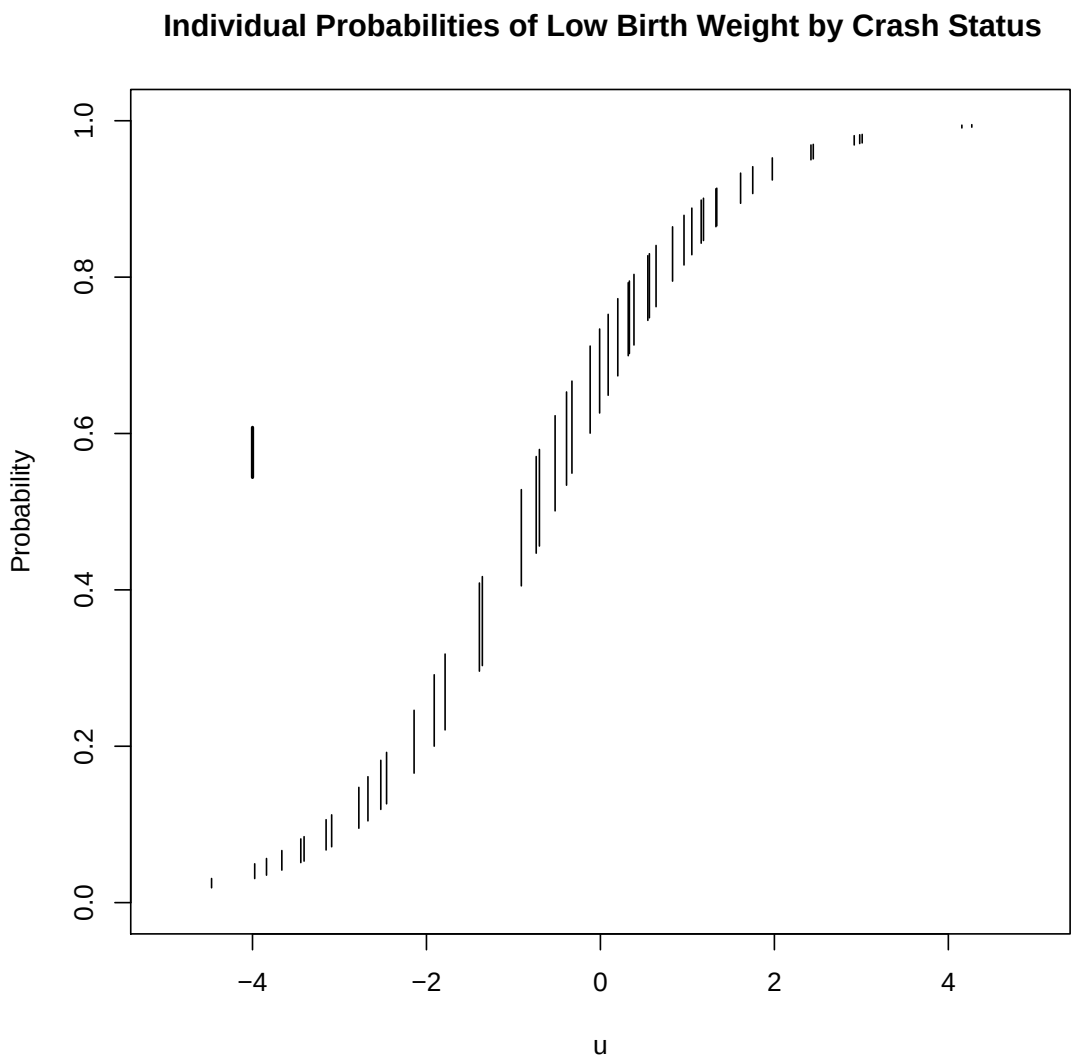

Fig. 1.1: Fifty Samples of the Estimated Probability of a Low Weight Birth from the Estimates in Table 1.5.

Table 1.6: Odds Ratios for a Woman Not in a Crash Compared to a Woman Having a Crash Experiencing an Additional Low Weight Birth Given a Specified Number of Births and Previous Low Weight Births

Number Born	Number Low Weight	Odds No Crash	Odds Crash	Odds Ratio
2	0	0.328	0.386	1.18
	1	4.331	5.099	1.18
3	0	0.090	0.106	1.18
	1	1.192	1.404	1.18
	2	15.74	18.52	1.18

N(0,7) distribution for U and calculating the probability of a low birth weight child for women not experiencing a crash and women experiencing a crash using the model parameters in Table 1.5. The bottom of each line shows the estimated probability of a low weight birth for a given u for women not experiencing a crash, while the top of each line shows the probability of a low weight birth for a given u for women experiencing a crash. Depending on the value of $U = u$, the difference between the two probabilities can range from small to large. However, the value of the odds ratio between the two points remains the same, exp(0.50). The dark line at the left of Figure 1.1 shows the average probabilities for each of the two groups. It is this value the GEE model is estimating. Using the equation presented in Section 1.2.5 we can calculate the marginal estimate, $(0.346 * 6.99 + 1)^{-1/2} * 0.497 = 0.269$, which is very close to the GEE estimate of 0.263.

Interpretation of the QEM model relies on a given number of births (i.e. cluster size). From Equation (1.13), we can calculate conditional odds and odds ratios given a specified number of births and number of other low birth weight children (Table 1.6). Several aspects of the QEM model are apparent from Table 1.6. First is the way an outcome for one child depends on the outcomes of siblings born at the same time. This can be noted in the way the odds for an additional low birth weight

child increase dramatically as the number of other low birth weight children increase. Looking at triplets, if the other two children were of normal weight, then the odds of the third child being low weight are only 0.09. However, if the two other children were low weight then the odds of an additional low weight child are 15.74. The odds ratio of being low weight for two siblings, regardless of crash status, is estimated by $\exp(2\delta) = 13.2 = \frac{1.192}{0.09}$. The other item of interest in Table 1.6 is that once the number of births and number of prior low births has been established, the effect of experiencing a crash is constant, $\exp(0.16) = 1.18$. Therefore, the parameter for crash in this model can be thought of as the increase in log odds of a low weight birth for a child whose mother experienced a crash compared to a child whose mother did not experience a crash, conditioned on a specified number of births and low weight siblings.

Depending on the modeling method, different estimates for the effect of a crash can be obtained. The key to understanding these estimates is understanding the assumptions of the models and how they operate. The random effects model, with the largest estimate, gives each mother a specific estimate of the effect of a crash on the low birth weight of children. The QEM on the other hand gives the lowest estimate. This is due to the strong correlation between outcomes of children born to the same mother and the QEM modeling separate effects for outcomes within clusters and those due the crash, between clusters.

1.3.2 Example 2: Airbags

Results from the different analytic approaches applied to the data in Table 1.2 are presented in Table 1.7. As with the previous example, interpretation depends on which method we consider. GEE yields a population-averaged interpretation of the airbag effect. The GEE model assumes that all passengers have a baseline probability

Table 1.7: Parameter Estimates from the Airbag Crash Data

	Logistic Regression		GEE		Random Effects		QEM	
Parameter	Estimate	SD	Estimate	SD	Estimate	SD	Estimate	SD
Intercept	-1.73	0.06	-1.70	0.06	-3.05	0.13	-1.51	0.57
Airbag	-0.18	0.07	-0.21	0.07	-0.35	0.12	-0.13	0.07
Dispersion	NA	NA	0.52	-	5.59	0.44	1.30	0.08

of being hospitalized or killed of $\frac{\exp(-1.7)}{1+\exp(-1.7)} = 0.15$, regardless of whether an airbag was present. The average odds ratio comparing occupants in cars with airbags to occupants in cars without is given from our model by $\exp(-0.21) = 0.81$. Thus, occupants in cars with airbags are at decreased risk of being hospitalized or dying as a result of the crash compared to occupants of cars without airbags.

The random effects model assumes that each vehicle has its own probability for an occupant being killed or hospitalized which depends on an unobservable variable U. This randomness could be due to the age of the vehicle, vehicle design, or many other factors. However, given a specified value $U = u$, all occupants in the car have independent outcomes. Table 1.7 shows that for a vehicle with a value for $u = 0$, the baseline probability of an occupant being hospitalized or dying is $\frac{\exp(-3.05)}{1+\exp(-3.05)} = 0.05$. However, we also see a rather large estimate for the variance of U at 5.58. This again indicates that there is great heterogeneity between the outcomes of occupants in different vehicles; occupants of some vehicles have virtually no chance of being hospitalized or killed, while occupants of other vehicles are much more likely to be hospitalized or killed. If one were to assume a given value of $U = u$, then the odds of hospitalization or death for occupants of a vehicle equipped with an airbag are $\exp(-0.35) = 0.7$ times that of occupants of a vehicle without airbags and a similar value of u.

The QEM model again identifies the strong relationship between occupants of the

Table 1.8: Parameter Estimates from the Mice Malformation Data

	Logistic Regression		GEE		Random Effects		QEM	
	Estimate	SD	Estimate	SD	Estimate	SD	Estimate	SD
Inter.	-3.948	0.249	3.938	0.290	-4.542	0.461	-2.123	0.916
Dose	0.005	0.004	0.005	0.004	0.006	0.005	0.006	0.004
Disp.	NA	NA	0.022	-	1.34	0.866	0.162	0.082

same vehicle, $\hat{\delta} = 1.3$. If we assume that the front seat of a vehicle has two occupants and that one of them is not killed or hospitalized, then the odds that the second occupant is killed or hospitalized for a vehicle with airbags are $\exp(-0.13) = 0.88$ times the odds of a vehicle without airbags and two passengers with one uninjured. This odds ratio remains constant for fixed values of number of occupants and number of occupants killed or injured.

1.3.3 Example 3: Malformation in Mice Fetuses

Using several methods, Corcoran *et al.* analyzed the data from Table 1.3 and showed that the trend displayed by these data is marginally significant[9]. Estimates of the effect were calculated using GEE, mixed models, and the QEM and are presented in Table 1.8. Table 1.8 shows that the estimates of a dose effect are fairly close, regardless of the modeling method. The coefficients for logistic regression, GEE, random effects, and QEM are 0.005, 0.005, 0.006, 0.006, respectively. The similarity in estimates is due to little correlation between litter mates. The GEE estimates the correlation to be only 0.022. The random effects model estimates the variance of U, the unobserved factor that influences each female mouse's probability of producing an offspring with a malformation, to be 1.34. However, the standard error for this estimate is 0.866. Similarly, the QEM estimates half the log odds of malformation between litter mates to be 0.162, with a standard error of 0.082. Assuming no cor-

Table 1.9: Parameter Estimates from the Corneal Graft Data

	Logistic Regression		GEE		Random Effects		QEM	
Parameter	Estimate	SD	Estimate	SD	Estimate	SD	Estimate	SD
Intercept	-12.57	189.3	.	.	-3.16	1.79	-30.47	.
Age	12.57	189.3	.	.	3.30	1.92	30.47	.
Dispersion	NA	NA	.	.	0	.	-0.69	0.63

relation between litter mates, conventional logistic regression would be appropriate. The logistic regression results estimate the baseline probability of malformation to be $\frac{\exp(-3.95)}{1+\exp(-3.95)} = 0.02$. For each unit increase in dose, the odds of a malformation increase by $\exp(0.005) = 1.005$. The increase in odds of malformation for a fetus whose mother received a dose of 100 compared to a fetus whose mother received a dose of zero is $\exp(100 * 0.005) = 1.65$.

While all methods appear to find little evidence of significant intra-cluster correlation, this may be due to lack of power with so few observed malformations. The sparseness of these data may also render the asymptotic inferences suspect.

1.3.4 Example 4: Corneal Graphs

Corcoran *et al.*, applied an exact trend test to the data in Table 1.4 and showed a significant (p = 0.048) trend associated with age [9]. To estimate an age effect, we used logistic regression, GEE, a random effects model, and a QEM, with the results displayed in Table 1.9. Three of the methods produced estimates, but since all rejections occurred in the older age group, it is unclear whether these results can be trusted. The lack of estimates and standard errors seen in Table 1.9 is the result of trying to calculate an estimate, when the value lies on the boundary of the parameter space, essentially, $+\infty$. This example further illustrates the need for a small sample

method to estimate effect size especially one which handles unstable situations, such as when separation occurs.

CHAPTER 2

ESTIMATION APPROACHES FOR SMALL OR SPARSE SAMPLES

For all of the examples presented in Chapter 1, inference using GEE, random effects, and conditional models generally relies on large-sample theory. However, as demonstrated by the toxicology and corneal grafts examples in sections 1.3.3 and 1.3.4, for small or sparse samples asymptotic results may not be reliable. While there are several analytic options for small samples of independent data (such as exact conditional logistic regression), similar methods are lacking for estimating effects in the presence of clustering. Of GEE, random effects, and conditional models, only the conditional approach provides an analogue to conditional logistic regression. The exponential family nature of the quadratic exponential model lends itself to other developed algorithms for small-sample analysis.

Exact inference has a basis in conditioning on nuisance parameters and working with the conditional likelihood. Conditioning has been shown to be a successful approach for estimation when unconditional methods either fail or are biased. Cox and Hinkley have shown that when the parameter space is large in relation to the sample size, unconditional methods can be biased, and they suggest using conditional inference [11]. Breslow and Day provide another example in the estimation of a common odds ratio from matched pairs data [6]. If stratum-specific parameters are estimated, rather than being conditioned out of the model, the MLE estimate is shown to converge to the square of its actual value. For a detailed argument in favor of conditioning see Yates [54].

2.1 Exact Logistic Regression for Independent Binomials

Tools for exact conditional regression with independent binomials are well understood and widely available (for example see, [24], [35] and [38]). We briefly describe exact conditional logistic regression here to facilitate our discussion of exact methods for correlated data.

Exact conditional logistic regression is accomplished by conditioning on the sufficient statistics for the model regression parameters, in order to remove them from the likelihood. To illustrate, assume a sample of n independent binomial outcomes, where Y_i represents the number of successes in n_i Bernoulli trials, each with probability π_i of success. Let x_i be a covariate which is hypothesized to influence the outcome. Using the canonical link function, $\theta_i = \alpha + \beta x_i$ in Equation (1.4) we can express the probability of a success for the ith observation, y_i, as

$$P[Y_i = y_i] = \binom{n_i}{y_i} \frac{\exp[y_i(\alpha + \beta x_i)]}{1 + \exp(\alpha + \beta x_i)}. \tag{2.1}$$

From (2.1) we obtain the likelihood for the sample

$$P[Y_1 = y_1, \ldots, Y_n = y_n] = \prod_{i=1}^{n} \binom{n_i}{y_i} \frac{\exp[y_i(\alpha + \beta x_i)]}{1 + \exp(\alpha + \beta x_i)} \tag{2.2}$$

$$= \frac{\prod_{i=1}^{n} \binom{n_i}{y_i} \exp\left(\alpha \sum_{i=1}^{n} y_i + \beta \sum_{i=1}^{n} y_i x_i\right)}{\prod_{i=1}^{n} \left[1 + \exp(\alpha + \beta x_i)\right]}. \tag{2.3}$$

It is clear the sufficient statistics for α and β are, respectively, $t_1 = \sum_{i=1}^{n} y_i$ and $t_2 = \sum_{i=1}^{n} y_i x_i$.

If one is only interested in inference about the effect of x and not on the baseline probability determined by α, then we can eliminate α from the likelihood by conditioning on its corresponding sufficient statistic, $T_1 = \sum_{i=1}^{n} Y_i$. Let the reference set

of samples having $t_1 = \sum_{i=1}^{n} y_i$ be

$$S(t_1) = \{(y_1^*, \ldots, y_n^*) : \sum_{i=1}^{n} y_i^* = t_1\}, \tag{2.4}$$

Where $y_1^*, \ldots, y_n^*$ represent n binomial variables with number of trial fixed. By only considering tables in $S(t_1)$ the conditional likelihood becomes

$$P\Big(Y_1 = y_1, \ldots, Y_n = y_n \Big| \sum_{i=1}^{n} y_i = t_1\Big) = \frac{P[Y_1 = y_1, \ldots, Y_n = y_n]}{\sum_{S(t_1)} P[Y_1 = y_1^*, \ldots, Y_n = y_n^*]} \tag{2.5}$$

$$= \frac{\prod_{i=1}^{n} \binom{n_i}{y_i} \exp[\alpha t_1 + \beta \sum_{i=1}^{n} y_i x_i] / \prod_{i=1}^{n} [1 + \exp[\alpha + \beta x_i]]}{\sum_{S(t_1)} \prod_{i=1}^{n} \binom{n_i}{y_i^*} \exp[\alpha t_1 + \beta \sum_{i=1}^{n} y_i^* x_i] / \prod_{i=1}^{n} [1 + \exp[\alpha + \beta x_i]]} \tag{2.6}$$

$$= \frac{\prod_{i=1}^{n} \binom{n_i}{y_i} \exp[\beta \sum_{i=1}^{n} y_i x_i]}{\sum_{S(t_1)} \prod_{i=1}^{n} \binom{n_i}{y_i^*} \exp[\beta \sum_{i=1}^{n} y_i^* x_i]}. \tag{2.7}$$

Note that the conditional likelihood in Equation (2.7) is free of α. A conditional estimate and asymptotic variance for β can be calculated using conditional likelihood theory.

As we are interested in small or sparse samples, we consider methods for conducting inference about β using the exact distribution of its sufficient statistic. Defining $c(t_1, u) = \binom{n}{u}$ to be the number of samples in $S(t_1)$ for which $T_2 = u$, Equation (2.7) can be re-expressed as

$$P(T_2 = t_2 | T_1 = t_1) = \frac{c(t_1, t_2) \exp(t_2 \beta)}{\sum_u c(t_1, u) \exp(u\beta)}. \tag{2.8}$$

2.2 Conditional Exact Estimation

The exact conditional distribution can now be used to estimate and compute confidence intervals for β (for example see Hirji, Mehta, and Patel [24], Mehta, Patel,

and Senchaudhuri [37], and Mehta and Patel [38]). The exact conditional probability that $T_2 = t_2$ given t_1 can be expressed as

$$f_\beta(t_2|t_1) = \frac{c(t_1, t_2)\exp(t_2\beta)}{\sum_{t_{2,min}}^{t_{2,max}} c(t_1, u)\exp(u\beta)}. \tag{2.9}$$

where $t_{2,min}$ and $t_{2,max}$ are the minimum and maximum values respectively in the range of values assumed by T_2 conditioned on $T_1 = t_1$. The conditional maximum likelihood estimate (CMLE) for $\hat{\beta}$ is defined to be the value of β which maximizes Equation (2.9). If the observed value of t_2 is at either of the extremes of its distribution ($t_{2,min}$ or $t_{2,max}$) then it is not possible to maximize Equation (2.9) by choice of β. In this case, $\hat{\beta}$ is defined to be $-\infty$ when $t_2 = t_{2,min}$ or ∞ when $t_2 = t_{2,max}$.

When $\hat{\beta}$ is not defined, an alternative method for estimation is the median unbiased estimate (MUE) [38]. When $t_2 = t_{2,min}$ or $t_2 = t_{2,max}$ the MUE point estimate $\hat{\beta}$ satisfies the condition

$$f_{\hat{\beta}}(t_2|t_1) = 0.5 \tag{2.10}$$

When the CMLE exists, Mehta and Patel define the MUE to be the average of the end points of the 95% exact conditional confidence interval, defined below [38].

To obtain an exact confidence interval for β define the left and right tails of the distribution of T_2 given $T_1 = t_1$ to be

$$F_\beta(t_2) = \sum_{u=t_{2,min}}^{t} f_\beta(u|t_1), \tag{2.11}$$

$$G_\beta(t_2) = \sum_{u=t_2}^{t_{2,max}} f_\beta(u|t_1). \tag{2.12}$$

Let β_- and β_+ be the lower and upper bounds of a two-sided (1 - α)% confidence interval for β. Then β_- is defined to be such that

$$G_{\beta_-}(t_2) = \alpha/2 \text{ if } t_{2,min} < t_2 \leq t_{2,max}, \tag{2.13}$$

$$\beta_- = -\infty \text{ if } t_2 = t_{2,min}. \tag{2.14}$$

Similarly β_+ is

$$F_{\beta_+}(t_2) = \alpha/2 \text{ if } t_{2,min} \leq t_2 < t_{2,max}, \tag{2.15}$$

$$\beta_+ = \infty \text{ if } t_2 = t_{2,max}. \tag{2.16}$$

Under this framework it is also possible to calculate one-sided confidence intervals for β. Define $F_0(t_2)$ and $G_0(t_2)$ to be the left and right tails of the distribution of T_2 given $T_1 = t_1$ assuming $\beta = 0$. Then a one-sided confidence bound can be calculated depending on the smallest of $F_0(t_2)$ and $G_0(t_2)$.

If $F_0(t_2) \leq G_0(t_2)$ then a one-sided confidence interval has the form $(-\infty, \beta_+)$ where the upper confidence bound β_+ satisfies

$$F_{\beta_+} = \alpha.$$

If $F_0(t_2) > G_0(t_2)$ then a one-sided confidence interval has the form (β_-, ∞) where the lower confidence bound β_- satisfies

$$G_{\beta_-} = \alpha.$$

While several methods for calculating p-values for exact logistic regression exist, we describe the method which preserves consistency between p-values and confidence

intervals. A one-sided p-value, p_1, is defined to be

$$p_1 = \min\{F_0(t_2), G_0(t_2)\}.$$

The two-sided p-value, p_2 is defined to be double the one-sided p-value, i.e.

$$p_2 = 2p_1.$$

2.3 An Exact Framework for Clustered Binomials

We can extend exact conditional regression to correlated binomials by applying these principles to the quadratic exponential model defined in (1.12). Throughout this section we will assume that we have a sample of N independent clusters. Each cluster has n_i observations, where $i = 1 \ldots N$. Each observation y_{ij} is a binary variable with an associated fixed factor x_i, where y_{ij} is a specific realization of Y_{ij}. Note we assume x_i is a cluster level covariate that is hypothesized to influence the probability that $Y_{ij} = 1$. Let $z_i = \sum_{i=1}^{N} \sum_{j=1}^{n_i} y_{ij}$ represent the number of successes in cluster i. Using quadratic exponential model in (1.12) we obtain

(2.17)

$$P[Z_i = z_i|x_i] = \binom{n_i}{z_i} \exp\{(\alpha + \beta x_i)z_i - \delta z_i(n_i - z_i) + A_i(\alpha, \beta, \delta)\}, \quad z_i = 0, \ldots, n_i,$$

where α represents a baseline probability of a success given that there is no inter-cluster correlation, β represents the increase in the log odds of a success for a given cluster size and number of successes per unit increase in x, and δ represents the amount of inter-cluster correlation.

2.3.1 Conditional Likelihood of the QEM

Since clusters are independent, the joint distribution of the sample, $\mathbf{Z} = (Z_1, \ldots, Z_N)'$ given $\mathbf{x} = (x_1, \ldots, x_N)'$ can be expressed as

$$P[\mathbf{Z} = \mathbf{z}|\mathbf{x}] = \prod_{i=1}^{N} \binom{n_i}{z_i} \exp\{(\alpha + \beta x_i)z_i - \delta z_i(n_i - z_i) + A_i(\alpha, \beta, \delta)\} \tag{2.18}$$

$$= \left[\prod_{i=1}^{N} \binom{n_i}{z_i}\right] \exp\{\alpha s_1 + \beta t - \delta s_2 + \sum_{i=1}^{N} A_i(\alpha, \beta, \delta)\}, \tag{2.19}$$

where $s_1 = \sum_i z_i$, $t = \sum_i x_i z_i$, and $s_2 = \sum_i z_i(n_i - z_i)$. Because the density is from the exponential family, s_1, s_2, and t are sufficient statistics for α, δ, and β, respectively.

We can now condition on the sufficient statistics of the nuisance parameters, α and δ, and proceed to calculate a conditional estimate for β. Define the conditional reference set $\Gamma(s_1, s_2)$ as the set of all samples, $\mathbf{z}^*$, with sufficient statistics for α and δ equal to the observed values of s_1 and s_2,

$$\Gamma(s_1, s_2) = \left\{\mathbf{z}^* : \sum_{i=1}^{N} z_i^* = s_1, \sum_{i=1}^{N} z_i^*(n_i - z_i^*) = s_2\right\}, \tag{2.20}$$

where z^* is any table of the form $z^* = z_1^*, \ldots, z_N^*$ with fixed cluster sizes. Then, by conditioning on $S_1 = s_1$ and $S_2 = s_2$, and letting $u = \sum_{i=1}^{N} x_i z_i^*$, the conditional density of t becomes,

(2.21)

$$P[\mathbf{Z}=\mathbf{z}|x,s_1,s_2] = \frac{\left[\prod_{i=1}^{N}\binom{n_i}{z_i}\right]\exp\{\alpha s_1+\beta t-\delta s_2+\sum_{i=1}^{N}A_i(\alpha,\beta,\delta)\}}{\sum_{z^*\in\Gamma(s_1,s_2)}\left[\prod_{i=1}^{N}\binom{n_i}{z_i^*}\right]\exp\{\alpha s_1+\beta u-\delta s_2+\sum_{i=1}^{N}A_i(\alpha,\beta,\delta)\}}$$

(2.22)
$$= \frac{\prod_{i=1}^{N}\binom{n_i}{z_i}\exp(\beta t)}{\sum_{z^*\in\Gamma(s_1,s_2)}\prod_{i=1}^{N}\binom{n_i}{z_i^*}\exp(\beta u)}.$$

Note that the exact conditional model has been used primarily for hypothesis testing with respect to a single ordinal covariate x_i. We extend this approach here for effect size estimation, to facilitate general exact logistic regression in the presence of several covariates.

2.3.2 Conditional Estimation

Conditional estimation of β can be conducted using methods analogous to Section 2.2. Define f_β to be the exact conditional probability that $T=t$ given $S_1=s_1$ and $S_2=s_2$, i.e.

(2.23)
$$f_\beta(t|s_1,s_2) = \frac{\prod_{i=1}^{N}\binom{n_i}{z_i}\exp(\beta t)}{\sum_{z^*\in\Gamma(s_1,s_2)}\prod_{i=1}^{N}\binom{n_i}{z_i^*}\exp(\beta u)}.$$

Furthermore, define t_{min} and t_{max} as the minimum and maximum values respectively in the range of values assumed by T conditioned on s_1 and s_2. The conditional maximum likelihood estimate (CMLE) $\hat{\beta}$ is defined to be the value of β which maximizes Equation (2.23). If the observed value of t is at either of the extremes of its distribution (t_{min} or t_{max}) then it is not possible to maximize Equation (2.23) by choice of β. In these cases, $\hat{\beta}$ is defined to be $-\infty$ when $t=t_{min}$ or ∞ when $t=t_{max}$.

When $\hat{\beta}$ is not defined, a median unbiased estimate (MUE) can again be estimated [38]. When $t=t_{min}$ or $t=t_{max}$ the MUE point estimate $\hat{\beta}$ satisfies the

condition

$$f_{\hat{\beta}}(t|s_1, s_2) = 0.5 \tag{2.24}$$

When the CMLE exists, Mehta and Patel define the MUE to be the average of the end points of a 95% exact conditional confidence interval, defined below [38].

To obtain an exact confidence interval for β define the left and right tails of the distribution of T given $S_1 = s_1$ and $S_2 = s_2$ to be

$$F_\beta(t) = \sum_{u=t_{min}}^{t} f_\beta(u|s_1, s_2), \tag{2.25}$$

$$G_\beta(t) = \sum_{u=t}^{t_{max}} f_\beta(u|s_1, s_2). \tag{2.26}$$

Let β_- and β_+ be the lower and upper bounds of a two-sided (1 - α)% confidence interval for β. Then β_- is defined to be such that

$$G_{\beta_-}(t) = \alpha/2 \text{ if } t_{min} < t \leq t_{max}, \tag{2.27}$$

$$\beta_- = -\infty \text{ if } t = t_{min}. \tag{2.28}$$

Similarly β_+ is

$$F_{\beta_+}(t) = \alpha/2 \text{ if } t_{min} \leq t < t_{max}, \tag{2.29}$$

$$\beta_+ = \infty \text{ if } t = t_{max}. \tag{2.30}$$

Under this framework it is possible to construct one-sided confidence intervals for β. Define $F_0(t)$ and $G_0(t)$ to be the left and right tails of the distribution of T

given $S_1 = s_1$ and $S_2 = s_2$ assuming $\beta = 0$. A one-sided confidence bound can then be calculated depending on the smallest of $F_0(t)$ and $G_0(t)$.

If $F_0(t) \leq G_0(t)$ then a one-sided confidence interval has the form $(-\infty, \beta_+)$ where the upper confidence bound β_+ satisfies

$$F_{\beta_+} = \alpha.$$

If $F_0(t) > G_0(t)$ then a one-sided confidence interval has the form (β_-, ∞) where the lower confidence bound β_- satisfies

$$G_{\beta_-} = \alpha.$$

While several methods for calculating exact p-values exist, we describe the method which preserves consistency between p-values and confidence intervals. A one-sided p-value, p_1, is defined to be

$$p_1 = \min\{F_0(t), G_0(t)\}.$$

The two-sided p-value, p_2 is defined to be double the one-sided p-value, i.e.

$$p_2 = 2p_1.$$

CHAPTER 3

COMPUTATIONAL METHODS FOR CONDITIONAL ESTIMATION

One of the potential difficulties with exact conditional inference is the need to identify all tables in a reference set, such as that defined by (2.20). In this case, as the number of clusters or even the number of observations per cluster grows, the number of tables in the reference set $\Gamma(s_1, s_2)$ generally increases exponentially, and the set can not be stored explicitly. For instance, there are over 1.04 X 10^{55} tables in the reference set for the toxicology data of Example 1.3.3. While there are several competing computational methods for efficient exact computation, we implement two of the most widely considered and implemented: the network algorithm and Markov chain Monte Carlo (MCMC). Both approaches seek to minimize computational burdens by significantly reducing the memory required to identify a given reference set.

3.1 The Network Algorithm

The network approach *implicitly* represent all tables in $\Gamma(s_1, s_2)$ through the use of a graphical network. The following provides an outline of the necessary details for constructing the required network, for more detailed discussions see Corcoran [8] and Mehta, Patel, and Senchaudhuri [36].

We will use the data in Table 3.1 to illustrate the use of the network algorithm in determining and processing the exact conditional distribution. There are five clusters and three levels of the explanatory variable x. Per the previous discussion, we wish to determine the permutation distribution of $t = \sum_{i=1}^{5} x_i z_i$, which consists of only those tables with sufficient statistics $\sum z_i = 4$ and $\sum z_i(n_i - z_i) = 2$.

Table 3.1: Example Sample of Clustered Data

Cluster	1	2	3	4	5	
Dose	0	1	1	2	2	Total
z_i	0	1	0	1	2	4
$n_i - z_i$	3	1	2	1	0	6
n_i	3	2	2	2	2	10
$x_i z_i$	0	1	0	2	4	7
$z_i(n_i - z_i)$	0	1	0	1	0	2

Rather than trying to directly construct $\Gamma(s_1, s_2)$, we will first consider the superset of tables, $\Gamma(s_1)$, that meet the requirement of $\sum z_i = s_1$. The network representation for $\Gamma(s_1)$ is well-defined and has been applied in a variety of settings [8, 36, 37, 38]. Our strategy will therefore be to first implicitly identify the larger set $\Gamma(s_1)$. Next we will identify $\Gamma(s_1, s_2)$ by eliminating all elements of $\Gamma(s_1)$ not satisfying the quadratic requirement imposed by s_2.

Begin by representing the reference sets $\Gamma(s_1)$ and $\Gamma(s_1, s_2)$ individually as directed networks of nodes and arcs. Each network is divided into $N+1$ stages, indexed over $0, \ldots, N$, where N is the number of clusters. At each stage k there is a set of nodes. For $\Gamma(s_1)$, each node is indexed by two elements, denoted by (k, s_{1k}). The first element, k, represents the kth cluster. The second component, $s_{1k} = \sum_{i=1}^{k} z_k$, is one possible value of the partial sum of successes from the first k clusters. For the network representing $\Gamma(s_1, s_2)$, nodes are indexed by three elements, k, s_{1k}, s_{2k}. The elements k and s_{1k} are defined as in $\Gamma(s_1)$, while the third element, $s_{2k} = \sum_{i=1}^{k} z_i(n_i - z_i)$, represents one possible value of the partial sum of response/nonresponse products. For each network, there is a single initial node and a single terminal node. The initial nodes are (0,0) and (0,0,0), and the terminal nodes are (N, s_1) and (N, s_1, s_2) for $\Gamma(s_1)$ and $\Gamma(s_1, s_2)$, respectively. For either of the two networks, each path which begins at the initial node and ends at the terminal node corresponds to exactly one

table within the corresponding reference set, with observed successes obtained as $\mathbf{z} = [(s_{11} - 0), (s_{12} - s_{11}), \ldots, (s_{1N} - s_{1,N-1})]$. For any such path in either $\Gamma(s_1)$ or $\Gamma(s_1, s_2)$, $\sum_{i=1}^{N}(s_{1i} - s_{1,i-1}) = s_1$, and for a path in $\Gamma(s_1, s_2)$ it is also true that $\sum_{i=1}^{N}(s_{i1} - s_{1,i-1})(n_i - s_{i1} + s_{1,i-1}) = s_2$.

As mentioned previously, a network defined for a set of the form $\Gamma(s_1)$ can be constructed in closed form. Define $\mathcal{R}(k-1, s_{1,k-1})$ as the set of successor nodes to (k, s_{1k}). For each node $(k+1, u) \in \mathcal{R}(k, s_{1k})$ define the length r_{2k} of the connecting arc as $r_{2k} = (s_{1,k+1} - s_{1k})(n_{k+1} - s_{1,k+1} + s_{1k})$. Let $SP_2(k, s_{1k})$ and $LP_2(k, s_{1k})$, respectively, represent the shortest and longest path lengths $\sum_{l=1}^{k} r_{2l}$ over all partial paths that originate at node (0,0) and terminate at $(k, s_{1,k})$. The longest and shortest paths can be calculated recursively. For example, note that the set $\mathcal{P}$ of predecessor nodes to (k, s_{1k}) is given by $\mathcal{P}(k, s_{1k}) = \{(k, u) : (k, s_{1,k}) \in \mathcal{R}(k-1, u)\}$, for $k = 1, \ldots, N$. Therefore,

$$SP_2 = \min_{\{(k-1,u) \in \mathcal{P}(k, s_{1k})\}} \{SP_2(k-1, u) + r_{2k}\}.$$

LP_2 can be calculated similarly.

To generate $\Gamma(s_1, s_2)$ from $\Gamma(s_1)$, begin by creating the terminal node (N, s_1, s_2). Next, process the network backwards, stage by stage, beginning at stage N and ending at stage 0. At the kth stage

1. Choose node (k, s_{1k}, s_{2k}).

2. For each node $(k-1, u) \in \mathcal{P}(k, s_{1k})$:

   (a) Create the triple $(k-1, u, v)$, where $v = s_{2k} - r_{2k} = s_{2k} - (s_{1k} - u)(n_k - s_{1k} + u)$.

   (b) If $SP_2(k-1, u) > v$ or $LP_2(k-1, u) < v$, then $(k-1, u, v)$ cannot be a member of $\Gamma(s_1, s_2)$, and is discarded.

(c) If the triple $(k-1, u, v)$ passes the longest and shortest path tests above, then the node is possible, and is stored.

3. Repeat steps 1 and 2 until all nodes (k, s_{1k}, s_{2k}) for stage k have been exhausted.

The goal of the network algorithm is to efficiently calculate the exact conditional distribution of T for an observed table, conditional on the values of s_1 and s_2, i.e. Equation (2.23). To do so, the algorithm must recursively gather information regarding the density of T while finding those triples $(k-1, u, v)$ which belong to $\Gamma(s_1, s_2)$. First having identified node $(k-1, u, v)$ as a potential predecessor of (k, s_{1k}, s_{2k}), we compute the rank length $r_{1k} = x_k(s_{1k} - u)$, and compute the unnormalized probability length c_{0k} of the arc connecting $(k-1, u, v)$ to (k, s_{1k}, s_{2k}) as $c_{0k} = \binom{n_k}{s_{1k}-u}$. Note that for any path in $\Gamma(s_1, s_2)$ which begins at the initial node $(0, 0, 0)$ and ends at the terminal node (N, s_1, s_2), the value of t for that table can be computed as $\sum_{l=1}^{N} r_{1l}$, and the unnormalized probability under $H_0 : \beta = 0$ of having observed the corresponding table is $\prod_{l=1}^{N} c_{0k}$. Therefore we can define $SP_1(k-1, u, v)$ and $LP_1(k-1, u, v)$, respectively, as the shortest and longest pathes $\sum_{l=k}^{N} r_{1l}$ over all paths beginning at $(k-1, u, v)$ and ending at the terminal node (N, s_1, s_2). Likewise, let $TP(k-1, u, v)$ represent the sum of all unnormalized probability paths $\prod_{l=k}^{N} c_{0l}$ over all such partial paths. Therefore, $SP_1(N, s_1, s_2) = LP_1(N, s_1, s_2) = 0, TP(N, s_1, s_2) = 1$, and $TP(0, 0, 0)$ is the normalizing constant in the denominator of (2.23).

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

- If $(k-1, u, v)$ already exists (i.e., has already been identified as the predecessor of another node at the kth stage), then

  1. $SP_1(k-1, u, v) = \min\{SP_1(k-1, u, v), r_{1k} + SP_1(k, s_{1k}, s_{2k})\}$.

  2. $LP_1(k-1, u, v) = \max\{LP_1(k-1, u, v), r_{1k} + LP_1(k, s_{1k}, s_{2k})\}$.

3. $TP(k-1,u,v) = TP(k-1,u,v) + c_{0k}TP(k,s_{1k},s_{2k})$.

- if $(k-1,u,v)$ has not yet been identified, then

1. $SP_1(k-1,u,v) = r_{1k} + SP_1(k,s_{1k},s_{2k})$.
2. $LP_1(k-1,u,v) = r_{1k} + LP_1(k,s_{1k},s_{2k})$.
3. $TP(k-1,u,v_) = c_{0k}TP(k,s_{1k},s_{2k})$.

Now the network is traversed forward. Beginning at the initial node $(0,0,0)$ we process the network forward stage by stage. We will carry forward a set of records Υ_k at each stage for $k = 0,\ldots,N-1$. Each record of $v \in \Upsilon_k$ is of the form $v = (k,s_{1k},s_{2k},t_k,h_k)$, where k, s_{1k}, and s_{2k} are as defined previously, $t_k = \sum_k r_{1k}$ is a possible partial rank length of a path terminating at node (k,s_{1k},s_{2k}), and h_k represents the unnormalized sum of the probabilities of all partial paths with rank length r_{1k} that terminate at (k,s_{1k},s_{2k}).

To produce the needed distribution in (2.23) we first begin with a single record in Υ_0, $(0,0,0,0,1)$ associated with the initial node $(0,0,0)$, and proceed forward stage by stage, for $k \in \{0,\ldots,N-1\}$. At the kth stage:

1. Choose a record $v = (k,s_{1k},s_{2k},t_k,h_k) \in \Upsilon_k$.
2. For each $u = (k+1,s_{1,k+1},s_{2,k+1}) \in \mathcal{R}(k,s_{1k},s_{sk})$, transfer to Υ_{k+1} as follows:
   (a) If there already exists a $v' = (k+1,s_{1,k+1},s_{2,k+1},t_{k+1},h_{k+1}) \in \Upsilon_{k+1}$ such that $t_{k+1} = t_k + r_{1,k+1}$, then merge v and v' by letting

$$v' = (k+1,s_{1,k+1},s_{2,k+1},t_{k+1},h'_{k+1}),$$

where $h'_{k+1} = h_{k+1} + h_k\binom{n_{k+1}}{s_{1,k+1}-s_{1,k}}$

(b) Else create a new record $v' \in \Upsilon_{k+1}$ such that

$$v' = (k+1, s_{1,k+1}, s_{2,k+1}, t_k + r_{1,k+1}, h_k \binom{n_{k+1}}{s_{1,k+1} - s_{1k}}).$$

3. Continue until all $v \in \Upsilon_k$ are exhausted.

At stage N we now have a set of records Υ_N which contain the needed information about the distribution of T conditional on s_1 and s_2. Each record $v = (N, s_1, s_2, t_N, h_N) \in \Upsilon_N$ contains information about one element of f_β in Equation (2.23). For a given v

$$P[T = v] = \frac{h_k}{TP(0,0,0)}.$$

We can now use the methodology of Section 2.3.2 for point and interval estimation. If only a p-value is desired, Corcoran *et al.* have shown how one can take advantage of the inductive nature of the network algorithm and shortest path - longest path methodology to realize even more computational and memory savings [8].

Figures 3.1 and 3.2 display the network representations of $\Gamma(s_1)$ and $\Gamma(s_1, s_2)$ for the data presented in Table 3.1. The dashed line in Figure 3.2 corresponds to the observed table, which is the path

$$(0,0,0) \to (1,0,0) \to (2,1,1) \to (3,1,1) \to (4,2,2) \to (5,4,2).$$

The observed table can now be obtained from the network by subtracting the second element of each node from the second element of its predecessor node, i.e.

$$z_{obs} = (0-0, 1-0, 1-1, 2-1, 4-2) = (0,1,0,1,2).$$

Figure 3.2 shows that there are 16 distinct tables in the reference set: (0,0,1,1,2),

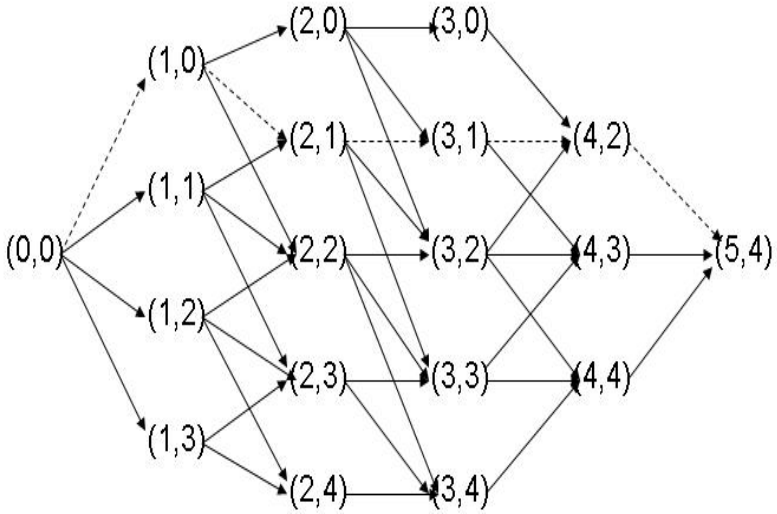

Fig. 3.1: Network Representation of $\Gamma(s_1)$ for the Data in Table 3.1.

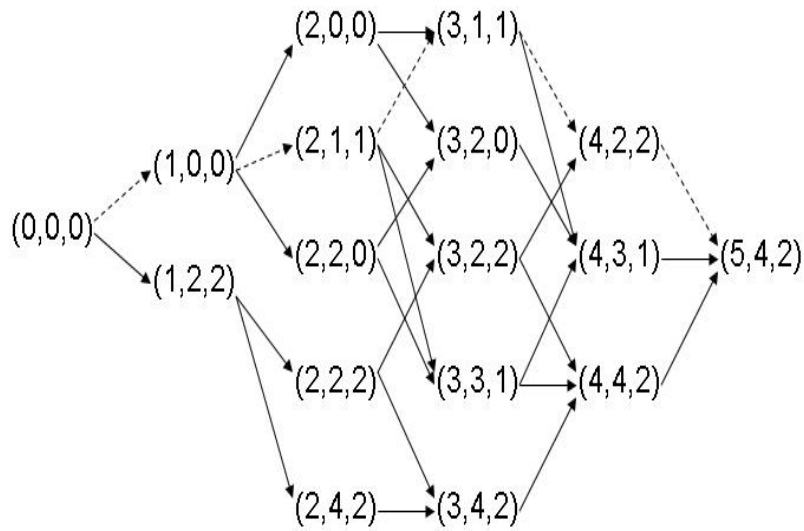

Fig. 3.2: Network Representation of $\Gamma(s_1, s_2)$ for the Data in Table 3.1.

Table 3.2: Example Sample of Clustered Data

$T = \sum z_i x_i$	$\mathrm{P}[T = t]$
2	.1
4	.1
5	.267
6	.267
7	.267

(0,0,1,2,1), (0,0,2,1,1), (0,1,0,1,2), (0,1,0,2,1), (0,1,1,0,2), (0,1,1,2,0), (0,1,2,0,1), (0,1,2,1,0), (0,2,1,1,0), (0,2,1,0,1), (0,2,0,1,1), (2,0,0,0,2), (2,0,2,0,0), (2,0,0,2,0), (2,2,0,0,0). By processing Figure 3.2 as outlined above it is possible to generate the distribution of $t = \sum_{i=1}^{5} z_i x_i$, which is displayed in Table 3.2.

3.2 Markov Chain Monte Carlo

While the network algorithm can process small to moderate-sized examples (such as the toxicology example of Section 1.1.3), larger examples may exhaust available storage space in memory. For this reason, it is necessary to develop a conditional analysis method that can handle very large tables. Markov chain Monte Carlo (MCMC) procedures may be one such method [13, 14, 15, 50].

3.2.1 Introduction

MCMC methods generate observations from a distribution under consideration when direct methods are not available. These observations can then be used to test hypotheses. The two most popular forms of MCMC are Metropolis-Hastings [22, 40] and the Gibbs sampler [19].

Researchers have used MCMC to generate data from exponential mixed models [45] and for approximate conditional inference on exponential families [29]. Using a Gibbs sampler method, Forster, McDonald, and Smith applied MCMC methodology

to logistic regression [18]. Their method generated individual conditional distributions based on the sufficient statistics outlined in Section 2. The attractiveness of this approach is that the conditional distributions are generated one at a time rather than generating observations from the joint density. However, Mehta, Patel, and Senchaudhuri present an example where this method never successfully samples a table different than the starting table [37]. Thus, rather than returning the exact p-value of 0.0426, Forster's, MCMC method returns a p-value of 1.0.

For this reason, research has been ongoing in deriving a Metropolis-Hastings algorithm for conducting MCMC exact logistic regression. Starting with an initial observation, the Metropolis-Hastings algorithm proceeds by generating a candidate from the proposed distribution which is accepted with a probability that depends on both the current and candidate points. In our situation, we would like to start with an observed table and generate candidate tables having sufficient statistics, $\sum z$ and $\sum z * (n - z)$, matching those which have been observed. Once a candidate table is generated, we will calculate a, where

$$a = \min\left\{\frac{\text{likelihood[candidate table]}}{\text{likelihood[current table]}}, 1\right\}. \tag{3.1}$$

Once a is calculated, one then generates u from a $U(0, 1)$ distribution. If $u \leq a$ the candidate table is accepted and a new candidate table is generated. Otherwise, the current table is retained and a new candidate is generated.

To illustrate, consider again Table 3.1. As determined previously, there are sixteen tables in the corresponding reference set, which are enumerated in Table 3.3. A Monte Carlo sampling approach would repeatedly sample tables from Table 3.3 with the correct hypergeometric probability shown in column 8. For illustration purposes, we will assume all possible tables are known and can be enumerated. However, this

Table 3.3: Probability of Obtaining Each Table in the Reference Set from Table 3.1

Table #	z_1	z_2	z_3	z_4	z_5	Ways to Generate Table	Probability of Table
1	0	0	1	1	2	4	0.0667
2	0	0	1	2	1	4	0.0667
3	0	0	2	1	1	4	0.0667
4	0	1	0	1	2	4	0.0667
5	0	1	0	2	1	4	0.0667
6	0	1	1	0	2	4	0.0667
7	0	1	1	2	0	4	0.0667
8	0	1	2	0	1	4	0.0667
9	0	1	2	1	0	4	0.0667
10	0	2	1	1	0	4	0.0667
11	0	2	1	0	1	4	0.0667
12	0	2	0	1	1	4	0.0667
13	2	0	2	0	0	3	0.05
14	2	0	0	0	2	3	0.05
15	2	0	0	2	0	3	0.05
16	2	2	0	0	0	3	0.05
Total						60	1

is rarely the case and we will later show how to incorporate Markov Chains into the sampling procedure to overcome this problem.

A Monte Carlo sample from the reference set would start with the observed data, shown in row 4 and denoted by $T^{(1)}$. We would next select one of the remaining tables with equal probability. For example, suppose the table contained in row 16 is chosen. Next, we calculate a as $\min\left(\frac{.05}{.0667}, 1\right) = 0.75$. Next, we generate u from a $U(0,1)$ density. If $u \leq 0.75$, The table represented in row 16 is selected as $T^{(2)}$. Otherwise, $T^{(2)}$ is $T^{(1)}$, the observed table. Following this process we generated a Monte Carlo sample of size 100,000 tables. The Monte Carlo estimates of the actual probabilities are shown in Table 3.4.

Table 3.4: MCMC Relative Frequency Obtained for Each of the Tables in the Reference Set from Table 3.1

Table #	z_1	z_2	z_3	z_4	z_5	Monte Carlo Estimate	Actual Probability
1	0	0	1	1	2	0.0680	0.0667
2	0	0	1	2	1	0.0690	0.0667
3	0	0	2	1	1	0.0690	0.0667
4	0	1	0	1	2	0.0650	0.0667
5	0	1	0	2	1	0.0730	0.0667
6	0	1	1	0	2	0.0580	0.0667
7	0	1	1	2	0	0.0700	0.0667
8	0	1	2	0	1	0.0640	0.0667
9	0	1	2	1	0	0.0665	0.0667
10	0	2	1	1	0	0.0670	0.0667
11	0	2	1	0	1	0.0725	0.0667
12	0	2	0	1	1	0.0705	0.0667
13	2	0	2	0	0	0.0490	0.05
14	2	0	0	0	2	0.0465	0.05
15	2	0	0	2	0	0.0490	0.05
16	2	2	0	0	0	0.0430	0.05

3.2.2 Random Walk on Contingency Tables with Fixed Row and Column Sums

The main limitation with this example is that one does not usually have the reference set from which to directly generate candidate tables. For this reason, Diaconis and Sturmfels incorporated Markov Chains to create a Markov chain Monte Carlo (MCMC) method for estimating the p-value from Fisher's exact test on I x J tables [15]. Fisher's exact test requires that row and column totals in a two-way table be fixed. In order to generate a Monte Carlo Sample, we therefore need to randomly generate tables from the set of all tables having row and column sums the same as those observed. The Diaconis-Sturmfels Markov basis involves picking two rows, r_1 and r_2, and two columns, c_1 and c_2, at random. Next perform one of two moves, each with fifty percent probability. Move 1 adds one to the cell at the intersection of row 1 and column 1, c_{11}, and subtracts 1 from c_{12}, this keeps the row sum constant. To keep the column sums constant, 1 is subtracted from c_{21} and 1 is added to c_{22}. The

second move is the same as the first, but with all the signs reversed. If the resulting table contains any negative cell counts it is rejected, otherwise the chain moves to the new table with probability proportional to the hypergeometric probability of table. Using this method, it is highly likely that two consecutively generated tables will be dependent. Therefore, Diaconis and Sturmfels suggest ignoring the first 50,000 generated tables. This is referred to as the burn-in period. Once the burn-in period has completed, a staggered sampling technique is recommended where only the results of every 50th table are saved.

In our clustered binomial setting, we are interested in two-way tables of the form illustrated in Table 3.3. In this case, the fixed cluster sizes and the conditioning constraints require tables in the sample space to have the same number of total success, so that - as in the Diaconis - Sturmfels setting - we are concerned with tables having fixed row and column sums. We are only imposing an additional requirement, namely, $\sum z_i(n_i - z_i)$ is also fixed. Therefore, we propose combining the Diaconis and Sturmfels procedure with rejection sampling. This hybrid approach will identify tables in $\Gamma(s_1)$, and only those meeting the additional constraint, i.e. those in $\Gamma(s_1, s_2)$ will be retained.

3.2.3 Conditional Likelihood and a

In order to perform the MCMC as described above, we need to determine the appropriate method for calculating a, which controls the transitions between tables. Takken shows how a can be constructed from the conditional likelihood in the unclustered binary setting which we have adapted for the QEM [50]. To control the transition between tables we make use of the conditional likelihood of T given s_1 and

s_2 calculated in Section 2.3.1, namely,

$$P[\mathbf{Z} = \mathbf{z}|x, s_1, s_2] = \frac{\prod_{i=1}^{N} \binom{n_i}{z_i} \exp(\beta t)}{\sum_{z^* \in \Gamma(s_1,s_2)} \prod_{i=1}^{N} \binom{n_i}{z_i^*} \exp(\beta u)}. \tag{3.2}$$

To calculate a given two tables in the reference set, $\Gamma(s_1, s_2)$, we need to take the ratio of the conditional likelihoods for the two tables. Let T^i and T^{i+1} be tables in $\Gamma(s_1, s_2)$ and let T^i be the current state of our Markov Chain and T^{i+1} be a proposed table to which the chain might move. Similarly, we define z_j^i to be the number of successes in cluster j for table T^i and z_j^{i+1} as the number of successes in cluster j for table T^{i+1}. Finally, let t^i be the value of the sufficient statistic, $t^i = \sum_{j=1}^{N} x z_j^i$, for the table T^i and t^{i+1} be the value of $t^{i+1} = \sum_{j=1}^{N} x z_j^{i+1}$ for table T^{i+1}. We then calculate a as the ratio of Equation (3.2) between the two tables.

$$a = \frac{P[T^{i+1}]}{P[T^i]} \tag{3.3}$$

$$= \frac{\dfrac{\prod_{j=1}^{N} \binom{n_j}{z_j^{i+1}} \exp(\beta t^{i+1})}{\sum_{z^* \in \Gamma(s_1,s_2)} \prod_{j=1}^{N} \binom{n_j}{z_j^*} \exp(\beta u)}}{\dfrac{\prod_{j=1}^{N} \binom{n_j}{z_j^i} \exp(\beta t^i)}{\sum_{z^* \in \Gamma(s_1,s_2)} \prod_{j=1}^{N} \binom{n_j}{z_j^*} \exp(\beta u)}} \tag{3.4}$$

$$= \frac{\prod_{j=1}^{N} \binom{n_j}{z_j^{i+1}}}{\prod_{j=1}^{N} \binom{n_j}{z^i{}_j}} \exp\left(\beta(t^{i+1} - t^i)\right) \tag{3.5}$$

To test the hypothesis that $H_o : \beta = 0$, Equation (3.5) reduces to $a = \frac{\prod_{j=1}^{N} \binom{n_j}{z_j^{i+1}}}{\prod_{j=1}^{N} \binom{n_j}{z_j^i}}$.

To see how our earlier example fits into this framework let's return to Table 3.1, setting $\beta = 0$. Again assume that the chain currently stands at Row 4 and we have a proposal of Row 16. Using Equation (3.5) we get a value of

$$a = \frac{3 * 1 * 1 * 1 * 1}{1 * 2 * 1 * 2 * 1} = 0.75,$$

which is the needed value.

3.2.4 Random Walk on the Quadratic Exponential Model

We will employ a combination of the Diaconis and Sturmfels method and rejection sampling to our situation. Rejection sampling is necessary because the theorems used for generating the +1,-1 moves only ensure that linear constraints are held fixed. These theorems do not apply to quadratic constraints, such as $s_2 = \sum z_i(n_i - z_i)$. However, these theorems do ensure that our other constraints, $s_1 = \sum z_i$ and cluster sizes remain constant.

To implement our hybrid algorithm, we will start with the observed table. We will then use one of four moves to create a proposed table. The first move will select two clusters at random, c_1 and c_2 with number of successes z_1 and z_2. z_1 will become z_1+1 and z_2 will become z_2-1. Move two will be the same as move one but the signs will be reversed, so that z_1 becomes z_1-1 and z_2 becomes z_2+1. The third move will select three clusters, c_1, c_2, c_3, with number of successes z_1, z_2, z_3. To keep the number of successes constant, we will do the following $z_1 \to z_1+1, z_2 \to z_2-2, z_3 \to z_3+1$. Finally, the fourth move will be the same as move three but with the signs reversed, i.e. $z_1 \to z_1-1, z_2 \to z_2+2, z_3 \to z_3-1$. Even though these last two moves are the sums of two of the smaller moves recommended by Diaconis and Sturmfels, Jones and O'Neil recommend incorporating them to help mix the distribution [26].

Given the observed table is T^1, test statistic $t^1 = \sum_{j=1}^{N} xz_j^1$ and observed constraints of $s_1 = \sum z$ and $s_2 = \sum z(n-z)$, as recommended by Diaconis and Sturmfels [15] we will run our algorithm for 50,000 burn-in steps, followed by the generation of 100,000 steps, retaining every 50th table. Given the Markov chain currently stands at Table T^i, at the $(i+1)$st step our algorithm proceeds as follows:

1. Given T^i generate a proposal table T^i_* using one of the four moves described above.

2. If any cluster in T^i_* has negative successes or failures discard T^i_* and return to Step 1.

3. Otherwise, calculate $s_{2_*} = \sum_{j=1}^{n} z^i_{*_j}(n_j - z^i_{*_j})$.

   (a) If $s_{2_*} \neq s_2$, set $T^i = T^i_*$ and return to Step 1.

   (b) Otherwise calculate a using Equation (3.5).

      i. If $a \geq 1$,

         A. $T^{i+1} = T^i_*$

         B. Store $t^{i+1} = \sum_{j=1}^{N} xz^{i+1}_j$.

         C. Return to Step 1.

      ii. Otherwise generate u from a U(0,1) distribution.

         A. If $u \leq a$ then $T^{i+1} = T^i_*$.

         B. Otherwise $T^{i+1} = T^i$.

         C. Store $t^{i+1} = \sum_{j=1}^{N} xz^{i+1}_j$.

         D. Return to Step 1.

Note, it is necessary to not outright reject T^i_* in Step 3a and start over when a proposed table does not match the required value of s_2 in order to keep the chain connected.

3.2.5 Parameter Estimation

Parameters can be estimated using the MCMC random walk approach as well. In order to compute a confidence interval, use Equation (3.5). Let β_- and β_+ be the lower and upper bounds to a $(1-\alpha)\%$ confidence interval.

Then β_- is defined to be:

1. $\beta_- = -\infty$ if $t = t_{\min}$.

2. β_- is the value of β that returns a p-value of $\frac{\alpha}{2}$ from the random walk.

Similarly, β_+ can be defined as:

1. $\beta_+ = \infty$ if $t = t_{\max}$.

2. β_+ is the value of β that returns a p-value of $\frac{1-\alpha}{2}$ from the random walk.

In an analogous manner to the MUE discussed above, a point estimate $\hat{\beta}$ can be calculated by finding the value of β which returns a p-value of 0.5 from the random walk. A point estimate similar to the CMLE can also be calculated by finding the value of β which maximizes the estimate of Equation (2.23). However, a root finding method such as the bisection method is be needed to determine the value of β satisfying the conditions for the confidence interval, MUE, and CMLE. Thus estimation with the MCMC becomes very computationally intensive.

3.3 Analysis of Examples

We demonstrate and compare both the network and MCMC algorithms using the examples presented in Sections 1.1.3 and 1.1.4.

3.3.1 Toxicology Data

The data in Table 1.3 are from Bradstreet and Liss [5]. One hundred female mice were randomized to either control or one of three dose levels (8, 80, or 800 mg/kg) of a potentially harmful drug, then mated with untreated male mice. Twenty-five mice were assigned to each category; however, not all were impregnated. At day seventeen of gestation, the animals were sacrificed and their offspring observed for

Table 3.5: Parameter Estimates from the Mice Malformation Data

Method	Estimate	two-sided p-value	LCL	UCL
Asymptotic	0.006	0.1336	-0.002	0.014
CMLE	0.006	0.1182	-0.002	0.014

malformations. Table 1.3 displays, for each mouse, the number of malformations as a proportion of total offspring.

Network Estimation

The exact distribution of t conditioned on s_1 and s_2 is displayed in the top panel of Figure 3.3. We can see that the distribution of t is skewed and multimodal. The mean of the distribution is 739.1 and the standard deviation is 221.9. The observed value of 1,086 lies at the far right of the distribution. Table 3.5 shows the asymptotic parameter estimate and confidence for β and the CMLE estimate and exact confidence interval for β are nearly identical. However, the exact p-value is smaller than its asymptotic counterpart. The one-sided exact p-value for the hypothesis test $H_o : \beta = 0$ vs. $H_0 : \beta > 0$ is 0.0591 and the two-sided p-value is 0.1182.

MCMC Estimation

We ran the MCMC algorithm described in Section 3.2 for the toxicology data. Table 3.6 shows the resultant one-sided p-values from twenty separate chains. The penultimate row shows the average of the twenty chains, and the final row shows the true distribution as calculated by the network algorithm. All MCMC chains were combined, and the resulting distribution is graphed in the bottom panel of Figure 3.3. A comparison of the MCMC approximation with the exact distribution shows that this procedure is quite successful in this case.

We calculated a CMLE estimate of 0.006 for β with a two-sided 95% confidence

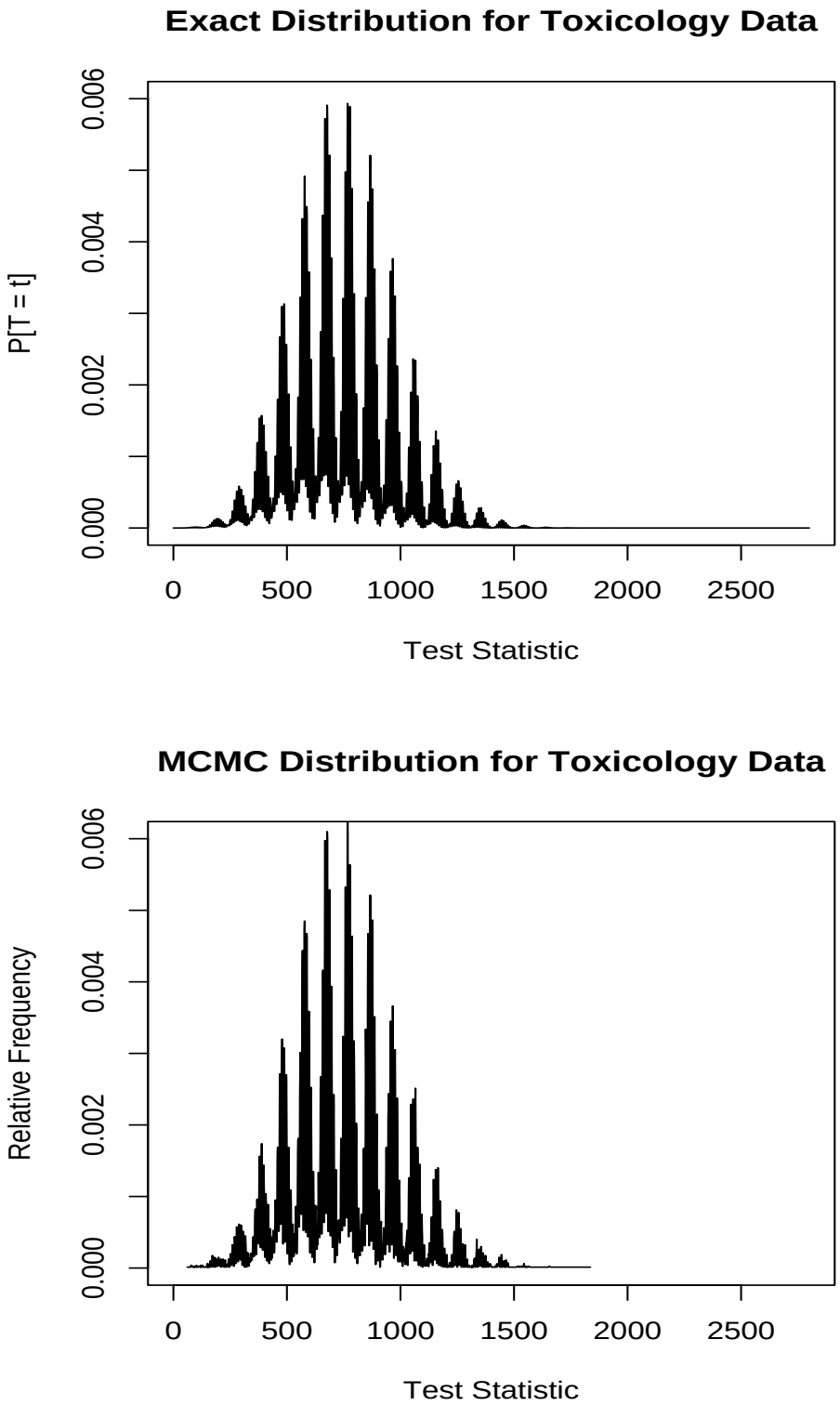

Fig. 3.3: Exact and Estimated MCMC Conditional Distribution of T.

Table 3.6: Results of 20 MCMC Runs for the Data in Table 3.1

Chain	p-value
1	0.0575
2	0.0555
3	0.0535
4	0.0545
5	0.0535
6	0.0605
7	0.0555
8	0.0635
9	0.0490
10	0.0640
11	0.0565
12	0.0575
13	0.0550
14	0.0555
15	0.0605
16	0.0610
17	0.0500
18	0.0540
19	0.0580
20	0.0530
Average	0.0564
Exact	0.0591

Table 3.7: P-values for Several Choices of β for the Toxicology Data in Table 1.3

β	p-value
0.000	0.058075
-0.002	0.021075
-0.0015	0.027537
-0.001	0.035450
0.014	0.979237
0.012	0.931788
0.013	0.961313
0.006	0.468688
0.007	0.567013
0.0065	0.514150
0.00625	0.490963

interval of (-0.002, 0.014) using the network algorithm for these data. In order to obtain parameter estimates, we ran our MCMC random walk for with 50,000 burn-in steps, then for 4,000,000 tables, saving every 50th, for a total of 80,000 tables. Table 3.7 presents the resultant p-values for several choices of β. The endpoints of the 95% confidence interval correspond to the values of β which return p-values of 0.025 and 0.975. The MCMC point estimate $\hat{\beta}$ is the value of β which returns a p-value of 0.5. $\beta = 0.006$, the estimate calculated from the exact distribution, gives us a p-value of 0.469. This is slightly lower than 0.5 which is close to the p-value observed when $\beta = 0.00625$. For the endpoints of the confidence interval, p-values of 0.0211 for $\beta = -0.002$ and 0.979 for $\beta = 0.014$ were obtained. MCMC sampling error accounts for the slight disagreement with the exact results.

3.3.2 Corneal Grafts

Table 1.4 shows the rejection results of corneal grafts for nine children categorized into two age groups. Recall these data posed complications for methods that rely

Table 3.8: Exact Permutation Distribution for $t = \sum z$

t	P[T = t]
0	0.0952
1	0.2857
2	0.2857
3	0.2857
4	0.0476

Table 3.9: Parameter Estimates from the Corneal Graft Data

Method	Estimate	two-sided p-value	LCL	UCL
Asymptotic	.	.	.	.
CMLE	∞	0.096	-0.29	∞

on large sample procedures. The small sample size and the fact that all rejections occurred in the older age group impacted our ability to estimate the age effect.

Network Estimation

For this problem, we are interested in finding the conditional distribution of $t = \sum xz$ while controlling for $s_1 = \sum z = 4$ and $s_2 = \sum z(n-z) = 2$. The exact permutation distribution of t is displayed in Table 3.8. One can see that our observed value, $t = 4$, is at the maximum of the distribution. We estimated the trend parameter, β for these data using both asymptotic and exact theory. The results are displayed in Table 3.9. One can see the effect of separation on the asymptotic method, which was unable to calculate an estimate or a p-value. The exact method does provide a p-value and confidence interval. The 95% two-sided confidence interval is $(-0.29, \infty)$. Since the observed value of t falls at the maximum of its conditional distribution, the CMLE estimate is ∞. In this case we calculated the MUE. For these data the MUE for β is 1.94, indicating that older children are $\exp(1.94) = 7.0$

times more likely to have rejections in both eyes given that one eye rejected a graft compared to younger children with two grafts and one rejection.

Note, the one-sided p-value for testing $H_0 : \beta = 0$ vs. $H_0 : \beta > 0$ is equal to 0.0476. It is also possible to calculate a one-sided confidence interval for β. For this example the one-sided lower confidence bound is 0.023, which is greater than zero and agrees with the one-tail p-value of 0.048.

MCMC Estimation

We ran our algorithm as described on the eye data. Table 3.10 shows the results of twenty separate chains. The penultimate row displays the average of the twenty chains and the final row shows the true distribution as calculated by the network algorithm. Again the MCMC method successfully approximates the exact conditional distribution.

Recall the network estimate of β was 1.94 with a confidence interval of (-0.29, ∞) for these data. We ran our MCMC random walk with 50,000 burn-in steps, then for 4,000,000 tables, saving every 50th, for a total of 80,000 tables. Table 3.11 presents the resulting p-values for several choices of β. For β of 1.94, we observe a p-value of 0.0507. For $\beta = -0.29$ we obtain a p-value of 0.0252. Again, our MCMC procedure appears to be successfully approximating the exact parameter estimates and confidence intervals.

Table 3.10: Results of 20 MCMC Runs for the Data in Table 1.4

Chain	P[$T = 0$]	P[$T = 1$]	P[$T = 2$]	P[$T = 63$	P[$T = 4$]
1	0.1040	0.2940	0.2905	0.2660	0.0455
2	0.0975	0.2860	0.2855	0.2880	0.0430
3	0.1015	0.2825	0.2950	0.2710	0.0500
4	0.1050	0.2995	0.2805	0.2705	0.0445
5	0.1060	0.3075	0.2750	0.2705	0.0410
6	0.0920	0.2915	0.2960	0.2725	0.0480
7	0.1065	0.2835	0.2765	0.2875	0.0460
8	0.1045	0.2850	0.2760	0.2945	0.0400
9	0.0985	0.2875	0.2825	0.2850	0.0465
10	0.0940	0.3115	0.2665	0.2835	0.0445
11	0.0960	0.2900	0.2865	0.2800	0.0475
12	0.0970	0.2970	0.2875	0.2680	0.0505
13	0.1000	0.2905	0.2875	0.2720	0.0500
14	0.1010	0.2795	0.2840	0.2905	0.0450
15	0.0945	0.2760	0.2915	0.2915	0.0465
16	0.0950	0.2895	0.2885	0.2805	0.0465
17	0.0915	0.2890	0.2910	0.2850	0.0435
18	0.0930	0.3060	0.2790	0.2725	0.0495
19	0.1060	0.2785	0.2945	0.2820	0.0390
20	0.0880	0.2900	0.2905	0.2855	0.0460
Average	0.098575	0.290725	0.285225	0.279825	0.04565
Exact	0.095	0.286	0.286	0.286	0.0476

Table 3.11: P-values for Several Choices of β for the corneal grafts data in Table 1.4

β	p-value
0	0.0486
-0.5	0.015213
-0.25	0.028575
-0.30	0.025112
-0.29	0.025175
-0.28	0.026875
-0.32	0.023913
-0.31	0.024925
-0.305	0.024225
-0.3025	0.024612
-0.3015	0.024048
2.00	0.5200050
1.94	0.506650

CHAPTER 4

SMALL SAMPLE PROPERTIES OF PARAMETER ESTIMATES

This study was conducted to demonstrate properties of estimates for β calculated using the three methodologies presented: maximum likelihood estimates (MLE), conditional maximum likelihood estimates (CMLE), and median unbiased estimates (MUE) in small sample, sparse data sets.

4.1 Simulation Settings

Several simulations were undertaken, always generating data from the quadratic exponential model given in Equation (2.17). The baseline probability of a success with no correlation was held constant by setting $\alpha = -2$, which corresponds to a probability of a success of $\frac{\exp(-2)}{1+\exp(-2)} = 0.12$. We considered three values of the correlation parameter, δ: 0, 0.1, and 0.5. A value of $\delta = 0$ indicates that there is no correlation among cluster-mates, while a value of 0.5 indicates a great deal of correlation. Three different values of the trend parameter, β, were considered: 0, 0.7, and 1.1. When $\beta = 0$ there is no increase in the conditional odds ratio of a response regardless the value of x. $\beta = 0.7$ represents a doubling of the conditional odds ratio of response for every unit increase in x, while $\beta = 1.1$ is equivalent to a three-fold increase in the conditional odds ratio for each unit increase in x. We considered several levels of the explanatory variable, x. Simulations were carried out when x only had two levels (0 and 1), three levels (0, 1, and 2), and four levels (0, 1, 2, and 3). Regardless of the setting, we held the number of clusters per level of x at three. Thus, when $x \in \{0, 1\}$ there were only six clusters, while when $x \in \{0, 1, 2, 3\}$ there

were twelve clusters. Lastly, we considered three different cluster sizes, 2, 5, and 10. Therefore, we had 81 different parameter settings.

One thousand samples were generated for each of the 81 combinations described above. For each sample the MLE estimate and 95% confidence interval for β were calculated using the software package R. Using the network algorithm and the methods presented in Section 2.3.2, the corresponding exact CMLE estimate and 95% confidence interval were also calculated along with the corresponding MUE for each sample.

4.2 Parameter Estimate Properties

The results of the parameter estimation are presented in Figures 4.1, 4.2, and 4.3. Each figure is divided into six rows. The first three rows of each Figure display the mean parameter estimates for each of the settings of β. The first row corresponds to simulations where $\beta = 0$, the second row corresponds to $\beta = 0.7$, and the third row is for $\beta = 1.1$. On each graph there are four curves lines: one each for the CMLE (solid line with circles), MUE (dotted line with triangles), and MLE (dashed line with crosses), and the true value of β (solid straight line). The y-axis on each graph is the average parameter estimate for a simulation and the x-axis represents the cluster size 2, 5, or 10. The three columns represent the three different settings of δ 0, 0.1, and 0.5.

Rows four through six of each Figure are arranged similarly to the first three rows. However, rather than displaying the parameter estimates, these plots show the percent of samples for which an estimate was not calculable. Recall from Section 2.3.2 the CMLE does not exist when the test statistic t reaches either the maximum or minimum of its conditional distribution. Similarly the MLE does not exist when separation occurs or other conditions that may cause the likelihood to be monotone

increasing or decreasing. On the other hand, the MUE always exists, therefore it is not presented on these graphs.

4.2.1 $x \in \{0, 1\}$

Figure 4.1 displays the results of the simulation when $x \in \{0, 1\}$. The first row corresponds to results when $\beta = 0$. These plots show both the MUE and CMLE do a good job of estimating β, while the MLE does not. In all three panels of the first row the MLE consistently underestimates the true value of β. Comparing row 4 of Figure 4.1 we see that the number of samples for which an estimate is not calculable is nearly the same for the CMLE and MLE. When $\delta = 0$ and $\delta = 0.1$ it is not possible to calculate an estimate for nearly 70% of samples. When a high degree of correlation is introduced into the model, $\delta = 0.5$, the number of samples where it is not possible to use the CMLE or MLE greatly increases, especially when $n = 10$. This is due to the relationship between δ and cluster size in the QEM [42]. Molenberghs and Ryan have shown that when δ and cluster size start to increase, the individual probability of a success becomes small when $\alpha + \beta x < 0$, forcing nearly all subjects to be failures. Likewise, the individual probability of a success becomes large when $\alpha + \beta x > 0$, resulting in nearly all subjects being successes. In this situation, levels of x result in $\alpha + \beta x < 0$ which produces numerous situations where only one subject in the entire sample was a success. This scenario results in the observed value of the test statistic reaching the boundary of the conditional likelihood and separation occurring in unconditional likelihood estimation.

The second and fifth rows of Figure 4.1 show the results of the simulations when $\beta = 0.7$. A similar pattern can be seen for the parameter estimates in the first two panels of row two. When the cluster size, n, is two all three methods greatly

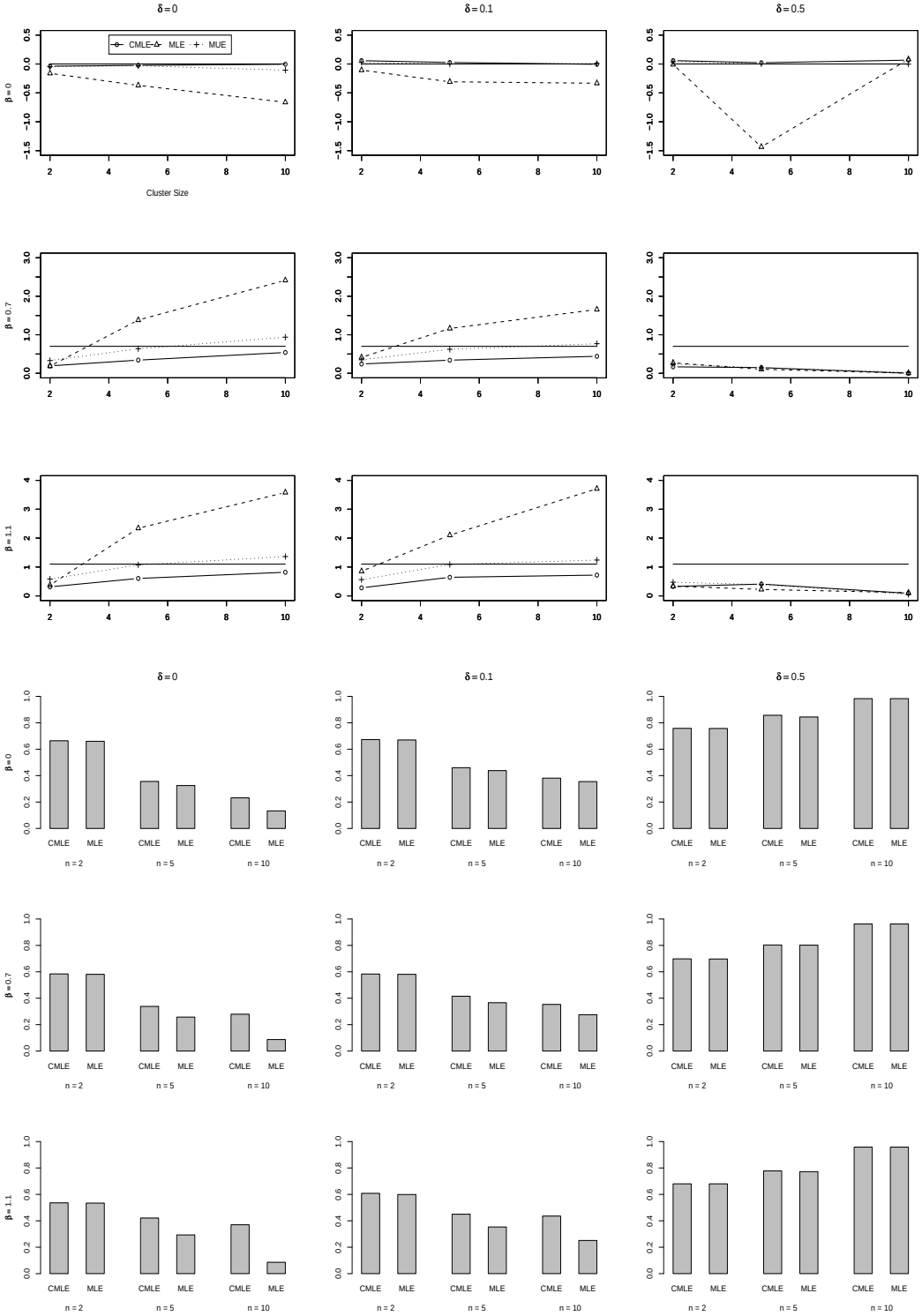

Fig. 4.1: Parameter Estimates of β and Percent of Nonestimable Samples When $x \in \{0, 1\}$.

underestimate β. As the cluster size increases the MLE estimate greatly increases, to the point where it is overestimating the truth. The CMLE tends to produce a consistent estimate which is about half of the correct value of β. The MUE on the other hand does approach the correct value of β, especially when $n = 10$. When $\delta = 0.5$ all three methods estimate β to be similar to what they predicted under the lower correlation settings. However, all three methods converge to an estimate of zero as n gets larger.

The third and sixth row display the results when $\beta = 1.1$. The results are similar to $\beta = 0.7$. In the settings with no or slight correlation the MLE underestimates the true value of β but then quickly increases so that the MLE estimate is nearly three times the true value when $n = 10$. The CMLE again stays pretty constant and produces an estimate that is about half of the true value. Again, the MUE does a reasonable job of estimating β in this situation. Finally, under the extreme correlation setting, all three estimates converge to zero as n increases.

4.2.2 $x \in \{0, 1, 2\}$

Figure 4.2 displays the results of the simulation when $x \in \{0, 1, 2\}$. The first row corresponds to results when $\beta = 0$. These plots show both the MUE and CMLE do a good job of estimating β. However, as when $x \in \{0, 1\}$ the MLE consistently underestimates the true value of β. The fourth row of Figure 4.2 shows that we are able to produce estimates for a much larger percentage of the samples than when $x \in \{0, 1\}$. However, when $\delta = 0.5$ and $n = 10$ it is still not possible for the MLE and CMLE to produce estimates for over half the samples.

The second and third rows of Figure 4.2 show when there is no to slight correlation among cluster-mates both the MLE and MUE tend to overestimate the true value

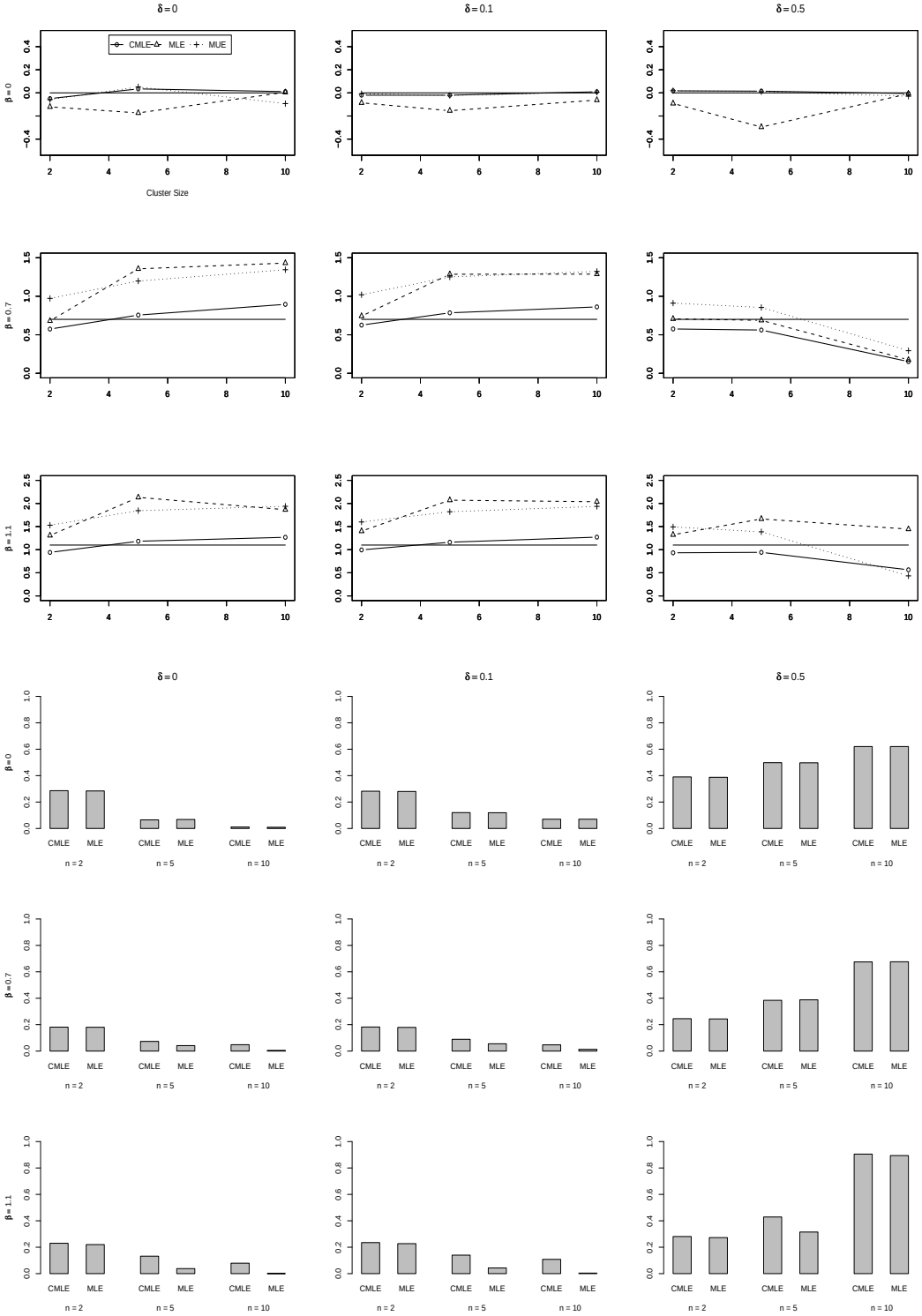

Fig. 4.2: Parameter Estimates of β and Percent of Nonestimable Samples When $x \in \{0, 1, 2\}$.

of β. The CMLE on the other hand shows that it is consistently close to β. In the extreme correlation case for $\beta = 0.7$, all three methods produce estimates close to zero when $n = 10$. However, for $n = 2$ and $n = 5$, the CMLE is again close to the true value of β, as is the MLE. The MUE, however, still overstates the true impact of x. When $\beta = 1.1$ and $\delta = 0.5$, the MLE consistently overestimates the true value of the parameter. The CMLE slightly underestimates β for the smaller cluster sizes and then decreases when $n = 10$. The MUE on the other hand, shows a steady decline as n increases. Rows five and six of Figure 4.2 show a similar pattern in the number of samples for which the MLE and CMLE could not produce estimates as when $\beta = 0$.

4.2.3 $x \in \{0, 1, 2, 3\}$

Figure 4.3 displays the results of the simulation when $x \in \{0, 1, 2, 3\}$. All three methods are able to correctly estimate the true value of the parameter when $\beta = 0$, regardless of cluster size or level of correlation between cluster-mates. Once x does impact the outcome, i.e. $\beta \neq 0$, only the CMLE estimates the true value of β accurately. When $\beta = 0.7$, the CMLE consistently estimates the correct value even in the extreme correlation setting. The MLE overestimates β in all three correlation settings. Additionally, in the extreme combination of δ and n, the MLE begins to tend downwards, whereas the CMLE stays constant. The results of MUE are even more extreme than the MLE results. When $\beta = 1.1$ the CMLE is again accurate. Only in the extreme correlation setting does the CMLE begin to perform poorly. As with $\beta = 0.7$, both the MLE and MUE greatly overestimate β and show an even more drastic effect in the extreme combination of $\delta = 0.5$ and $n = 10$. The bottom three rows in Figure 4.3 show that only in the extreme combination of $\delta = 0.5$ and $n = 10$ are there a large number of samples generated for which estimates are not calculable.

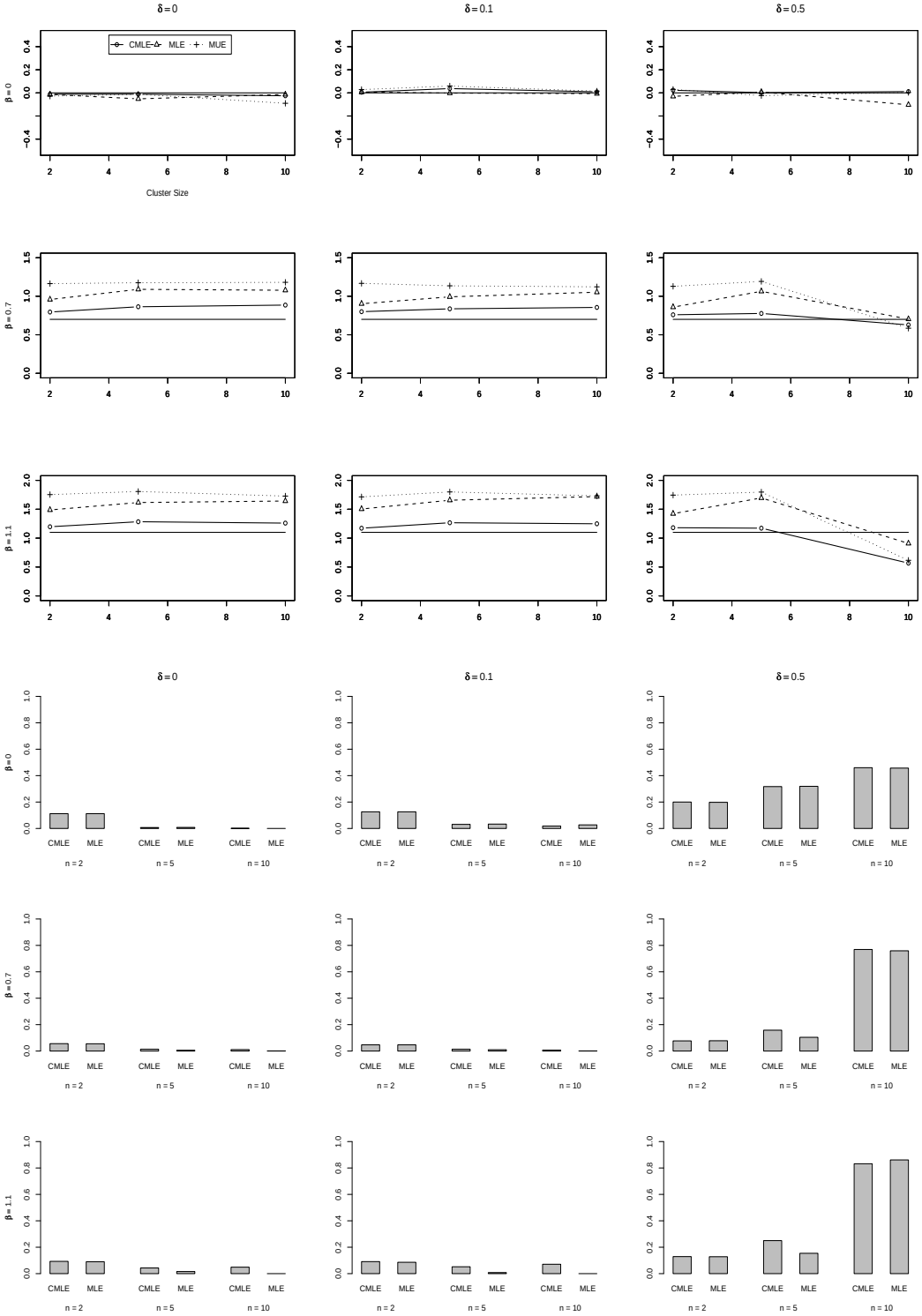

Fig. 4.3: Parameter Estimates of β and Percent of Nonestimable Samples When $x \in \{0, 1, 2, 3\}$.

4.3 Confidence Intervals

In addition to the individual parameter estimates, we considered confidence intervals. Confidence intervals were judged on average coverage and width. Since there were several different combinations of parameters, we compare confidence intervals in several ways. First, we consider cluster size, combining all other parameter levels into one graph; secondly, we consider the number of levels of x and collapse all other parameter settings. Finally, we separate the results by the value of β: 0, 0.7, and 1.1.

4.3.1 Confidence Interval Coverage

For each of the 81 combinations of parameter settings, we calculated the percent of confidence intervals that contained the correct value of β. Since 95% confidence intervals were calculated, it should be expected that the proportion of confidence intervals containing the correct value of β to be at least 95%. The results of the exact 95% confidence intervals are presented in Figure 4.4, and the results of the 95% MLE histograms are shown in Figure 4.5.

Figure 4.4 shows that the exact confidence intervals always have at least 95% coverage. However, it appears that the exact intervals quite frequently have much more than 95% coverage. In fact, only occasionally do any of the histograms have coverage of 0.95. The exact method appears to be especially conservative when $x \in \{0, 1\}$ where the overwhelming majority of intervals have close to 100% coverage. It is interesting to note that it appears that cluster size has relatively little impact on confidence interval coverage using the exact method (top row of Figure 4.4). Conversely, changing the number of levels of x, and, therefore, the number of clusters, does have a great impact on confidence interval coverage. As the number of clusters grows, the conservativeness of the exact intervals decreases substantially. The confidence

intervals for $\beta = 0$ appear to be more conservative than the confidence intervals for $\beta \in \{0.7, 1.1\}$.

Figure 4.5 shows that MLE confidence intervals are actually more conservative than their exact counterparts. As with the exact case, the most conservative situation arises when $x \in \{0, 1\}$. As x increases to three and four levels, the intervals become less conservative. However, the MLE intervals appear to be much more conservative when $\beta = 0$ than the exact intervals. Changing the cluster size does impact the coverage of the MLE confidence intervals but not to the extent as increasing the number of levels of x.

4.3.2 Confidence Interval Length

Short length is another desirable property of confidence intervals. However, by definition, many of our intervals had either infinite upper or lower bounds. Thus, the usual calculation of confidence interval length, upper bound - lower bound, is not meaningful. Therefore, we define length in terms of the Neyman shortness [20, 39]. Let d denote a fixed distance from β. Then for any fixed parameter setting we plot the proportion of the 1,000 simulations that include $\beta \pm d$. The Neyman shortness is displayed in Figure 4.6. The solid lines represent the exact confidence intervals, and the dashed lines are the MLE intervals.

There does not appear to be much change in the shortness of intervals as the cluster size increases from two to ten. It does appear that changing the number of levels of x and, therefore, the number of clusters, does change the width of the intervals quite a bit. When $x \in \{0, 1\}$, both the exact and MLE intervals have very heavy tails; both methods still have a 50% probability of containing values that are ± 3 units away from β. However, when $x \in \{0, 1, 2\}$ and $x \in \{0, 1, 2, 3\}$ the tails of both methods decrease sharply. When $x \in \{0, 1, 2, 3\}$, there is almost no chance

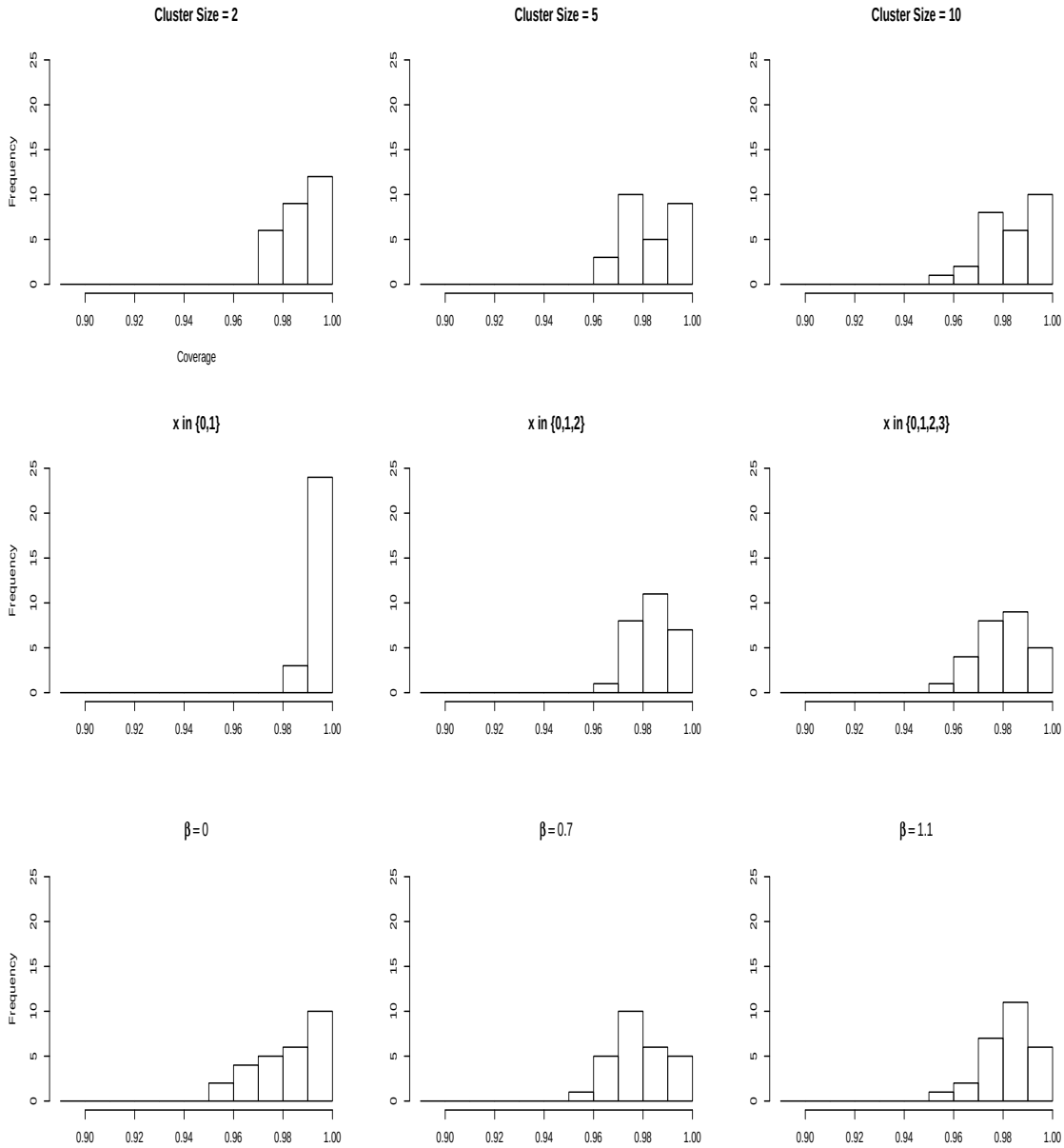

Fig. 4.4: Histograms of the Exact Confidence Interval Coverage.

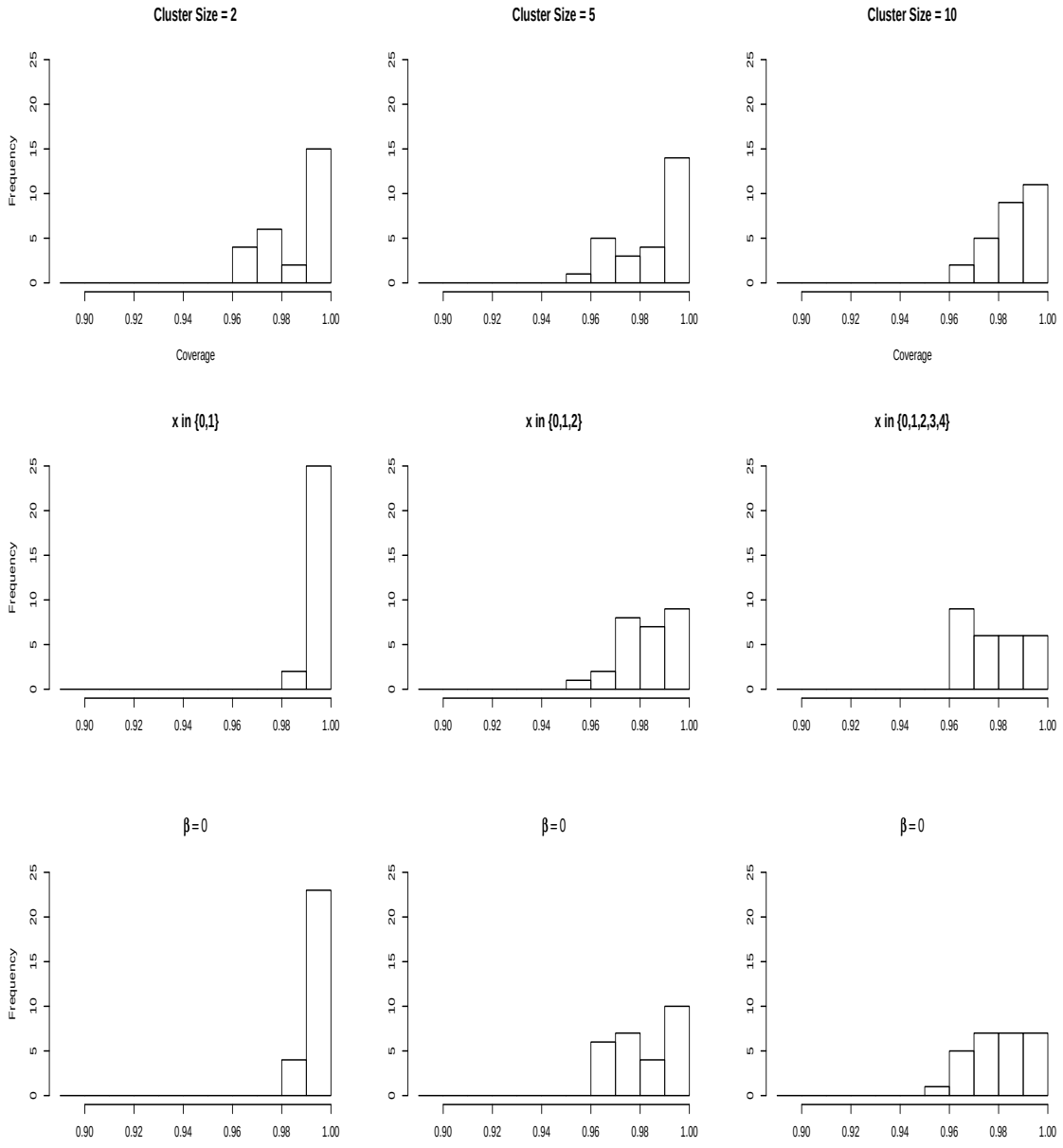

Fig. 4.5: Histograms of the MLE Confidence Interval Coverage.

that either an MLE or exact interval will contain values below $\beta - 3$. The exact intervals appear to do a better job of identifying the lower bound of the confidence interval when $x \in \{0, 1, 2, 3\}$. This is an interesting feature since we only considered situations where $\beta \geq 0$. This indicates that the exact method could be more powerful when only a one-sided interval is needed. We again see that the exact method has a shorter left tail when $\beta = 1.1$ than the MLE method. Again, this indicates that the exact method is more powerful at identifying $\beta > 0$, which is ideal when a one-sided interval is needed.

4.3.3 Discussion

In this chapter we have compared the MLE, CMLE, and MUE methods for estimating β from a QEM in several small sample settings. When comparing the parameter estimates alone, we have shown that the CMLE outperforms the MUE and MLE estimates in nearly all settings considered. The MLE frequently overstates the true value. The MUE method also tended to overestimate the true value of the parameter when $\beta > 0$. On the other hand, the MUE always exists regardless of the make-up of the sample, especially in the extreme settings ($\delta = 0.5$ and $n = 10$), where the MUE and CMLE were unable to calculate an estimate. As we have presented earlier, we suggest using a combined method that relies on the CMLE when estimable and on the MUE when the CMLE does not exist. Additionally, we found that both the exact and MLE confidence intervals always had the desired coverage. However, we found the MLE intervals to be slightly more conservative in the small sample settings considered. This suggests the best method for estimating β is to use a combination of the CMLE and MUE estimates and the exact confidence interval. This combination of point estimate and confidence interval for exact estimation and inference has been suggested before [35]. Additionally, Corcoran *et al.* has shown the exact method to

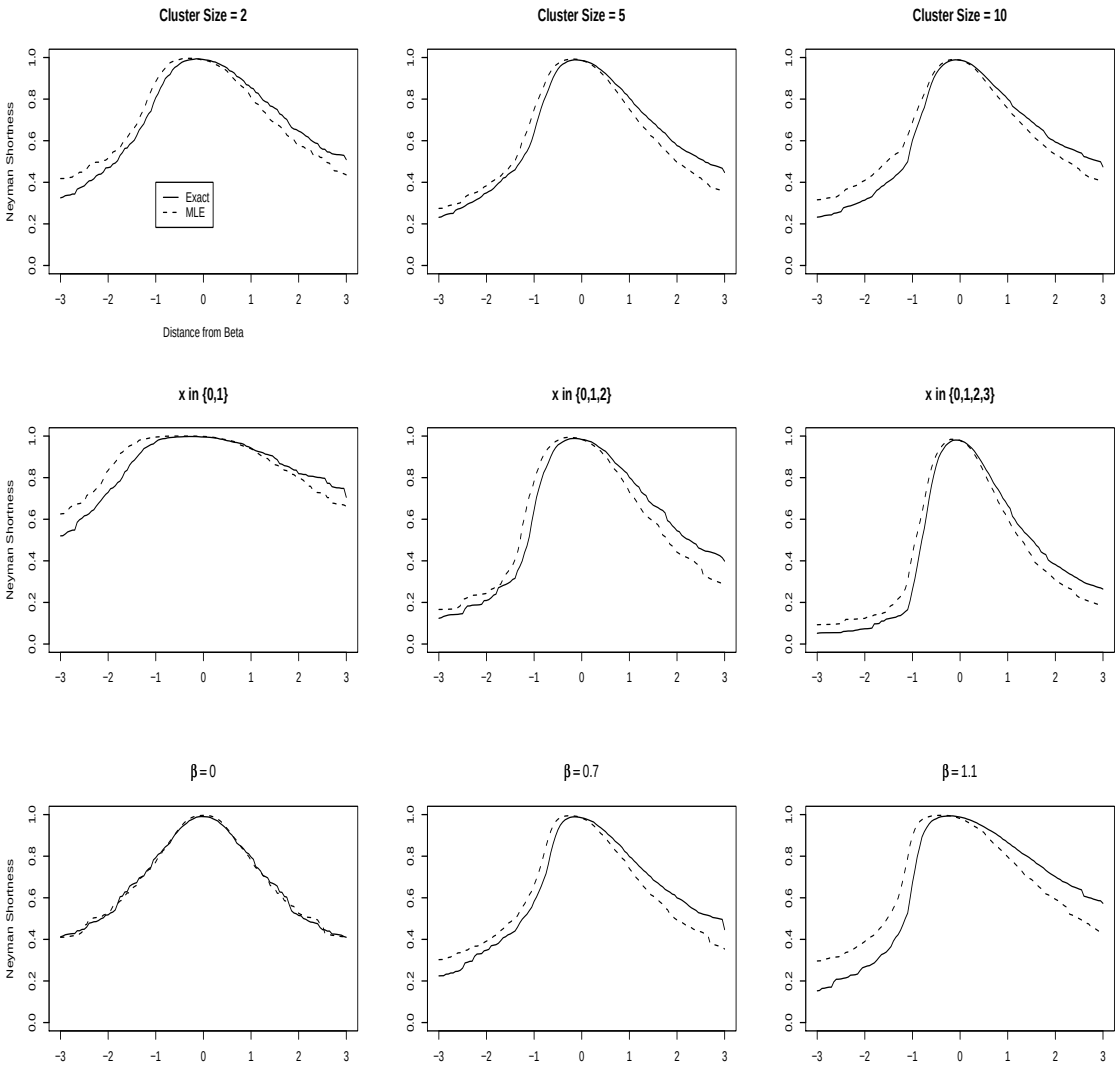

Fig. 4.6: Neyman Shortness Distance for Exact and MLE Confidence Intervals.

be more powerful than many other methods at identifying a trend in clustered binary data [8, 9]. This property was evident in our analysis of confidence interval length. Given the potential increase in power and the fact that the CMLE method produces estimates which are less biased than the MLE estimates, we recommend the exact method be used in these small sample settings.

CHAPTER 5

ACCURACY AND COMPUTATIONAL EFFICIENCY OF MCMC

In light of the shortcomings of other MCMC approaches applied to conditional inference, it is important to determine the operating characteristics of our MCMC approach [37]. With this goal in mind, we conducted simulation studies to examine the accuracy of approximations generated by the MCMC method. We designed these studies also to evaluate the computational efficiency of MCMC compared to the network algorithm.

5.1 Simulation Settings

For this study, data were generated from the quadratic exponential model given in Equation (2.17). For this simulation we were specifically interested in sparse samples with a large number of clusters. As such, we chose parameters to limit the number of successes observed. The baseline probability of a success with no correlation was held constant by setting $\alpha = -2$, corresponding to an average probability of success of 0.12 if no correlation existed among cluster-mates. The within cluster correlation was kept constant by holding $\delta = 0.1$. We used four equally spaced dose levels, and considered three different levels of β: -0.01, 0.0, 0.01. A value of $\beta = 0$ indicates no dose-response effect. Given a fixed cluster size and number of previous successes within the cluster, $\beta = 0.01$ corresponds to a 1% increase in the average log odds of an additional success. Likewise, $\beta = -0.01$ corresponds to a 1% decrease in the average log odds of an additional success under similar circumstances. Cluster size was held constant at five observations per cluster for samples with 40, 80, 120, 160, 200, 240, 280, 320, and 360 clusters.

For each combination of experimental conditions we generated 100 samples, calculating the exact permutation distribution and an MCMC approximation for each. We compared the Monte Carlo sample to the exact distribution using the chi-square goodness-of-fit test. In addition, we also compared the MCMC p-value for the test $H_0 : \beta = 0$ to the exact p-value calculated from the conditional distribution for each sample. Finally, we observed the time for both the exact and MCMC methods to finish calculating the respective distributions. We present the time distributions as box plots.

5.2 MCMC Distribution Accuracy

For each sample in the simulation, we used the network algorithm to generate the exact permutation distribution of the test statistic $t = \sum xz$. The exact conditional distribution is the gold standard to which we are comparing our MCMC approximations. As shown in Figure 3.3, exact distributions for sparse data can be very non-normal, highly skewed, and have portions of the distribution with very small probabilities. The result is that many of the expected counts for the goodness-of-fit test are less than five. For this reason, we used MCMC to approximate the chi-square goodness-of-fit p-value.

Under this scenario we are testing H_0 : the MCMC distribution approximates the gold standard exact distribution. If the MCMC distribution does in fact approximate the exact distribution, i.e. the null hypothesis is true, p-values generated from the goodness-of-fit test will follow a U(0,1) distribution. Figure 5.1 shows the QQ-plots making this comparison. The data for Figure 5.1 were created by pooling the p-values from each setting of β and stratifying by cluster size. The linearity of these plots indicates that the distribution of p-values is approximately uniform, suggesting that the MCMC method is approximating the exact conditional distribution.

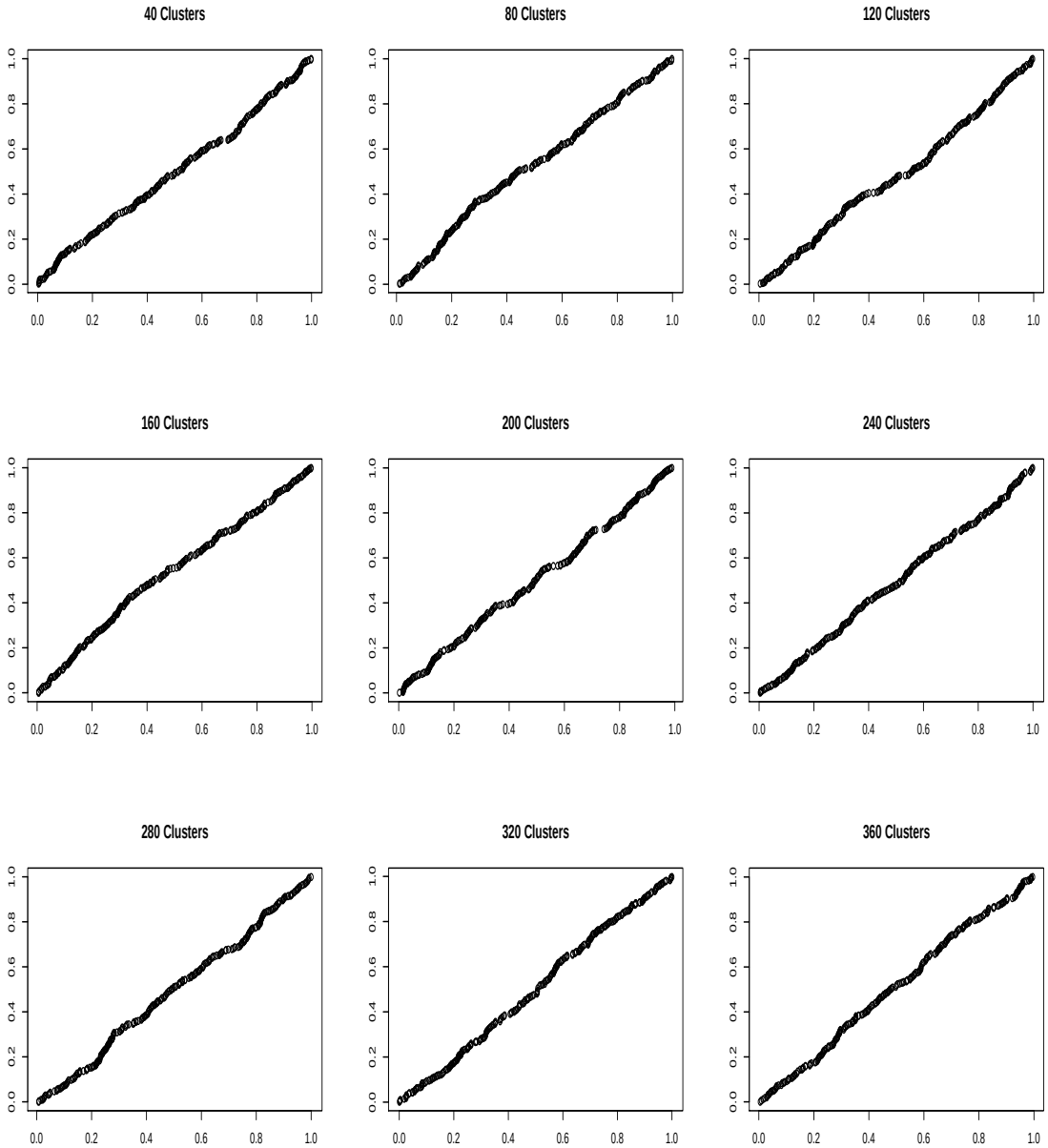

Fig. 5.1: QQ Plot of P-values from Goodness-of-Fit Tests vs. U(0,1) by Number of Clusters.

Table 5.1: Proportion of 95%Confidence Intervals Not Containing the True p-value

Sample Size	% CI's Not Containing True p-value
40	0.077
80	0.043
120	0.063
160	0.063
200	0.060
240	0.054
280	0.057
320	0.057
360	0.054

Next, we were interested in how well p-values for testing $H_0 : \beta = 0$ generated by the MCMC method approximate the true p-value calculated from the exact distribution. For this analysis we calculated the 95% confidence intervals for the proportion of tables generated by the MCMC method with test statistics as or more extreme than the observed test statistic for each simulation. Table 5.1 displays the percent of these confidence intervals which failed to cover the true p-value calculated from the exact distribution for the different cluster sizes. The true p-value was not contained in the intervals slightly more than expected. However, it does appear that these results improve as the sample size gets large - the very setting in which the MCMC approach is most helpful.

5.3 Computational Efficiency

As sample sizes get larger, more memory is needed to store all the arcs and nodes associated with the network algorithm, resulting in significant increases in run time to compute conditional distributions. In this section we compare the computation time of the network algorithm and MCMC method to determine if the MCMC method is a viable option for replacing the network algorithm for large sample problems. As

Table 5.2: Summary Statistics for Run Times (in Seconds) of the Network Algorithm

Cluster	Min	Q1	Median	Q3	Max
40	0.05	0.06	0.08	0.11	0.36
80	0.17	0.50	0.78	1.3	4.0
120	0.67	2.9	4.9	6.3	19
160	2.6	11	16	23	49
200	3.7	33	44	61	312
240	22	72	104	146	331
280	59	156	205	272	572
320	118	309	409	515	1,184
360	255	577	745	1,015	2,006

Table 5.3: Summary Statistics for Run Times (in Seconds) of the MCMC Method

Clusters	Min	Q1	Median	Q3	Max
40	9	11	19	25	581
80	10	12	20	32	793
120	12	13	20	31	602
160	12	15	24	33	1,825
200	13	17	24	42	1,669
240	15	19	29	51	4,031
280	15	22	28	52	3,987
320	17	23	32	63	142,100
360	19	25	40	86	127,700

such, we calculated the length of time it took each method to calculate the conditional distribution of the test statistic.

Table 5.2 shows, as expected, the run time in seconds increases quite dramatically as the number of clusters increases for the network algorithm. The median run time ranges from less than a tenth of a second to close to 12.5 minutes for 360 clusters. The distribution of run times for the MCMC method are summarized in Table 5.3. The summary statistics show that for relatively small sample sizes, the MCMC method is significantly slower than the network algorithm. For 40 clusters, the median run time for the network algorithm was under a tenth of second, compared to a median of

19 seconds for MCMC. The computational savings of MCMC appear to be realized for relatively larger samples. The MCMC method becomes faster than the network algorithm when sample sizes reach a point between 160 and 200 clusters. At 160 clusters, the network algorithm has a median run time of 16 seconds and the MCMC method's median run time is 23.5 seconds. However, with 200 clusters the MCMC method's median run time has only increased by half-a-second to 24, while the network algorithm's median run time has nearly tripled to 43.6 seconds. For 360 clusters the median of the MCMC method had increased to 40 seconds, compared to a median of 12.5 minutes for the network algorithm.

Figure 5.2 shows a graphical representation of the run times for the network algorithm and MCMC method by cluster size. The top panel displays run times for the network algorithm, and the bottom panel presents run times for the MCMC method. Run time is defined as log base 10 of the number of seconds required to calculate the distribution. The impact that sample size has on the network algorithm is quite apparent. Unlike the network algorithm, the MCMC run times only slightly increase as the number of clusters increases. One concern with MCMC is the greater right tail variability of run times. Across all cluster sizes, the run time for MCMC seems prone to some outliers. For example the maximum run time for 320 clusters was over 39 hours. However, the other MCMC run times, even the outliers, were significantly faster. Finally, we reran the analysis for 360 clusters, increasing cluster size from 5 to 15 subjects. The result was the network algorithm was unable to store all the necessary information and crashed, however, the MCMC method frequently completed samples of this size in under 30 seconds.

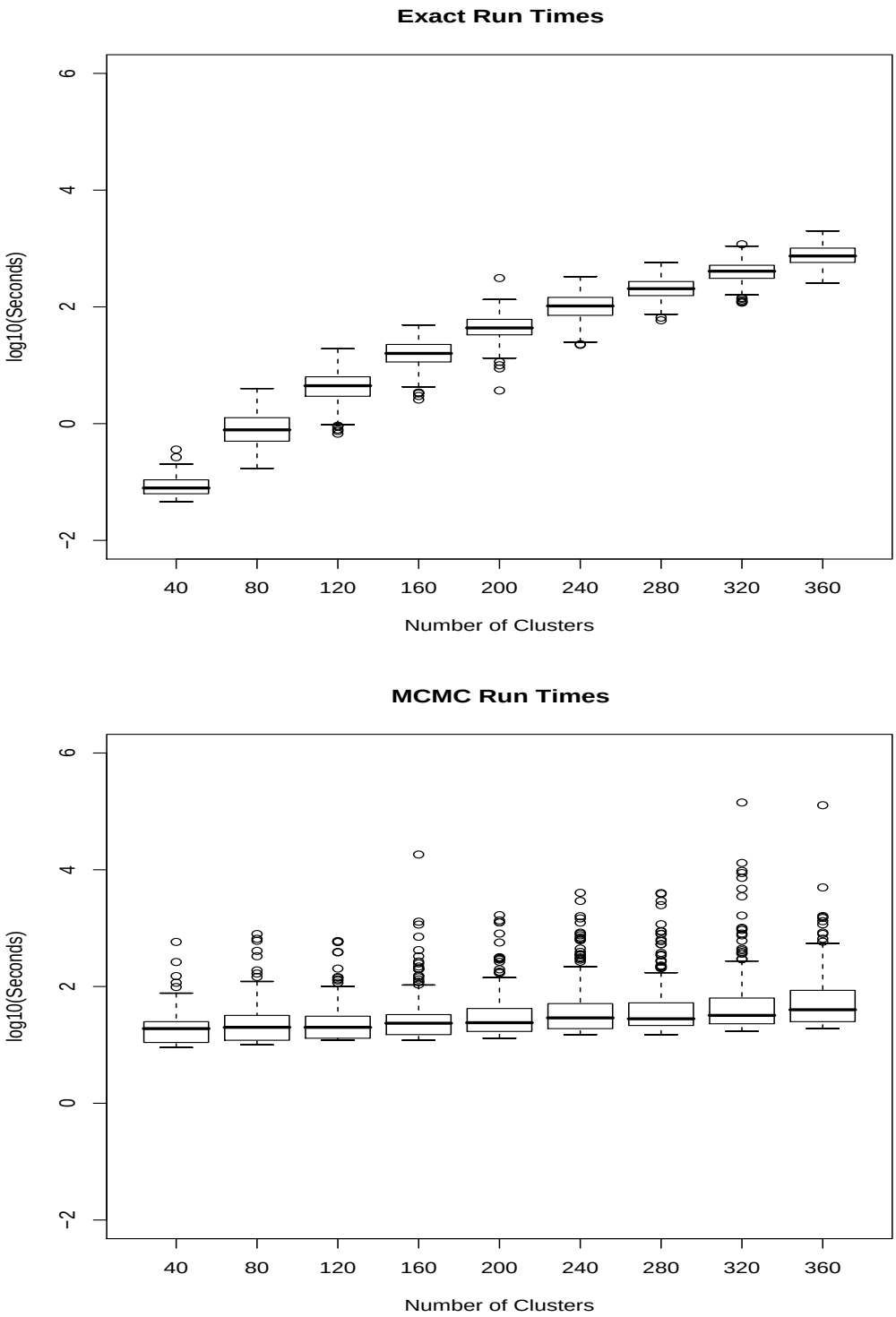

Fig. 5.2: Box Plot of Log Base 10 Run Times for Network and MCMC Distribution Calculations.

5.4 Computational Burden of Rejection Sampling

Since our MCMC method uses rejection sampling, we hypothesized that high rejection rates may be the root cause of run time outliers. For this investigation, we used the network algorithm to calculate the number of tables in each of $\Gamma(s_1)$ and $\Gamma(s_1, s_2)$ as defined in Section 3.1. The MCMC method is guaranteed to produce tables in $\Gamma(s_1)$ - we are only rejecting those tables not in $\Gamma(s_1, s_2)$. Thus, if the cardinality of $\Gamma(s_1)$ is significantly larger than the cardinality of $\Gamma(s_1, s_2)$, our algorithm may consume an inordinate amount of time rejecting tables not in $\Gamma(s_1, s_2)$. The results of this comparison are displayed in Figure 5.3. For the x-axis in Figure 5.3, we computed the log base 10 of the ratio $\frac{\text{Cardinality of } \Gamma(s_1,2)}{\text{Cardinality of } \Gamma(s_1)}$. A value of -2 on the x-axis states that for every table in $\Gamma(s_1, s_2)$, there are $10^2 = 100$ tables in $\Gamma(s_1)$. Therefore, our method would expect to reject about 99 tables for every table that is retained. The y-axis in Figure 5.3 represents the log base 10 of the run time for a particular sample. The outlier in the upper left corner corresponds to a sample requiring over 100,000 tables to be sampled from $\Gamma(s_1)$ to yield one table from $\Gamma(s_1, s_2)$. This particular case took over 100,000 seconds to estimate the exact conditional distribution.

5.5 Discussion

In this chapter we have examined the MCMC method's approximation to the exact conditional distribution and compared its efficiency to that of the state-of-the-art tool for exact inference, the network algorithm. While the network algorithm calculates the true exact distribution, based on our comparison the MCMC method provides a fairly accurate approximation. For small to moderately large sample sizes the network algorithm is faster than the MCMC method. However, we recommend MCMC for samples with over 150 clusters. Additionally, the computational time for the MCMC method tends to remain constant as the number of clusters become

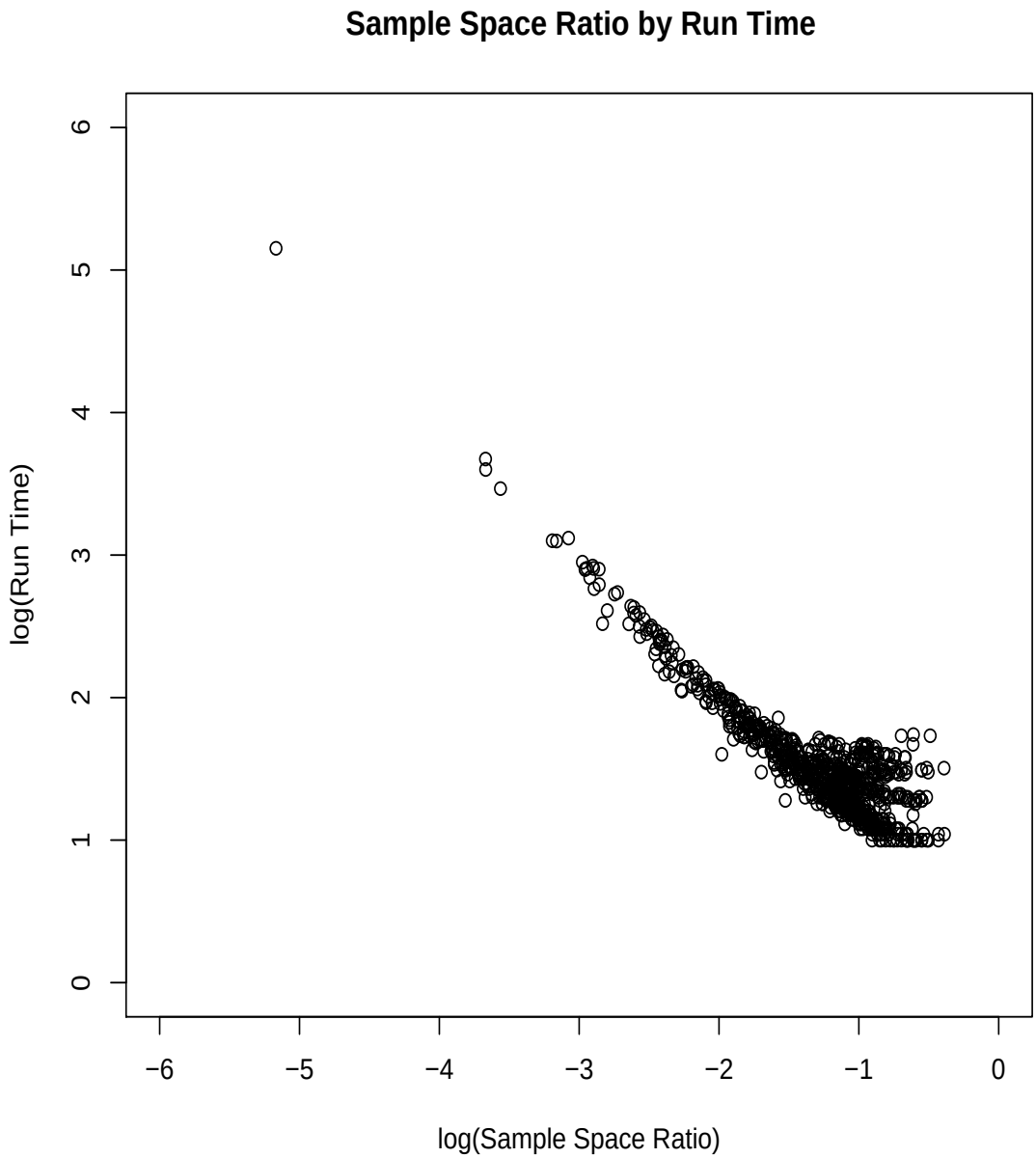

Fig. 5.3: Scatter Plot of log(Sample Space Ratio) by log(Run Time).

extremely large. The drawback of the MCMC approach appears to be the somewhat rare table requiring an inordinately large solution time. This phenomenon is directly related to the ratio of the number of tables satisfying the linear constraint (cardinality of $\Gamma(s_1)$) to the number of tables satisfying both the linear and quadratic constraints (cardinality of $\Gamma(s_1, s_2)$). Unfortunately, we have not been able to discover an *a priori* method for predicting which tables will have large differences between $\Gamma(s_1)$ and $\Gamma(s_1, s_2)$ and which will not. Therefore, for large samples, we suggest monitoring the time it takes the MCMC method to produce tables in the $\Gamma(s_1, s_2)$. If the time betweens tables appears to be large, the network algorithm should be used. Based on Figure 5.3 and Tables 5.2 and 5.3, we suggest that if the rejection rate increases to 1,000 rejected tables for every table in $\Gamma(s_1, s_2)$ or greater, the MCMC method be abandoned in favor of the network algorithm.

CHAPTER 6

SUMMARY

There are several solutions for analysis of clustered binary data. However, the two most common tools in use today, generalized estimating equations and random effects or mixed models rely heavily on asymptotic theory. In this dissertation we have presented examples where the asymptotic assumptions may not be met. Therefore we have explored the utility of the quadratic exponential model and conditional analysis to estimate the effect size of a trend parameter in small sample and sparse data settings. Our findings indicate that conditional estimates do indeed outperform their unconditional maximum likelihood counterparts.

Efficient algorithms are needed to generate the reference set for conditional estimation to be feasible. We have shown that the network algorithm is a highly useful tool for this situation. However, one of the disadvantages of the network algorithm is that as the number of clusters and subjects increases, the computational overhead also greatly increases. This had led us to investigated the use of Markov chain Monte Carlo sampling as a means to approximate the conditional distribution for large sample sizes. Our approach uses a combination of the Diaconis and Sturmfels method for Fisher's Exact Test and rejection sampling. Our simulations show that once the number of clusters becomes large, say more than 150, the MCMC method is more efficient than the network algorithm. Additionally, we have shown that MCMC appears to provide an accurate approximation of the true distribution and p-value for testing $H_o : \beta = 0$ as calculated by the network algorithm.

Finally, this conditional approach allows us to address many of the past criticisms regarding conditional models. The first criticism of conditional models is that

interpretation of model parameters depends on cluster size. We have demonstrated this property in Table 1.6. However, our estimates are computed by conditioning on the cluster sizes. Another property apparent in Table 1.6 is that the interpretation of parameters are dependent on the outcomes of other cluster-mates. Our method also conditions on the total number of successes in the data set. One additional criticism of conditional models is that the interpretation of one parameter depends on the values of other parameters. Again, our method addresses this concern by conditioning nuisance parameters out of the likelihood resulting in a distribution containing only the parameter of interest, β.

There are several potential studies that can be carried forward from this research. The most pressing is the need to include additional covariates in the model. Additional conditioning required by an increase in covariates will produce more computational overhead for the network algorithm and its performance will decline. Diaconis and Sturmfels provide further sets of moves beyond the ± 1 moves described in Section 3.2 which may be useful for extending the MCMC approach to more than one covariate. However, their research involves independent outcomes. Work will be needed to determine how to best incorporate these additional moves into the clustered data setting. Our research only considers an exchangeable correlation structure. Future work should investigate other correlation structures. Autoregressive correlation may allow us to extend our method to longitudinal studies for instance. It is likely that different correlation structures will impose different conditioning requirements than have been explored here. Another potential area of future research should be directed at introducing subject level covariates to the model. We have only investigated cluster level covariates. Variables which can vary between cluster-mates need to be explored. Again, these types of variables are likely to impose different conditioning requirements than the ones which we have explored. Finally, we have suggested re-

placing the CMLE with the MUE when the CMLE is infinite. Similar solutions exist in unconditional estimation, such as using the penalized maximum likelihood estimation (PMLE) when the likelihood is monotone [23]. Future studies should compare the performance of the MUE and other options such as the PMLE in these extreme situations.

REFERENCES

[1] Agresti A., *Categorical Data Analysis*, 2nd edition ed., John Wiley & Sons, Inc., New York, 2002.

[2] M. Aerts, L. Declerck, and G. Molenberghs, *Estimation efficiency and tests of covariate effects with clustered binary data*, Biometrics **49** (1993), pp. 989–996.

[3] D.A. Anderson and M. Aitkin, *Variance component models with binary response: interviewer variability*, J. Roy. Statist. Soc. Ser. B **47** (1985), pp. 203–210.

[4] Y.M.M. Bishop, S.E. Fienberg, and P.W. Holland, *Discrete Multivariate Analysis: Theory and Practice*, MIT Press, 1975.

[5] T.E. Bradstreet and C.L. Liss, *Favorite data sets from early (and late) phases of drug research – part 4*, ASA Proceedings of the Section on Statistical Education (1995), pp. 335–340.

[6] N.E. Breslow and N.E. Day, *Statistical Methods in Cancer Research, Vol.1 – The Analysis of Case–Control Studies*, IARC, Lyon, 1980.

[7] M.R. Conaway, *Analysis of repeated categorical methods*, J. Amer. Statist. Assoc. **84** (1989), pp. 53–62.

[8] C. Corcoran, *An exact trend test for correlated binary data*, Ph.D. thesis, Harvard, Boston, 1996.

[9] C. Corcoran, L. Ryan, P. Senchaudhuri, C. Mehta, N. Patel, and G. Molenberghs, *An exact trend test for correlated binary data*, Biometrics **57** (2001), pp. 941–948.

[10] D.R. Cox, *The analysis of binary data*, Applied Statistics **21** (1972), pp. 113–120.

[11] D.R. Cox and D.V. Hinkley, *Theoretical Statistics*, Chapman and Hall, London, 1974.

[12] D.R. Cox and N. Wermuth, *A note on the quadratic exponential binary distribution*, Biometrika **81** (1972), pp. 403–408.

[13] P. Diaconis, D. Eisenbud, and B. Sturmfels, *Lattice walks and primary decomposition*, in Mathematical Essays in Honor of Gian–Carlo Rota (B.E. Sagan and R. P. Stanley, eds.), Birkhauser, Boston, 1998, pp. 173–193.

[14] P. Diaconis, R.L. Graham, and B. Sturmfels, *Primitive partition identities*, in Combinatorics Paul Erdős is Eighty (D. Miklos, T. Szonyi, and V.T. Sos, eds.), Bolyai Society Mathematical Studies Series, vol. 2, American Mathematical Society, Budapest, 1996, pp. 173–191.

[15] P. Diaconis and B. Sturmfels, *Algebraic algorithms for sampling from conditional distributions*, Annals of Statistics **26** (1998), no. 1, pp. 363–397.

[16] P.J. Diggle, P. Heagerty, K. Y. Liang, and S.L. Zeeger, *Analysis of Longitudinal Data*, 2 ed., Oxford University Press, 2002.

[17] G. M. Fitzmaurice and N. M. Laird, *A likelihood–based method for analysing longitudinal binary responses*, Biometrika **80** (1993), no. 1, pp. 141–151.

[18] J.J. Forster, J.W. McDonald, and P.W.F. Smith, *Markov chain monte carlo tests for log–linear and logistic models*, J. Roy. Statist. Soc. Ser. B **58** (1996), pp. 445–453.

[19] S. Geman and D. Geman, *Stochastic relaxation, gibbs distributions, and the bayesian restoration of images*, IEEE trans. Pattn. Anal. Mach. Inetl. **6** (1984), pp. 721–741.

[20] B.K. Ghosh, *A comparison of some approximate confidence intervals for the binomial parameter*, J. Amer. Statist. Assoc. **74** (1979), pp. 894–900.

[21] J.K. Haseman and L.L. Kupper, *Analysis of dichotomous response data from certain toxicological experiments*, Biometrics **35** (1979), pp. 281–294.

[22] W.K. Hastings, *Monte carlo sampling methods using markov chains and their applications.*, Biometrika **57** (1970), pp. 97–109.

[23] G. Heinze and Schemper M., *A solution to the problem of separation in logistic regression*, Stats. Med. **21** (2002), pp. 2409–2419.

[24] K.F. Hirji, C.R. Mehta, and N.R. Patel, *Computing distributions for exact logistic regression*, J. Amer. Statist. Assoc. **82** (1997), pp. 1110–1117.

[25] L.K. Hyde, L.J. Cook, L.M. Olson, H.B. Weiss, and J.M. Dean, *Effect of motor vehicle crashes on adverse fetal outcomes*, Obstet. Gynecol. **102** (2003), pp. 279–286.

[26] L.K. Jones and P.J. O'Neil, *On markov chain monte carlo algorithms for computing conditional expectations based on sufficient statistics*, J. Comput. Graph. Statist. **11** (2002), no. 3, pp. 660–677.

[27] M.R. Karim and S.L. Zeger, *Generalized linear models with random effects: Salamander mating revisited*, Biometrics **48** (1992), pp. 631–644.

[28] J.C. Kleinman, *Proportions with extraneous variance: single and independent samples*, J. Amer. Statist. Assoc. **68** (1984), pp. 46–54.

[29] J.E. Kolassa and M.A. Tanner, *Approximate conditional inference in exponential families via the gibbs sampler*, J. Amer. Statist. Assoc. **89** (1994), pp. 697–702.

[30] K.Y. Liang and P. McCullagh, *Case studies in binary dispersion*, Biometrics **49** (1993), pp. 623–630.

[31] K.Y. Liang and S.L. Zeger, *Longitudinal analysis using generalised linear models*, Biometrika **73** (1986), pp. 13–22.

[32] K.Y. Liang, S.L. Zeger, and B. Qaqish, *Multivariate regression analyses of categorical data*, J. Roy. Statist. Soc. Ser. B **54** (1992), pp. 3–40.

[33] S.R. Lipsitz, N.M. Laird, and D. P. Harrington, *Generalized estimating equations for correlated binary: using the odds ratio as a measure of association*, Biometrika **78** (1991), pp. 153–160.

[34] P. McCullagh and J.A. Nelder, *Generalized Linear Models*, vol. 2, Chapman & Hall/CRC, 1999.

[35] C.R. Mehta and N. Patel, *Logxact 5: User Manual*, CYTEL Software Corporation, Cambridge, MA, 2002.

[36] C.R. Mehta, N. Patel, and P. Senchaudhuri, *Exact stratified linear rank tests for ordered categorical and binary data*, J. Computat. and Graph. Statist. **1** (1992), no. 1, pp. 21–40.

[37] ———, *Efficient monte carlo methods for conditional logistic regression*, J. Amer. Statist. Assoc. **95** (2000), no. 449, pp. 99–108.

[38] C.R. Mehta and N.R. Patel, *Exact logistic regression: Theory and examples*, Stats. Med. **14** (1995), pp. 2143–2160.

[39] C.R. Mehta and S.J. Walsh, *Comparison of exact, mid–p, mantel–haenzel confidence intervals for the common odds ratio across several two by two contingency tables*, Amer. Statist. **46** (1992), pp. 146–150.

[40] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller, *Equations of state calculations by fast computing machine*, J. Chem. Phys. **21** (1953), pp. 1087–1091.

[41] G. Molenberghs, L. Declerck, and M. Aerts, *Misspecifying the likelihood for clustered binary data*, Comput. Statist. Data Anal. **26** (1998), pp. 327–349.

[42] G. Molenberghs and L. M. Ryan, *An exponential model for clustered multivariate binary data*, Environmentrics **10** (1999), pp. 279–300.

[43] J.M. Neuhaus, *Estimation efficiency and tests of covariate effects with clustered binary data*, Biometrics **49** (1993), pp. 989–996.

[44] J.M. Neuhaus, J.D. Kalbfleisch, and W.W. Hauck, *A comparison of cluster–specific and population–averaged approaches for analyzing correlated binary data*, Internat. Statist. Rev. **59** (1991), pp. 25–35.

[45] J. Palmgren and S. Ripatti, *Fitting exponential family mixed models*, Stat. Model. **2** (2002), pp. 23–38.

[46] J.F. Pendergast, S.J. Gange, M.A. Newton, M.J. Lindstrom, M. Palta, Fisher, and M.R., *Estimation efficiency and tests of covariate effects with clustered binary data*, Biometrics **49** (1993), pp. 989–996.

[47] R.M. Royall, *Model robust confidence intervals using maximum likelihood estimators*, Internat. Statist. Rev. **54** (1986), pp. 221–226.

[48] D.A. Schaumberg, A.L. Moyes, J.A. Gomes, and M.R. Dana, *Corneal transplantation in young children with congenital hereditary endothelial dystrophy*, Am. J. Ophthalmol. **127** (1999), pp. 373–378.

[49] R. Stiratelli, N. Laird, and J.H. Ware, *Random effects model for serial observations with binary response*, Biometrics **40** (1984), pp. 961–971.

[50] A.R. Takken, *Monte carlo goodness–of–fit tests for discrete data*, Ph.D. thesis, Stanford, Stanford, 1993.

[51] R. Wedderburn, *Quasilikelihood functions, generalized linear models and the gauss–newton method*, Biometrika **61** (1974), pp. 439–447.

[52] H. White, *Maximum likelihood estimation of misspecified models*, Econometrica **50** (1982), pp. 1–26.

[53] D.A. Williams, *Extra–binomial variation in logistic linear models*, Appl. Statist. **33** (1982), pp. 144–148.

[54] F. Yates, *Tests of significance for 2 x 2 contingency tables*, J. Roy. Statist. Soc. Ser. A **147** (1984), pp. 426–463.

[55] S.L. Zeger and M.K. Karim, *Generalized linear models with random effects: A gibbs sampling approach*, J. Amer. Statist. Assoc. **86** (1989), pp. 79–86.

[56] S.L. Zeger, L.Y. Liang, and P.S. Albert, *Models for longitudinal data: a generalized estimating equation approach*, Biometrics **44** (1988), pp. 1049–1060.

[57] L.P. Zhao and R.L. Prentice, *Correlated binary regression using a quadratic exponential model*, Biometrika **77** (1990), pp. 642–648.

APPENDICES

APPENDIX A

PROGRAMS FOR PARAMETER ESTIMATION FROM THE NETWORK ALGORITHM

A.1 RunAllNoRecurse.txt - R program for calculating network algorithm

```
GetFirstNodes = function(Zstar,Ns) {

Stages = length(Ns)
NodeNumber = 1
Nodes = matrix(c(Stages,Zstar,NodeNumber),nrow=1,ncol=3)
Arcs = matrix(c(0,0,0),nrow=1,ncol=3)

for(k in (Stages-1):0) {

FromNodes = matrix(Nodes[Nodes[,1] == (k+1)],ncol=3)
NodeNumbers = FromNodes[,3]

for(i in NodeNumbers) {

TotalLeft = FromNodes[FromNodes[,3] == i,2]

j = min(Ns[k+1],TotalLeft)

while(j >= 0) {

Most = min(Zstar,sum(Ns[1:k]))

if(TotalLeft - j <= Most) {
NewNode = c(k,TotalLeft - j)
```

```
if(!(NewNode[1]==0 && NewNode[2] != 0)) {

if(sum(NewNode[1] == Nodes[,1] & NewNode[2] == Nodes[,2]) > 0) {

DupNode = Nodes[NewNode[1] == Nodes[,1] & NewNode[2] == Nodes[,2],3]
Arcs = rbind(Arcs,c(i,DupNode,j))
}

if(sum(NewNode[1] == Nodes[,1] & NewNode[2] == Nodes[,2]) == 0) {

NodeNumber = NodeNumber + 1
Nodes = rbind(Nodes,c(NewNode,NodeNumber))
Arcs = rbind(Arcs,c(i,NodeNumber,j))

}

}
}

j = j - 1

}
}

}
```

```
return(list(Nodes,Arcs[2:nrow(Arcs),]))

}

LongShort = function(Nodes,Arcs,Ns) {

Stages = max(Nodes[,1])

NodesLS = matrix(c(0,0,max(Nodes[,3]),0,0),nrow=1,ncol=5)

for(k in 1:Stages) {

TheseNodes = Nodes[Nodes[,1] == k, 3]

for(i in TheseNodes) {

ThisNode = Nodes[Nodes[,3] == i,]
TheseArcs = matrix(Arcs[Arcs[,1] == i,],ncol=3)
FromNode = NodesLS[NodesLS[,3] == TheseArcs[1,2],]

Contr = TheseArcs[1,3]*(Ns[k] - TheseArcs[1,3])
Long = FromNode[5] + Contr
Short = FromNode[4] + Contr

if(nrow(TheseArcs) > 1) {
```

```
for(j in 2:nrow(TheseArcs)) {

FromNode = NodesLS[NodesLS[,3] == TheseArcs[j,2],]
Contr = TheseArcs[j,3]*(Ns[k] - TheseArcs[j,3])
Long = max(Long,FromNode[5] + Contr)
Short = min(Short,FromNode[4] + Contr)

}

}

NodesLS = rbind(NodesLS,c(ThisNode[1],ThisNode[2],ThisNode[3],
Short,Long))

}

}

return(NodesLS)

}

FinalNetwork = function(ZStar,Ns,OldNodes,OldArcs) {

Stages = max(OldNodes[,1])
NewNodeNum = 1
```

```
PrevNode = OldNodes[OldNodes[,1] == Stages,]
Nodes = matrix(c(PrevNode[1], PrevNode[2], ZStar, NewNodeNum,
PrevNode[3]),nrow=1,ncol=5)
Arcs = matrix(c(0,0,0),nrow=1,ncol=3)
for(k in (Stages):1) {

StageNodes = matrix(Nodes[Nodes[,1] == k,],ncol=5)

for(i in 1:nrow(StageNodes)) {

ArcsFrom = matrix(OldArcs[OldArcs[,1] == StageNodes[i,5]],ncol = 3)

for(j in 1:nrow(ArcsFrom)) {

ThisNode = OldNodes[OldNodes[,3] == ArcsFrom[j,2],]
Min = ThisNode[4]
Max = ThisNode[5]

v = StageNodes[i,3] - ArcsFrom[j,3]*(Ns[k] - ArcsFrom[j,3])

Possible = (Min <= v && v <= Max)

if(Possible) {

NewNode = c(ThisNode[1],ThisNode[2],v)
if(sum(NewNode[1]==Nodes[,1] & NewNode[2] == Nodes[,2] &
```

```
NewNode[3] == Nodes[,3]) > 0){

DupNode = Nodes[NewNode[1] == Nodes[,1] & NewNode[2] == Nodes[,2] &
NewNode[3] == Nodes[,3],4]
Arcs = rbind(Arcs,c(StageNodes[i,4],DupNode,choose(Ns[k],
ArcsFrom[j,3])))
}

if(sum(NewNode[1]==Nodes[,1] & NewNode[2] == Nodes[,2] &
NewNode[3] == Nodes[,3]) == 0){

NewNodeNum = NewNodeNum + 1
Nodes = rbind(Nodes,c(NewNode,NewNodeNum,ThisNode[3]))
Arcs = rbind(Arcs,c(StageNodes[i,4],NewNodeNum,
choose(Ns[k],ArcsFrom[j,3])))

}

}

}

}

}
```

```
return(list(Nodes[,1:4],Arcs[2:nrow(Arcs),]))
}

DropNodes = function(Nodes,Arcs) {



Stages = max(Nodes[,1])

for (i in 1:(Stages-1)) {

StageNodes = matrix(Nodes[Nodes[,1]==i,],ncol=4)
for(j in 1:nrow(StageNodes)) {

DropIt = sum(Arcs[,1] == StageNodes[j,4])

if(DropIt == 0) {

DropNode =  StageNodes[j,4]
Nodes = Nodes[Nodes[,4] != DropNode,]

Arcs = Arcs[Arcs[,2] != DropNode,]

}

}
```

```
}

return(list(Nodes,Arcs))

}

DuplicateNodes = function(Nodes,Arcs) {

NumNodes = nrow(Nodes)
KeepNodes = 1:NumNodes
NumArcs = nrow(Arcs)
Stages = max(Nodes[,1])
TempArcs = cbind(Arcs,1:NumArcs)
Dups = 0
i = 0

ReturnNodes = Nodes[1,]

while(i  <= (Stages-1)) {

StageNodes = Nodes[Nodes[,1] == i,]
NodesInStage = length(StageNodes)/4
if(NodesInStage == 1)
{ReturnNodes = rbind(ReturnNodes,StageNodes)}

if(NodesInStage > 1) {
```

```
while(NodesInStage>1) {



ReturnNodes = rbind(ReturnNodes,StageNodes[1,])
CheckNode = StageNodes[1,]
NodeNum = CheckNode[4]

rmnodes =  StageNodes[2:NodesInStage,1] == CheckNode[1] &
StageNodes[2:NodesInStage,2] == CheckNode[2] &
StageNodes[2:NodesInStage,3] == CheckNode[3]
lose = StageNodes[c(F,rmnodes),4]

StageNodes = StageNodes[!is.element(StageNodes[,4],lose),]

NodesInStage = length(StageNodes)/4

if(length(lose) > 0) {

for (j in lose) {

Arcs[Arcs[,2] == j,2] = NodeNum
Arcs[Arcs[,1] == j,1] = NodeNum

}
```

```
}

if(NodesInStage > 1) {
StageNodes = StageNodes[2:NodesInStage,]
NodesInStage = length(StageNodes)/4
if(NodesInStage == 1)
{ReturnNodes = rbind(ReturnNodes,StageNodes)}

}



}

}

i = i + 1

}

Arcs = Arcs[!duplicated(Arcs),]
return(list(ReturnNodes,Arcs))

}

Traverse2 = function(Nodes,Arcs,Scores) {
```

```
NumNodes = nrow(Nodes)
Stages = max(Nodes[,1])
EndNode = Nodes[Nodes[,1]==Stages,4]

LastRound = matrix(c(EndNode,0,1),nrow=1,ncol=3)



for (i in (Stages-1):0) {

ThisRound = matrix(0,nrow=1,ncol=3)
StageNodes = matrix(Nodes[Nodes[,1] == i,],ncol=4)



for (j in 1:nrow(StageNodes)) {

CurrentNode = StageNodes[j,4]

ArcsTo = matrix(Arcs[Arcs[,2] == CurrentNode,],ncol=3)

OldNodes = matrix(Nodes[Nodes[,4] == ArcsTo[1,1],],ncol=4)

Add = Scores[i+1]*(OldNodes[,2] - StageNodes[j,2])
Mult = ArcsTo[1,3]

LastScores = matrix(LastRound[LastRound[,1] ==
ArcsTo[1,1],],ncol=3)
```

```
ThisRound = rbind(ThisRound,cbind(CurrentNode,Add + LastScores[,2],
Mult*LastScores[,3]))

if(nrow(ArcsTo) > 1) {

for(k in 2:nrow(ArcsTo)) {

OldNodes = matrix(Nodes[Nodes[,4] == ArcsTo[k,1],],ncol=4)

Add = Scores[i+1]*(OldNodes[,2] - StageNodes[j,2])
Mult = ArcsTo[k,3]

LastScores = matrix(LastRound[LastRound[,1] ==
ArcsTo[k,1],],ncol=3)

ThisRound = rbind(ThisRound,cbind(CurrentNode,Add + LastScores[,2],
Mult*LastScores[,3]))

}

}

}

LastRound = matrix(ThisRound[2:nrow(ThisRound),],ncol=3)
```

```
}


DistFact = factor(LastRound[,2])
DistLevels = as.numeric(levels(DistFact))
NumValues = length(DistLevels)
pdf = c(DistLevels[1],sum(LastRound[LastRound[,2] ==
 DistLevels[1],3]))
for (i in 2:NumValues) {
pdf = rbind(pdf,c(DistLevels[i],sum(LastRound[LastRound[,2] ==
DistLevels[i],3])))
}

return(pdf)

}


Samp = matrix(c(0,2,0,0,2,0,0,2,0,0,2,0,0,2,1,1,
2,1,1,2,1,1,1,1,1,1,1),ncol = 3,byrow = T)
Sizes = Samp[,2]
Zs = sum(Samp[,1])
ZNs = sum(Samp[,1]*(Sizes - Samp[,1]))
```

```
Xs = Samp[,3]

First = GetFirstNodes(Zs,Sizes)
FirstNodes = First[[1]]
FirstArcs = First[[2]]

LSNodes = LongShort(FirstNodes,FirstArcs,Sizes)

Test = FinalNetwork(ZNs,Sizes,LSNodes,FirstArcs)

TestNodes = Test[[1]]
TestArcs = Test[[2]]
TestArcs = TestArcs[TestArcs[,1] != 0,]

Next = DropNodes(TestNodes,TestArcs)
NewNodes = Next[[1]]
NewArcs = Next[[2]]

Final = DuplicateNodes(NewNodes,NewArcs)
FinalNodes = Final[[1]]
FinalArcs = Final[[2]]

Dist = Traverse2(FinalNodes,FinalArcs,Xs)

DistFact = factor(Dist[,1])
DistLevels = as.numeric(levels(DistFact))
```

```
NumValues = length(DistLevels)
pdf = c(DistLevels[1],sum(Dist[Dist[,1] == DistLevels[1],2]))
for (i in 2:NumValues) {
pdf = rbind(pdf,c(DistLevels[i],sum(Dist[Dist[,1] ==
DistLevels[i],2])))
}

pvalue = cbind(pdf,pdf[,2]/sum(pdf[,2]))
pvalue
```

A.2 clustexampmod.c - C program for gaining basic information prior to network calculation [8]

```
#include <stdio.h>
#include <stdlib.h>

#include <math.h>
#include <limits.h>
#include "nrutil.h"
#include "numrout.c"
#include "trend.c"

//#define INFILE "C:\\ExactLR\\TestProject\\grafts2.txt"
#define INFILE "C:\\ExactLR\\TestProject\\CurrentSamp.txt"
//#define NUMCLUST 9
#define NUMCLUST 6
#define ALPHA -2.0
#define DELTA 0.1

#define MCSEED 99999999

int main(){

int dose[NUMCLUST],litter[NUMCLUST],i,rowm,obscorr,obsstat;
int sampsz,icount,ncol,numclust,junk;
int dummy;
int x,y,n,cval,ierr,num,yij[NUMCLUST];
```

```
long mcseed=MCSEED;
double pval,mpval,xtilde=0,muhat=0,numer=0,den=0,sigma=0,
xdev=0,geestat;
double alpha,sumxdev=0,rhohat,den2,mrstat;
FILE *fin;

double mrscore(int numclust,int y[],int clsize[],int dose[],
   double alpha,double delta,int rowm,int obscorr,int
obsstat);

rowm=0;
obscorr=0;
obsstat=0;
sampsz=0;
xtilde=0;
ierr=0;
pval=0;
cval=0;
icount=0;
alpha=0.05;
ncol=NUMCLUST;
numclust = NUMCLUST;

fin=fopen(INFILE,"r");
if (fin == NULL) {
   printf("%s\n",INFILE);
```

```
   printf("Error\n");
   system("Pause");
}
if (fin!=NULL){
for(i=0;i<numclust;i++){
num=fscanf(fin,"%d%d%d",&y,&n,&x);
dose[i]=x;
yij[i]=y;
obsstat+=x*y;
obscorr+=y*(n-y);
sampsz+=n;
rowm+=y;
litter[i]=n;
dummy = 0;
xtilde+=(n*x);
}

fclose(fin);
muhat=(float)rowm/(float)sampsz;
xtilde=xtilde/(float)sampsz;
numer=pow(((float)obsstat-(rowm*xtilde)),2);
for(i=0;i<numclust;i++){
sigma=pow(((float)yij[i]-litter[i]*muhat),2);
xdev=pow(((float)dose[i]-xtilde),2);
sumxdev+=xdev;
den+=xdev*sigma;
```

```
}
trstat(&ncol,litter,dose,&sampsz,&rowm,&obsstat,&obscorr,
&alpha,&cval,&pval,&mpval,&icount,&mcseed,&ierr);
}
}
```

A.3 trend.c - C program for calculating network algorithm [8]

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

struct rec_tag{
int pastr;
double pastp;
REC *nextrec;
};

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

struct snd_tag{
int parcorr;
int spl;
int lpl;
double tp;
int numpred;
ARC *arc;
```

```
REC *rec;
};

typedef struct node_tag{
int splcorr;
int lplcorr;
int upper;
int lower;
int numsucc;
SUBND *subnodes;
}NODE;

#define MONTESEED 41169
#define MONTEREPS 10000

void trstat(int *ncol,int table[],int uwt[],int *sampsz,int *rowm,
int *obsstat,int *obscorr,double *alpha,int *cval,double
*pval,
double *mpval,int *icount,long *mcseed,int *ierror);

void trstat(int *ncol,int table[],int uwt[],int *sampsz,int *rowm,
int *obsstat,int *obscorr,double *alpha,int *cval,double
*pval,
double *mpval,int *icount,long *mcseed,int *ierror) {

int i,nnodes,high,lowval;
```

```
int *colm,*cumcol,*wt,*stpos,*minvl,*table2;
double *fact,normcon,mean,var;
REC *currec;
SUBND *cursnode;
NODE *nodes;

FILE *fout,*distout;



void faclog(int sampsz,double *fact);
void calnds(int ncol,int *table,int rowm,int *nnodes,
int *colm,int *cumcol,int *stpos,int *minvl);
NODE *NDvector(long nl, long nh);
void forind(int ncol,int rowm,int nnodes,int sampsz,
int *colm,int *cumcol,double *fact,int *stpos,int *minvl,
NODE *nodes,int *ierror);
void backind(int ncol,int rowm,int nnodes,int sampsz,
int obscorr,int *colm,int *cumcol,int *wt,double *fact,
int *stpos,int *minvl,NODE *nodes,int *ierror);
void printnd(int nnodes,int obscorr,NODE *nodes);
void finalpass(int nnodes,int ncol,int *minvl,int *stpos,
int obsstat,int lowval,int obscorr,NODE *nodes,double *rtail);
double monte(int ncol,int *stpos,NODE *nodes,long *mcseed,
int obscorr,int obsstat);
void free_NDvector(NODE *v, long nl, long nh);
void free_SNvector(SUBND *v, long nl, long nh);
```

```
void free_arc(ARC *arc);
void freerec(REC *rec);

*mpval=0.0;
if (*rowm==0 || *rowm==*sampsz) {
*mpval=1.0;
*pval=1.0;
return;
}
table2=ivector(0,*ncol-1);
colm=ivector(1,*ncol+1);
cumcol=ivector(1,*ncol+1);
wt=ivector(1,*ncol+1);
stpos=ivector(1,*ncol+1);
minvl=ivector(1,*ncol+1);
fact=dvector(1,*sampsz+1);

*ierror=0;
faclog(*sampsz,fact);
wt[1]=0;
for (i=2;i<=*ncol+1;i++){
table2[i-2]=table[*ncol-i+1];
wt[i]=uwt[i-2];
}
calnds(*ncol,table2,*rowm,&nnodes,colm,cumcol,stpos,minvl);
nodes=NDvector(1,nnodes);
```

```
forind(*ncol,*rowm,nnodes,*sampsz,colm,cumcol,fact,stpos,minvl,
nodes,ierror);
for (i=1;i<=nnodes;i++) {
nodes[i].upper=nodes[nnodes-i+1].lplcorr;
nodes[i].lower=nodes[nnodes-i+1].splcorr;
}
calnds(*ncol,table,*rowm,&nnodes,colm,cumcol,stpos,minvl);
fout=fopen("C:\\ExactLR\\TestProject\\clustout.txt","w");
fprintf(fout,"\ni colm_i  cumcol_i  wt_i  stpos_i  minvl_i\n");
for (i=1;i<=*ncol+1;i++)
fprintf(fout,"%d %d %d %d %d %d\n",i,colm[i],cumcol[i],wt[i],
stpos[i],minvl[i]);
fclose(fout);
forind(*ncol,*rowm,nnodes,*sampsz,colm,cumcol,fact,stpos,minvl,
nodes,ierror);

backind(*ncol,*rowm,nnodes,*sampsz,*obscorr,colm,cumcol,
wt,fact,stpos,minvl,nodes,ierror);

*mpval=monte(*ncol,stpos,nodes,mcseed,*obscorr,*obsstat);
lowval = 0;
normcon=nodes[stpos[1]].subnodes[*obscorr].tp;
*pval=(double)0.0;
finalpass(nnodes,*ncol,minvl,stpos,*obsstat,lowval,*obscorr,
nodes,pval);
cursnode=&(nodes[stpos[*ncol+1]].subnodes[0]);
```

```
currec=cursnode->rec;

distout=fopen("C:\\ExactLR\\TestProject\\distn.txt","w");

mean=0.0;
var=0.0;
while(currec != NULL) {


mean+=(currec->pastr)*exp(currec->pastp-normcon);
var+=(currec->pastr)*(currec->pastr)*exp(currec->pastp-normcon);


fprintf(distout,"%6d %20.8f %20.8f\n",currec->pastr,
currec->pastp,exp(currec->pastp-normcon));

if (currec->pastr >= *obsstat)
*pval=*pval+exp(currec->pastp-normcon);
currec=currec->nextrec;
}
var=var-(mean*mean);
for (i=1;i<=nnodes;i++){
high=min(nodes[i].upper,*obscorr);
if (nodes[i].subnodes != NULL) {
for (j=nodes[i].lower;j<=high;j++) {
if (nodes[i].subnodes[j].arc != NULL)
```

```
free_arc(nodes[i].subnodes[j].arc);
}
free_SNvector(nodes[i].subnodes,nodes[i].lower,high);
}
}
*/


fclose(distout);
cursnode=&(nodes[stpos[*ncol+1]].subnodes[0]);
currec=cursnode->rec;
freerec(currec);
high=min(nodes[stpos[*ncol+1]].upper,*obscorr);

free_SNvector(nodes[stpos[*ncol+1]].subnodes,
nodes[stpos[*ncol+1]].lower,high);
free_NDvector(nodes,1,nnodes);
free_ivector(table2,0,*ncol-1);
free_ivector(colm,1,*ncol+1);
free_ivector(cumcol,1,*ncol+1);
free_ivector(wt,1,*ncol+1);
free_ivector(stpos,1,*ncol+1);
free_ivector(minvl,1,*ncol+1);
free_dvector(fact,1,*sampsz+1);
return;
}
```

```
double monte(int ncol,int *stpos,NODE *nodes,long *mcseed,
int obscorr,int obsstat){

int i,rej,fathomed,parstat;
double cumprob,prob;
SUBND *curnode,*succ;
ARC *carc;

double ran1(long *idum);

rej=0;
for (i=1;i<=MONTEREPS;i++){
parstat=0;
fathomed=0;
curnode=&(nodes[stpos[1]].subnodes[obscorr]);
while (!fathomed){
prob=ran1(mcseed);
carc=curnode->arc;
succ=carc->child;
cumprob=exp((succ->tp)+(carc->pr)-(curnode->tp));

while (prob>cumprob){
if (carc->nextarc==NULL){
printf("\n\nERROR IN MONTE:  ran out of successor nodes!\n");
return(-1.0);
```

```
}
carc=carc->nextarc;
succ=carc->child;
cumprob+=exp(succ->tp+carc->pr-curnode->tp);
}
parstat+=carc->arc;
if ((parstat+succ->spl)>=obsstat){
rej++;
fathomed=1;
}
if ((parstat+succ->lpl)<obsstat)
fathomed=1;
curnode=succ;
}
}
return((double)rej/(double)MONTEREPS);
}

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

int pos,hlim,llim,r,stage,high,k,j;
double npr;
SUBND *cursnode,*succ;
ARC *carc;
```

```
REC *newrec,*crec,*curr,*nxt;

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

newrec=crerec(0,(double)0.0);
nodes[stpos[1]].subnodes[obscorr].rec=newrec;
for (stage=1;stage<=ncol;stage++){
llim=minvl[stage];
if (stage==1) hlim=llim;
else hlim=llim+stpos[stage-1]-stpos[stage]-1;
for (pos=stpos[stage];pos<=stpos[stage]+hlim-llim;pos++){
if (nodes[pos].subnodes == NULL)
continue;
high=min(nodes[pos].upper,obscorr);
for (k=nodes[pos].lower;k<=high;k++){
cursnode=&(nodes[pos].subnodes[k]);
if (cursnode->parcorr<0)
continue;
carc=cursnode->arc;
while (carc != NULL) {
succ=carc->child;
crec=cursnode->rec;
```

```
while (crec != NULL) {
r=crec->pastr+carc->arc;
if (r + succ->lpl >= lowval) {
if
(crec->pastp<0.0000001)
   npr=carc->pr;
else
npr=crec->pastp+carc->pr;
newrec=crerec(r,npr);
if (succ->rec == NULL)
   succ->rec=newrec;
else if (newrec->pastr <

   succ->rec->pastr) {

newrec->nextrec=succ->rec;

succ->rec=newrec;
}
else {
   curr=succ->rec;
   nxt=curr->nextrec;
   while (nxt != NULL){
      if (r ==
curr->pastr ||
 (nxt!=NULL &&
```

```
(r<nxt->pastr))) break;
      curr=nxt;
      nxt=curr->nextrec;
   }
   if (r ==
curr->pastr){

curr->pastp=addlog(curr->pastp,
      newrec->pastp);
      free(newrec);
   }
   else {

newrec->nextrec=nxt;

curr->nextrec=newrec;
   }
       }
       }
       crec=crec->nextrec;
}
carc=carc->nextarc;
}
if (cursnode->rec != NULL)
   freerec(cursnode->rec);
}
```

```
if (nodes[pos].subnodes != NULL) {
   for (j=nodes[pos].lower;j<=high;j++) {
      if (nodes[pos].subnodes[j].arc != NULL)
 free_arc(nodes[pos].subnodes[j].arc);
   }
}

free_SNvector(nodes[pos].subnodes,nodes[pos].lower,high);
}
}
}

REC * crerec(int r,double pr) {

REC *record;

record=(REC *)malloc(sizeof(REC));
record->pastr=r;
record->pastp=pr;
record->nextrec=NULL;
return(record);
}

void freerec(REC *rec){

void freerec(REC *rec);
```

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

void free_arc(ARC *arc) {

void free_arc(ARC *arc);

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

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

int llim,hlim,k,j,npos,parm,llm,hlm,ill,ihh,count;
int num,ipl,remcorr,ipl1,nodind,ref,high,up;
double prarc,newprob;
SUBND *cursnode;
ARC *newarc,*carc,*narc;
```

```
void lpsptp(int sampsz,int parm,int stage,int ncol,int rowm,
int nnodes,int *colm,int *cumcol,int *stpos,
int *minvl,NODE *nodes,int *spl,int *lpl);
SUBND *SNvector(long nl, long nh);
void init(int lower,int upper,SUBND *subnodes);
SUBND *crenode(int remcorr,int lp,int sp,int ipl,double prarc);
double prbar0(int coltot,int numsucc,double *fact);
double addlog(double num1,double num2);
void dropnd(SUBND *cursnode);

npos=stpos[ncol+1];
nodes[npos].subnodes=SNvector(0,0);
nodes[npos].subnodes[0].parcorr=obscorr;
nodes[npos].subnodes[0].lpl=0;
nodes[npos].subnodes[0].spl=0;
nodes[npos].subnodes[0].numpred=0;
nodes[npos].subnodes[0].tp=0.0;
nodes[npos].subnodes[0].arc=NULL;
nodes[npos].subnodes[0].rec=NULL;
for (j=ncol+1;j>1;j--){
llim=minvl[j];
hlim=llim+stpos[j-1]-stpos[j]-1;
for (parm=llim;parm<=hlim;parm++,npos++){
if (nodes[npos].subnodes == NULL) continue;
llm=max(0,parm-colm[j]);
```

```
hlm=min(parm,cumcol[j-1]);
ill=stpos[j-1]+llm-minvl[j-1];
ihh=ill+hlm-llm;
count=parm-llm;
up=min(nodes[npos].upper,obscorr);
for (nodind=nodes[npos].lower;nodind<=up;nodind++){
cursnode=&(nodes[npos].subnodes[nodind]);
if ((cursnode->parcorr)<0)
continue;
num=count;
for (k=ill;k<=ihh;k++,num--){
ipl=num*(colm[j]-num);
remcorr=cursnode->parcorr - ipl;
if (nodes[k].lplcorr<remcorr ||
nodes[k].splcorr>remcorr)
continue;
(cursnode->numpred)++;
if (nodes[k].subnodes==NULL) {

high=min(nodes[k].upper,obscorr);

nodes[k].subnodes=SNvector(nodes[k].lower,high);

init(nodes[k].lower,high,nodes[k].subnodes);
}
newarc=(ARC *)malloc(sizeof(ARC));
```

```
ipl1=wt[j]*num;
prarc=prbar0(colm[j],num,fact);
newarc->child=cursnode;
newarc->arc=ipl1;
newarc->pr=prarc;
newarc->nextarc=NULL;
ref=obscorr-remcorr;
if ((nodes[k].subnodes[ref].lpl)<0) {

nodes[k].subnodes[ref].parcorr=remcorr;

nodes[k].subnodes[ref].lpl=(cursnode->lpl)+
ipl1;

nodes[k].subnodes[ref].spl=(cursnode->spl)+
ipl1;

nodes[k].subnodes[ref].tp=(cursnode->tp)+
prarc;

nodes[k].subnodes[ref].arc=newarc;
}
else {
if ((cursnode->lpl)+ipl1 >

nodes[k].subnodes[ref].lpl)
```

```
nodes[k].subnodes[ref].lpl =
cursnode->lpl +
ipl1;
if ((cursnode->spl)+ipl1 <

nodes[k].subnodes[ref].spl)

nodes[k].subnodes[ref].spl =
cursnode->spl +
ipl1;
newprob=prarc+cursnode->tp;
nodes[k].subnodes[ref].tp=

addlog(nodes[k].subnodes[ref].tp,newprob);
if (nodes[k].subnodes[ref].arc
== NULL)

nodes[k].subnodes[ref].arc=newarc;
else {

carc=nodes[k].subnodes[ref].arc;
narc=carc->nextarc;
while (narc != NULL) {

carc=carc->nextarc;
```

```
narc=carc->nextarc;
}
carc->nextarc=newarc;
}
}
}
if ((cursnode->numpred)==0)
dropnd(cursnode);
}
}
}
}

void dropnd(SUBND *cursnode){

SUBND *cnode;
ARC *carc,*narc;

cursnode->lpl=-1;
cursnode->spl=-1;
cursnode->tp=-1;
cursnode->parcorr=-1;
carc=cursnode->arc;
while (carc != NULL) {
cnode=carc->child;
```

```
(cnode->numpred)--;
if ((cnode->numpred)==0)
dropnd(cnode);
narc=carc->nextarc;
free(carc);
carc=narc;
}
cursnode->arc=NULL;
return;
}

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

int k;

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

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

SUBND *newnode;

newnode=(SUBND *)malloc(sizeof(SUBND));
newnode->parcorr=remcorr;
newnode->lpl=lp+ipl;
newnode->spl=sp+ipl;
newnode->tp=prarc;
newnode->numpred=0;
return(newnode);
}

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

int llim,hlim,k,j,npos,spl,lpl;

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


npos=stpos[1];
```

```
nodes[npos].splcorr=0;
nodes[npos].lplcorr=0;
nodes[npos].subnodes=NULL;
for (j=2;j<=ncol+1;j++){
npos=stpos[j];
llim=minvl[j];
hlim=llim+stpos[j-1]-stpos[j]-1;
for (k=llim;k<=hlim;k++){
corrlpsp(sampsz,k,j,ncol,rowm,nnodes,colm,cumcol,
stpos,minvl,nodes,&spl,&lpl);
nodes[npos].splcorr=spl;
nodes[npos].lplcorr=lpl;
nodes[npos].numsucc=0;
nodes[npos].subnodes=NULL;
npos++;
}
}
}

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

int llm,hlm,ill,ihh,isp,ilp,ipl,isp2,ilp2,count,pos;

llm=max(0,parm-colm[stage]);
```

```
hlm=min(parm,cumcol[stage-1]);
ill=stpos[stage-1]+llm-minvl[stage-1];
ihh=ill+hlm-llm;

count=parm-llm;
ipl=count*(colm[stage]-count);
isp=nodes[ill].splcorr+ipl;
ilp=nodes[ill].lplcorr+ipl;
if (ill < ihh) {
for (pos=ill+1;pos<=ihh;pos++){
count--;
ipl=count*(colm[stage]-count);
isp2=nodes[pos].splcorr+ipl;
ilp2=nodes[pos].lplcorr+ipl;
if (isp2<isp) isp=isp2;
if (ilp2>ilp) ilp=ilp2;
}
}
*spl=isp;
*lpl=ilp;
return;
}

void flip(int ncol,int *colm,int *cumcol){

int i,tmp1,tmp2;
```

```
for (i=1;i<=(ncol/2);i++){
tmp1=colm[i];
tmp2=cumcol[i];
colm[i]=colm[ncol-i+1];
cumcol[i]=cumcol[ncol-i+1];
colm[ncol-i+1]=tmp1;
cumcol[ncol-i+1]=tmp2;
}
return;
}

void calnds(int ncol,int *table,int rowm,int *nnodes,int *colm,
int *cumcol,int *stpos,int *minvl){

int i,stage,iconst,llim,hlim,npos;

colm[1]=0;
cumcol[1]=0;

for (i=2;i<=ncol+1;i++){
colm[i]=table[i-2];
cumcol[i]=cumcol[i-1]+colm[i];
}
iconst=rowm-cumcol[ncol+1];
npos=2;
```

```
minvl[ncol+1]=rowm;
stpos[ncol+1]=1;

for (stage=ncol-1;stage>=0;stage--){
llim=max(0,iconst+cumcol[stage+1]);
hlim=min(rowm,cumcol[stage+1]);
stpos[stage+1]=npos;
minvl[stage+1]=llim;
npos=npos+(hlim-llim+1);
    }
*nnodes=npos-1;
return;
}

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

double logprob;

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

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

double total,t1,t2,tmax;

tmax=dmax(num1,num2);
t1=dmax(num1-tmax,-80.0);
t2=dmax(num2-tmax,-80.0);
total=tmax+log(exp(t1)+exp(t2));
return(total);
}

void faclog(int sampsz,double *fact){

int i;

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

void printnd(int nnodes,int obscorr,NODE *nodes){

int i,k,high;
SUBND *cursnode;
```

```
ARC *carc;
FILE *fout;

fout=fopen("clust.out","a");
fprintf(fout,"\n\n nnodes=%d\n\nNode SPL LPL LOW UPP\n",nnodes,
   "--------------------");
for (i=1;i<=nnodes;i++){
fprintf(fout,"\n%d   %d   %d   %d   %d",i,nodes[i].splcorr,
nodes[i].lplcorr,nodes[i].lower,nodes[i].upper);
if (nodes[i].subnodes != NULL) {
high=min(nodes[i].upper,obscorr);
for (k=nodes[i].lower;k<=high;k++){
if (nodes[i].subnodes[k].parcorr<0) continue;
cursnode=&(nodes[i].subnodes[k]);
fprintf(fout,"\n%d   %d   %d %5.3f",cursnode->parcorr,

cursnode->spl,cursnode->lpl,cursnode->tp);
carc=cursnode->arc;
while (carc != NULL){
  fprintf(fout,"\n       arc:  %d %5.3f",carc->arc,carc->pr);
  fprintf(fout,"\n       child:  %d %d %d %5.3f",
  carc->child->parcorr,carc->child->spl,
  carc->child->lpl,carc->child->tp);
  carc=carc->nextarc;
}
}
```

```
}
}
fprintf(fout,"\n\n\n");
fclose(fout);
}

#define NR_END 1
#define FREE_ARG char*

NODE *NDvector(long nl, long nh)
{
NODE *v;

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

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

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

```
void free_NDvector(NODE *v, long nl, long nh)
{
free((FREE_ARG) (v+nl-NR_END));
}

void free_SNvector(SUBND *v, long nl, long nh)
{
free((FREE_ARG) (v+nl-NR_END));
}
```

A.4 simc.txt - R program for generating random samples from a quadratic exponential distribution and reading the results of the network algorithm for parameter estimation

```
dyn.load("C:\\ExactLR\\TestProject\\trendtest2.dll")
source("Z:\\School\\Exact LR\\Program\\
unrestricted log likelihood.txt")
source("C:\\ExactLR\\CIFalse.txt")
source("C:\\ExactLR\\MUE.txt")

library(geepack)
alpha = -0.4
beta = 0.7

delta = 0.0

N = 20
x = c(0,1)
split = N/2
X = c(rep(0,split),rep(1,split))

numsim = 1000
thetas = rep(0,numsim)
GEEThetas = rep(0,numsim)
GEEStdErr = rep(0,numsim)
betau = rep(0,numsim)
codetest = rep(0,numsim)
```

```
CIs = matrix(0,ncol=2,nrow=numsim)

for (l in 1:numsim) {

print(l)

n = rep(30,N)
Z = rep(-1,N)

AllZero = 1;
while (AllZero == 1) {
for(i in 1:N) {
Values1 = rep(0,n[i]+1)
Values2 = rep(0,n[i]+1)

for (z in 0:n[i]) {

Values1[z+1] = choose(n[i],z)*exp(alpha*z + beta*x[1]*
z- delta*z*(n[i]-z))
Values2[z+1] = choose(n[i],z)*exp(alpha*z + beta*x[2]*
z- delta*z*(n[i]-z))

}

Values1 = Values1/sum(Values1)
```

```
Values2 = Values2/sum(Values2)

if(i <= split) {

take = runif(1)
stop = 0
j = 1
cum = Values1[1]
while (stop == 0) {

if (take <= cum) {

Z[i] = j - 1
stop = 1

}

j = j + 1
cum = cum + Values1[j]
}

}

if(i >= (split+1)) {

take = runif(1)
```

```
stop = 0
j = 1
cum = Values2[1]
while (stop == 0) {

if (take <= cum) {

Z[i] = j - 1
stop = 1

}

j = j + 1
cum = cum + Values2[j]
}

}

}

AllZero = (sum(Z[1:split]) == 0) || (sum(Z[(split+1):N]) == 0)
}

if (sum(Z) == 0) stop("All Zero")

TestSamp = cbind(Z,n,X)
```

```
write.table(TestSamp,"C:\\ExactLR\\TestProject\\CurrentSamp.txt",
row.names = F,col.names=F)
empty = .C("main")
pvalue = read.table("C:\\ExactLR\\TestProject\\distn.txt")

theta = -5
thetanew = 0.02
C = exp(pvalue[,2])
u = pvalue[,1]
t = as.numeric(TestSamp[,1]%*%X)
stop = 0
numsteps = 0

if (t == max(pvalue[,1])||sum(pvalue[pvalue[,1]>t,3])<0.0000001) {

stop = 1

thetanew = MUEFalse(pvalue,t)

}

if (t == min(pvalue[,1])) {

stop = 1
#thetanew = -Inf
```

```
thetanew = MUEFalse(pvalue,t)

}

while ( stop == 0 && numsteps < 100) {

theta = thetanew
H = t - sum(C*u*exp(u*theta))/sum(C*exp(u*theta))
J = sum(C*u^2*exp(u*theta))/sum(C*exp(u*theta)) -
(sum(C*u*exp(u*theta))/sum(C*exp(u*theta)))^2
thetanew = theta + H/J
stop = ifelse(abs(theta-thetanew) > 0.0000001,0,1)
numsteps = numsteps + 1


}


thetas[l] = thetanew

CI = CIFalse(pvalue,t,.05)
CIs[l,1] = CI[1]
CIs[l,2] = CI[2]
unrest = nlm(f,c(alpha,beta,delta),obs=TestSamp[,1],
size=TestSamp[,2],score=TestSamp[,3])
```

```
betau[l] = unrest$estimate[2]

codetest[l] = unrest$code


write.table(cbind(thetanew,CI[1],CI[2],unrest$estimate[2]),

"C:\\ExactLR\\FinalSim\\interim.txt",

row.names = F,col.names = F,append = T)


}
```

A.5 MUEFalse.txt - R program for calculating median unbiased estimate

```
MUEFalse = function(pvalue,t) {

stop = 0
numsteps = 0

C = exp(pvalue[,2])
u = pvalue[,1]

FTheta = matrix(pvalue[u <= t,],ncol=3)

FC = exp(FTheta[,2])
Fu = FTheta[,1]

GTheta = matrix(pvalue[u >= t,],ncol=3)

GC = exp(GTheta[,2])
Gu = GTheta[,1]

MUE = 0
a = -1
b= 1

### Lower Bound is -Inf ###

if(t == max(u)) {
```

```
stop = 1

}

if(stop == 0) {
Fofa = sum(FC*exp(Fu*a))/sum(C*exp(u*a)) - .5
Fofb = sum(FC*exp(Fu*b))/sum(C*exp(u*b)) - .5

while(sign(Fofa) == sign(Fofb)) {

b = abs(b) + 1
Fofb = sum(FC*exp(Fu*b))/sum(C*exp(u*b)) - .5

if (sign(Fofa) == sign(Fofb)) {

b = -b
Fofb = sum(FC*exp(Fu*b))/sum(C*exp(u*b)) - .5

}

}

ck = (Fofb*a - Fofa*b)/(Fofb - Fofa)
Fofck = sum(FC*exp(Fu*ck))/sum(C*exp(u*ck)) - .5
```

```
if(sign(Fofa) == sign(Fofck)) {

while (stop == 0 && numsteps < 1000) {

a = ck
Fofa = Fofck
ck = (Fofb*a - Fofa*b)/(Fofb - Fofa)
Fofck = sum(FC*exp(Fu*ck))/sum(C*exp(u*ck)) - .5

numsteps = numsteps + 1
stop = ifelse(is.na(ck),1,(abs(a-ck) < 0.0000001))

}

MUE = ck

}

if(sign(Fofb) == sign(Fofck)) {

while (stop == 0 && numsteps < 1000) {

b = ck
Fofb = Fofck

ck = (Fofb*a - Fofa*b)/(Fofb - Fofa)
```

```
Fofck = sum(FC*exp(Fu*ck))/sum(C*exp(u*ck)) - .5
numsteps = numsteps + 1
stop = ifelse(is.na(ck),1,(abs(b-ck) < 0.0000001))
}

MUE = ck
}
}

### Upper Bound is Inf  #####

theta = 0
theta1 = -.5

stop = 0
numsteps = 0

a = -1
b= 1

if(t == min(u)) {

stop = 1

}
```

```
if (stop == 0) {

Gofa = sum(GC*exp(Gu*a))/sum(C*exp(u*a)) - .5
Gofb = sum(GC*exp(Gu*b))/sum(C*exp(u*b)) - .5



while(sign(Gofa) == sign(Gofb)) {

b = abs(b) + 1
Gofb = sum(GC*exp(Gu*b))/sum(C*exp(u*b)) - .5

if (sign(Gofa) == sign(Gofb)) {

b = -b
Gofb = sum(GC*exp(Gu*b))/sum(C*exp(u*b)) - .5

}

}

ck = (Gofb*a - Gofa*b)/(Gofb - Gofa)
Gofck = sum(GC*exp(Gu*ck))/sum(C*exp(u*ck)) - .5

if(sign(Gofa) == sign(Gofck)) {

while (stop == 0 && numsteps < 1000) {
```

```
a = ck
Gofa = Gofck
ck = (Gofb*a - Gofa*b)/(Gofb - Gofa)
Gofck = sum(GC*exp(Gu*ck))/sum(C*exp(u*ck)) - .5

numsteps = numsteps + 1
stop = ifelse(is.na(ck),1,(abs(a-ck) < 0.0000001))

}

MUE = ck

}

if(sign(Gofb) == sign(Gofck)) {

while (stop == 0 && numsteps < 1000) {

b = ck
Gofb = Gofck

ck = (Gofb*a - Gofa*b)/(Gofb - Gofa)
Gofck = sum(GC*exp(Gu*ck))/sum(C*exp(u*ck)) - .5

numsteps = numsteps + 1
```

```
stop = ifelse(is.na(ck),1,(abs(b-ck) < 0.0000001))
}

MUE = ck
}
}

return(MUE)

}
```

A.6 CIFalse.txt - R program for calculating $(1-\alpha)\%$ confidence intervals

```
CIFalse = function(pvalue,t,alpha0) {

stop = 0
numsteps = 0



if (sum(pvalue[pvalue[,1] > t,3]) < 0.0000001) {
pvalue = pvalue[pvalue[,1]<=t,]
}

if (sum(pvalue[pvalue[,1] < t,3]) < 0.0000001) {
pvalue = pvalue[pvalue[,1]>=t,]
}

C = exp(pvalue[,2])
u = pvalue[,1]

FTheta = matrix(as.matrix(pvalue[u <= t,]),ncol=3)

FC = exp(FTheta[,2])
Fu = FTheta[,1]

GTheta = matrix(as.matrix(pvalue[u >= t,]),ncol=3)

GC = exp(GTheta[,2])
```

```
Gu = GTheta[,1]



### Upper Bound ####


a = -1

b= .5


if(t == max(u)) {


stop = 1

upper = Inf


}


if(stop == 0) {

Fofa = sum(FC*exp(Fu*a))/sum(C*exp(u*a)) - alpha0/2

Fofb = sum(FC*exp(Fu*b))/sum(C*exp(u*b)) - alpha0/2


while(sign(Fofa) == sign(Fofb)) {


b = abs(b) + .001

Fofb = sum(FC*exp(Fu*b))/sum(C*exp(u*b)) - alpha0/2


if (sign(Fofa) == sign(Fofb)) {
```

```
b = -b
Fofb = sum(FC*exp(Fu*b))/sum(C*exp(u*b)) - alpha0/2
if(is.nan(Fofb))
{b = -b}

}

}

ck = (Fofb*a - Fofa*b)/(Fofb - Fofa)
Fofck = sum(FC*exp(Fu*ck))/sum(C*exp(u*ck)) - alpha0/2

while (stop == 0 && numsteps < 1000) {

if(sign(Fofa) == sign(Fofck)) {

a = ck
Fofa = Fofck
ck = (Fofb*a - Fofa*b)/(Fofb - Fofa)
Fofck = sum(FC*exp(Fu*ck))/sum(C*exp(u*ck)) - alpha0/2

numsteps = numsteps + 1
stop = ifelse(is.na(ck),1,(abs(a-ck) < 0.0000000001))

#if(sign(Fofa) != sign(Fofck)) {
```

```
# ck = (a + ck)/2
# stop = T

#}

}

#upper = ck

#}

if(sign(Fofb) == sign(Fofck) && stop == F) {

# while (stop == 0 && numsteps < 1000) {

b = ck
Fofb = Fofck

ck = (Fofb*a - Fofa*b)/(Fofb - Fofa)
Fofck = sum(FC*exp(Fu*ck))/sum(C*exp(u*ck)) - alpha0/2
numsteps = numsteps + 1
stop = ifelse(is.na(ck),1,(abs(b-ck) < 0.0000000001))
#}

#upper = ck
}
```

```
}
upper = ck
}

### Lower Bound  #####

theta = 0
theta1 = -.5

stop = 0
numsteps = 0

a = -1
b= .05

if(t == min(u)) {

stop = 1
lower = -Inf

}

if (stop == 0) {

Gofa = sum(GC*exp(Gu*a))/sum(C*exp(u*a)) - alpha0/2
```

```
Gofb = sum(GC*exp(Gu*b))/sum(C*exp(u*b)) - alpha0/2



while(sign(Gofa) == sign(Gofb)) {


b = abs(b) + .01

Gofb = sum(GC*exp(Gu*b))/sum(C*exp(u*b)) - alpha0/2


if (sign(Gofa) == sign(Gofb)) {


b = -b

Gofb = sum(GC*exp(Gu*b))/sum(C*exp(u*b)) - alpha0/2

if(is.nan(Gofb))

{b = -b}


}


}


ck = (Gofb*a - Gofa*b)/(Gofb - Gofa)

Gofck = sum(GC*exp(Gu*ck))/sum(C*exp(u*ck)) - alpha0/2


while (stop == 0 && numsteps < 1000) {


if(sign(Gofa) == sign(Gofck)) {
```

```
a = ck
Gofa = Gofck
ck = (Gofb*a - Gofa*b)/(Gofb - Gofa)
Gofck = sum(GC*exp(Gu*ck))/sum(C*exp(u*ck)) - alpha0/2

numsteps = numsteps + 1
stop = ifelse(is.na(ck),1,(abs(a-ck) < 0.0000000001))

}

#lower = ck

#}

if(sign(Gofb) == sign(Gofck)) {

#while (stop == 0 && numsteps < 1000) {

b = ck
Gofb = Gofck

ck = (Gofb*a - Gofa*b)/(Gofb - Gofa)
Gofck = sum(GC*exp(Gu*ck))/sum(C*exp(u*ck)) - alpha0/2

numsteps = numsteps + 1
stop = ifelse(is.na(ck),1,(abs(b-ck) < 0.0000000001))
```

```
}

#lower = ck
}
lower = ck
}

return(c(lower,upper))

}
```

APPENDIX B

PROGRAMS FOR MARKOV CHAIN MONTE CARLO ESTIMATION

B.1 readsim.cpp - C program for MCMC estimation of the distribution of the trend parameter

```
#include <iostream.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>

#define TABLES "C:\\ExactLR\\MCMC\\C code\\MCTables.txt"
#define STAT "C:\\ExactLR\\MCMC\\C code\\stat.txt"
#define DIST "C:\\ExactLR\\Sim2 Stats\\mcmcdist.txt"
#define INFILE "C:\\ExactLR\\Sim2 Stats\\samples1.txt"
#define ps "C:\\ExactLR\\Sim2 Stats\\pvalues.txt"
#define goodtables "C:\\ExactLR\\Sim2 Stats\\tables.txt"
#define times "C:\\ExactLR\\Sim2 Stats\\mcmctimes.txt"


#define IM1 2147483563
#define IM2 2147483399
#define AM (1.0/IM1)
#define IMM1 (IM1-1)
#define IA1 40014
#define IA2 40692
#define IQ1 53668
#define IQ2 52774
#define IR1 12211
```

```
#define IR2 3791
#define NTAB 32
#define NDIV (1 + IMM1/NTAB)
#define EPS 1.2e-7
#define RNMX (1.0-EPS)

#define numsim 1
#define numclust 200
#define numsteps 100000
#define keepers numsteps/50 + 1
#define burn 50000

double pvalue(int max, int observed, double gamma,
int *x, int itter);
void gettable(int *newtable, int *current, int *n,
int *last);
float ran2(long *idum);

int gensets[keepers][numclust];
long idum = -1999997;

int main() {

int i, j, k, neg, count, corr, sampcorr, sample, teststat, max;
int a, b, c, samp;
int t;
```

```
int current[numclust];
int next[numclust];
int last[numclust];
int poss[numclust];
int stats[numclust];
int n[numclust];
int z[numclust];
int x[numclust];
int *curr = current;
int *nxt = next;

int change[3] = {0,0,0};
FILE *fp;
FILE *fin;
FILE *fppval;
FILE *ftab;
FILE *fdist;
FILE *ftime;

float move, flip, duration;
double changeposs, p;
char filename[20];
time_t start, finish;

fp = fopen(STAT,"w");
fclose(fp);
```

```
fppval = fopen(ps,"w");
fclose(fppval);
fin=fopen(INFILE,"r");
fdist = fopen(DIST,"w");
fclose(fdist);
ftime = fopen(times,"w");
fclose(ftime);

for(samp = 1; samp <= 100; samp ++){
printf("Sample # %d\n",samp);
start = time(0);
t = 0;
corr = 0;
srand (time(0));
max = 0;

for (i = 0; i < numclust; i++) {

   fscanf(fin,"%d%d%d",&b,&c,&a);
   x[i] = a;
   z[i] = b;
   n[i] = c;
   printf("%d %d %d\n",z[i], n[i], x[i]);
   current[i] = z[i];
   gensets[1][i] = z[i];
   t += z[i]*x[i];
```

```
   corr += z[i]*(n[i]-z[i]);
}

stats[0] = t;
gensets[0] = z;
last = current;

j = 1;
printf("Pre Burn\n");
while (j < burn ) {

  next = current;
  gettable(&next[0], &current[0], &n[0], &last[0]);
  current = next;

  sampcorr = 0;
  teststat = 0;

  for (i = 0; i < numclust; i++){
 sampcorr += current[i]*(n[i] - current[i]);
 teststat += current[i]*x[i];
  }

  if(sampcorr == corr) {
 printf("In the loop %d\n", j);
```

```
j++;
last = current;

  }

}
printf("Post Burn\n");

sample = 1;
count = 1;



while (sample < keepers ) {

  next = current;
  gettable(&next[0], &current[0], &n[0], &last[0]);
  current = next;

  sampcorr = 0;
  teststat = 0;

  for (i = 0; i < numclust; i++){
 sampcorr += current[i]*(n[i] - current[i]);
 teststat += current[i]*x[i];
  }
  if(sampcorr == corr) {
```

```
count++;
last = current;

if (count == 50) {

gensets[sample] = current;
if(teststat > max) max = teststat;
sample++;
fp = fopen(STAT,"a");
fprintf(fp, "%d \n",teststat);
fclose(fp);
printf("In the loop %d\n", sample);
count = 1;

}
  }

}

p = pvalue(max, t, 0.0, &x[0],samp);
finish = time(0);
duration = difftime(finish,start);

fppval = fopen(ps,"a");
fprintf(fppval,"%lf\n",p);
```

```
fclose(fppval);

ftime = fopen(times,"a");
fprintf(ftime,"%d %f\n",samp, duration);
fclose(ftime);

printf("The pvalue is %lf\n",p);

}
 system("PAUSE");



 return 0;
}



void gettable(int *newtable, int *current, int *n, int *last)
{
int neg, j, k, two, three, same;
int change[3];
int poss[numclust];

double changeposs, flip, move, split, alpha, u;

two = (numclust*(numclust-1))/2;
three = (numclust*(numclust-1)*(numclust-2))/6;
```

```
split = 0.1;
neg = 1;

while (neg) {

changeposs = ran2(&idum)*(numclust);
change[0] = (int) changeposs;

changeposs = ran2(&idum)*(numclust);
change[1] = (int)changeposs;

while (change[1] == change[0]){

  changeposs = ran2(&idum)*(numclust);
  change[1] = (int) changeposs;

}

changeposs = ran2(&idum)*(numclust);
change[2] = (int) changeposs;

while (change[2] == change[0] || change[2] == change[1]){

  changeposs = ran2(&idum)*(numclust);
  change[2] = (int) changeposs;
```

```
}

move = ran2(&idum);
flip = ran2(&idum);

if (flip < 0.5) {

  if (move < split) {
neg = (*(newtable + change[0]) + 1 > *(n + change[0]) ||
  *(newtable + change[1]) - 1 < 0);

if (!neg) {
   *(newtable + change[0]) = *(newtable + change[0]) + 1;
   *(newtable + change[1]) = *(newtable + change[1]) - 1;

   alpha = (double)(*(current + change[1]))*
 (double)(*(n + change[0])-*(current+change[0]))/
 (double)(*(n + change[1])-*(current + change[1])+1)/
 (double)(*(current + change[0])+1);
}
  }
  else{
neg = (*(newtable + change[0]) + 1 > *(n + change[0]) ||
  *(newtable + change[1]) - 2 < 0 ||
  *(newtable + change[2]) + 1 > *(n + change[2]));
```

```
if (!neg) {
   *(newtable + change[0]) = *(newtable + change[0]) + 1;
   *(newtable + change[1]) = *(newtable + change[1]) - 2;
   *(newtable + change[2]) = *(newtable + change[2]) + 1;

   alpha = (double)(*(current + change[1]))*
 (double)(*(current + change[1])-1)*
 (double)(*(n + change[0])-*(current + change[0]))*
 (double)(*(n + change[2])-*(current + change[2]))/
 (double)(*(current + change[0])+1)/
 (double)(*(current + change[2])+1)/
 (double)(*(n + change[1])-*(current + change[1])+1)/
 (double)(*(n + change[1])-*(current + change[1])+2);
}
  }
}

else {

  if (move < split) {
neg = (*(newtable + change[0]) - 1 < 0 ||
  *(newtable + change[1]) + 1 > *(n + change[1]));

if (!neg) {
   *(newtable + change[0]) = *(newtable + change[0]) - 1;
   *(newtable + change[1]) = *(newtable + change[1]) + 1;
```

```
   alpha = (double)(*(current + change[0]))/
 (double)(*(n + change[0])-*(current+change[0])+1)*
 (double)(*(n + change[1])-*(current + change[1]))/
 (double)(*(current + change[1])+1);
}
  }
  else{

neg = (*(newtable+change[0]) - 1 < 0 ||
  *(newtable+change[1]) + 2 > *(n + change[1]) ||
  *(newtable+change[2]) - 1 < 0);

if (!neg) {
   *(newtable+change[0]) = *(newtable+change[0]) - 1;
   *(newtable+change[1]) = *(newtable+change[1]) + 2;
   *(newtable+change[2]) = *(newtable+change[2]) - 1;

   alpha = (double)(*(current+change[0]))*
 (double)(*(current+change[2]))*
 (double)(*(n+change[1])-*(current+change[1]))*
 (double)(*(n+change[1])-*(current+change[1])-1)/
 (double)(*(current+change[1])+2)/
 (double)(*(current+change[1])+1)/
 (double)(*(n+change[0])-*(current+change[0])+1)/
 (double)(*(n+change[2])-*(current+change[2])+1);
```

```
}
  }
}

   }



u = ran2(&idum);



if (alpha < u){
for (j = 0; j <numclust; j++)
*(newtable+j) = *(current + j);
}
}



double pvalue(int max, int observed, double gamma, int *x,
int itter)
{
int i, j, k, currentstat;
int teststat[keepers], dist[max+1], current[numclust];
int unitables[keepers][numclust];
double alpha, u, p;
FILE *fp2;
```

```
unitables = gensets;
teststat[0] = observed;
dist[observed]++;

for (i = 0; i <= max; i++){
dist[i] = 0;}

current = unitables[0];
for (i = 1; i < keepers; i++) {

   currentstat = 0;
   alpha = 1.0;
   for (j = 0; j < numclust; j++) {

  currentstat = currentstat + unitables[i][j]*(*(x +j));

   }

   current = unitables[i];
   dist[currentstat] +=1;
   teststat[i] = currentstat;

}

for (i = 0; i < max+1; i++) {
if (dist[i] > 0) {
```

```
   fp2 = fopen(DIST,"a");
   fprintf(fp2,"%d %d %d \n", itter, i, dist[i]);
   fclose(fp2);
   printf("P[t = %d] = %d \n", i, dist[i]);
}
}

p = 0.0;
for (i = observed; i <= max; i++)
p += ((double)dist[i]/((double)keepers-1));

if(p > 0.5) {
p = 0.0;
for(i = observed; i >= 0; i--)
p += ((double)dist[i]/((double)keepers-1));
}

return(p);

}



float ran2(long *idum)
{
 int j;
 long k;
```

```
 static long idum2 = 123456789;
 static long iy=0;
 static long iv[NTAB];
 float temp;

 if (*idum <= 0) {
if (-(*idum) < 1) *idum = 1;
else *idum = -(*idum);
idum2 = (*idum);
for (j = NTAB+7;j>=0;j--) {
k=(*idum)/IQ1;
*idum=IA1*(*idum-k*IQ1)-k*IR1;
if (*idum < 0) *idum += IM1;
if (j < NTAB) iv[j] = *idum;
}
iy=iv[0];
 }
 k=(*idum)/IQ1;
 *idum=IA1*(*idum-k*IQ1)-k*IR1;
 if (*idum <0) *idum += IM1;
 k=idum2/IQ2;
 idum2 = IA2*(idum2-k*IQ2)-k*IR2;
 if (idum2 < 0) idum2 += IM2;
 j= iy/NDIV;
 iy=iv[j]-idum2;
 iv[j] = *idum;
```

```
 if (iy < 1) iy += IMM1;
 if ((temp=AM*iy) > RNMX) return RNMX;
 else return temp;
}
```

B.2 parameterestimate.cpp - C program for calculating MCMC parameter estimates

```
#include <iostream.h>

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <time.h>


#define TABLES "C:\\ExactLR\\MCMC\\C code\\MCTables.txt"

#define STAT "C:\\ExactLR\\MCMC\\C code\\stat.txt"

#define INFILE "C:\\ExactLR\\MCMC\\C code\\bigexample.txt"



#define IM1 2147483563

#define IM2 2147483399

#define AM (1.0/IM1)

#define IMM1 (IM1-1)

#define IA1 40014

#define IA2 40692

#define IQ1 53668

#define IQ2 52774

#define IR1 12211

#define IR2 3791

#define NTAB 32

#define NDIV (1 + IMM1/NTAB)

#define EPS 1.2e-7

#define RNMX (1.0-EPS)
```

```
#define numclust 96
#define numsteps 4000000
#define keepers numsteps/50 + 1
#define burn 50000
#define beta 0.00625

double pvalue(int max, int observed, double gamma, int *x);
void gettable(int *newtable, int *current, int *n,
int *last, int *x);
float ran2(long *idum);

int gensets[keepers][numclust];
long idum = -1999997;

int main() {

int i, j, k, neg, count, corr, sampcorr, sample, teststat, max;
int a, b, c;
int t;
int current[numclust];
int next[numclust];
int last[numclust];
int poss[numclust];
int stats[numclust];
int n[numclust];
```

```
int z[numclust];
int x[numclust];
int *curr = current;
int *nxt = next;

int change[3] = {0,0,0};
FILE *fp;
FILE *fin;
float move, flip;
double changeposs, p;
char filename[20];

fp = fopen(STAT,"w");
fclose(fp);

fin=fopen(INFILE,"r");

t = 0;
corr = 0;
srand (time(0));
max = 0;

for (i = 0; i < numclust; i++) {

fscanf(fin,"%d%d%d",&b,&c,&a);
x[i] = a;
```

```
z[i] = b;
n[i] = c;

current[i] = z[i];
gensets[1][i] = z[i];
t += z[i]*x[i];
corr += z[i]*(n[i]-z[i]);
}

fclose(fin);
stats[0] = t;
gensets[0] = z;
last = current;

j = 1;
while (j < burn ) {

   next = current;
   gettable(&next[0], &current[0], &n[0], &last[0], &x[0]);
   current = next;

   sampcorr = 0;
   teststat = 0;

   for (i = 0; i < numclust; i++){
   sampcorr += current[i]*(n[i] - current[i]);
```

```
   teststat += current[i]*x[i];
   }

   if(sampcorr == corr) {

  j++;
  last = current;

   }

}

sample = 1;
count = 1;



while (sample < keepers ) {

   next = current;
   gettable(&next[0], &current[0], &n[0], &last[0], &x[0]);
   current = next;

   sampcorr = 0;
   teststat = 0;

   for (i = 0; i < numclust; i++){
```

```
   sampcorr += current[i]*(n[i] - current[i]);
   teststat += current[i]*x[i];
   }
   if(sampcorr == corr) {

  count++;
  last = current;

  if (count == 50) {

  gensets[sample] = current;
  if(teststat > max) max = teststat;
  sample++;
  fp = fopen(STAT,"a");
  fprintf(fp, "%d \n",teststat);
  fclose(fp);

  count = 1;

  }
   }

}

p = pvalue(max, t, 0.0, &x[0]);
printf("The pvalue is %lf\n",p);
```

```
  system("PAUSE");
  return 0;
}



void gettable(int *newtable, int *current, int *n, int *last,
int *x)
{
int neg, j, k, two, three, same, t1, t2;
int change[3];
int poss[numclust];

double changeposs, flip, move, split, alpha, u;

two = (numclust*(numclust-1))/2;
three = (numclust*(numclust-1)*(numclust-2))/6;
split = 0.5;
neg = 1;

while (neg) {

  changeposs = ran2(&idum)*(numclust);
  change[0] = (int) changeposs;

  changeposs = ran2(&idum)*(numclust);
  change[1] = (int)changeposs;
```

```
 while (change[1] == change[0]){

changeposs = ran2(&idum)*(numclust);
change[1] = (int) changeposs;

 }

 changeposs = ran2(&idum)*(numclust);
 change[2] = (int) changeposs;

 while (change[2] == change[0] || change[2] == change[1]){

changeposs = ran2(&idum)*(numclust);
change[2] = (int) changeposs;

 }

 move = ran2(&idum);
 flip = ran2(&idum);

 if (flip < 0.5) {

if (move < split) {
neg = (*(newtable + change[0]) + 1 > *(n + change[0]) ||
   *(newtable + change[1]) - 1 < 0);
```

```
if (!neg) {
   *(newtable + change[0]) = *(newtable + change[0]) + 1;
   *(newtable + change[1]) = *(newtable + change[1]) - 1;

   alpha = (double)(*(current + change[1]))*
   (double)(*(n + change[0])-*(current+change[0]))/
   (double)(*(n + change[1])-*(current + change[1])+1)/
   (double)(*(current + change[0])+1);
}
 }
 else{
neg = (*(newtable + change[0]) + 1 > *(n + change[0]) ||
   *(newtable + change[1]) - 2 < 0 ||
   *(newtable + change[2]) + 1 > *(n + change[2]));

if (!neg) {
   *(newtable + change[0]) = *(newtable + change[0]) + 1;
   *(newtable + change[1]) = *(newtable + change[1]) - 2;
   *(newtable + change[2]) = *(newtable + change[2]) + 1;

   alpha = (double)(*(current + change[1]))*
   (double)(*(current + change[1])-1)*
   (double)(*(n + change[0])-*(current + change[0]))*
   (double)(*(n + change[2])-*(current + change[2]))/
   (double)(*(current + change[0])+1)/
```

```
   (double)(*(current + change[2])+1)/
   (double)(*(n + change[1])-*(current + change[1])+1)/
   (double)(*(n + change[1])-*(current + change[1])+2);
}
 }
  }

  else {

 if (move < split) {
neg = (*(newtable + change[0]) - 1 < 0 ||
   *(newtable + change[1]) + 1 > *(n + change[1]));

if (!neg) {
   *(newtable + change[0]) = *(newtable + change[0]) - 1;
   *(newtable + change[1]) = *(newtable + change[1]) + 1;

   alpha = (double)(*(current + change[0]))/
   (double)(*(n + change[0])-*(current+change[0])+1)*
   (double)(*(n + change[1])-*(current + change[1]))/
   (double)(*(current + change[1])+1);
}
 }
 else{

neg = (*(newtable+change[0]) - 1 < 0 ||
```

```
   *(newtable+change[1]) + 2 > *(n + change[1]) ||
   *(newtable+change[2]) - 1 < 0);

if (!neg) {
   *(newtable+change[0]) = *(newtable+change[0]) - 1;
   *(newtable+change[1]) = *(newtable+change[1]) + 2;
   *(newtable+change[2]) = *(newtable+change[2]) - 1;

   alpha = (double)(*(current+change[0]))*
   (double)(*(current+change[2]))*
   (double)(*(n+change[1])-*(current+change[1]))*
   (double)(*(n+change[1])-*(current+change[1])-1)/
   (double)(*(current+change[1])+2)/
   (double)(*(current+change[1])+1)/
   (double)(*(n+change[0])-*(current+change[0])+1)/
   (double)(*(n+change[2])-*(current+change[2])+1);
}
 }
  }

}

 t1 = 0;
 t2 = 0;

for (j = 0; j < numclust; j++) {
```

```
t1 += *(x + j)*(*(current + j));
t2 += *(x + j)*(*(newtable + j));
 }

 alpha *= exp(beta*(double)(t2 - t1));
 u = ran2(&idum);



 if (alpha < u){
 for (j = 0; j <numclust; j++)
  *(newtable+j) = *(current + j);
 }
}



double pvalue(int max, int observed, double gamma, int *x)
{
 int i, j, k, currentstat;
 int teststat[keepers], dist[max+1], current[numclust];
 int unitables[keepers][numclust];
 double alpha, u, p;
 FILE *fp2;

 unitables = gensets;
 teststat[0] = observed;
 dist[observed]++;
```

```
 for (i = 0; i <= max; i++){
 dist[i] = 0;}

 current = unitables[0];
 for (i = 1; i < keepers; i++) {

currentstat = 0;
alpha = 1.0;
for (j = 0; j < numclust; j++) {

currentstat = currentstat + unitables[i][j]*(*(x +j));

}

current = unitables[i];
dist[currentstat] +=1;
teststat[i] = currentstat;

 }



 for (i = 0; i < max+1; i++) {
  if (dist[i] > 0) {
  printf("P[t = %d] = %d \n", i, dist[i]);
  }
```

```
 }

 p = 0.0;
 for (i = observed; i <= max; i++)
  p += ((double)dist[i]/((double)keepers-1));

 return(p);
}



float ran2(long *idum)
{
 int j;
 long k;
 static long idum2 = 123456789;
 static long iy=0;
 static long iv[NTAB];
 float temp;

 if (*idum <= 0) {
 if (-(*idum) < 1) *idum = 1;
 else *idum = -(*idum);
 idum2 = (*idum);
 for (j = NTAB+7;j>=0;j--) {
 k=(*idum)/IQ1;
 *idum=IA1*(*idum-k*IQ1)-k*IR1;
```

```
 if (*idum < 0) *idum += IM1;
 if (j < NTAB) iv[j] = *idum;
 }
 iy=iv[0];
 }
 k=(*idum)/IQ1;
 *idum=IA1*(*idum-k*IQ1)-k*IR1;
 if (*idum <0) *idum += IM1;
 k=idum2/IQ2;
 idum2 = IA2*(idum2-k*IQ2)-k*IR2;
 if (idum2 < 0) idum2 += IM2;
 j= iy/NDIV;
 iy=iv[j]-idum2;
 iv[j] = *idum;
 if (iy < 1) iy += IMM1;
 if ((temp=AM*iy) > RNMX) return RNMX;
 else return temp;
}
```

VITA

Published Journal Articles

- Cook LJ, Hoggins JL, Olson LM. Observed seatbelt usage among drivers of heavy commercial vehicles in Utah. *Accid. Anal. Prev. In Press.*

- Weiss HB, Sauber-Schatz EK, Cook LJ. The epidemiology of pregnancy-associated emergency department injury visits and their impact on birth outcomes. *Accid. Anal. Prev. In Press.*

- Li Z, Knight S, Cook LJ, Hyde LK, Holubkov R, Olson LM. Modeling motor vehicle crashes for street racers using zero-inflated models. *Accid. Anal. Prev.* 2008;40:pp. 835 - 9.

- Bissonette JA, Kassar C, Cook LJ, An assessment of costs associated with deer-vehicle collisions: Human death and injury, vehicle damage, and deer loss. *Human-Wildlife Conflicts* 2008;2:pp. 17 - 27.

- Zhu M, Cummings P, Chu H, Cook LJ, Association of rear seat safety belt use with death in a traffic crash: a matched cohort study. *Inj. Prev.* 2007;13:pp. 183-5.

- Markenson D, Tunik M, Cooper A, Olson LM, Cook L, Matza-Haughton H, et al. A national assessment of knowledge, attitudes, and confidence of prehospital providers in the assessment and management of child maltreatment. *Pediatrics.* 2006;119:pp. e103-e108.

- Donaldson AE, Cook LJ, Hutchings CA, Dean JM. Crossing county lines: The impact of crash location and driver's residence on motor vehicle crash fatality. *Accid. Anal. Prev.* 2006;38:pp. 723-7.

- Zhu, M, Hardman SB, Cook LJ, Backseat safety belt use and crash outcome. *J. Safety Res.* 2005;36:pp. 505-7.

- Cook LJ, Knight S, Olson LM, A comparison of aggressive and DUI crashes. *J. Safety Res.* 2005;36:pp. 491-3.

- Hyde LK, Cook LJ, Knight S, Olson LM, Graduated driver licensing in Utah: is it effective? *Ann. Emerg. Med.* 2005;45:pp. 147-54.

- Mann NC, Knight S, Olson LM, Cook LJ. Underestimating injury mortality using statewide databases. *J Trauma* 2005;58:pp. 162-7.

- Cochran A, Mann NC, Dean JM, Cook LJ, Barton RG. Resource utilization and its management in splenic trauma. *Am. J. Surg.* 2004;187:pp. 713-719.

- Cook LJ, Knight S, Junkins EP, Mann CL, Dean JM, Olson LM. Repeat patients to the emergency department in a statewide database. *Acad. Emerg. Med.* 2004;11:pp. 256-263.

- Smith R, Cook LJ, Olson LM, Reading JC, Dean JM: Trends of behavioral risk factors in motor vehicle crashes in Utah, 1992 - 1996. *Accid. Anal. Prev.* 2004;36: pp. 249 - 255.

- Vernon DD, Cook LJ, Peterson KJ, Dean JM: Effect of repeal of the national maximum speed limit law on occurrence of crashes, injury crashes, and fatal crashes on Utah highways. *Accid. Anal. Prev.* 2004;36: pp. 223-229

- Knight S, Cook LJ, Olson LM. The fast and the fatal: Street racing fatal crashes in the United States. *Inj. Prev.* 2004;10:pp. 53-55

- Knight S, Olson LM, Cook LJ, Mann NC, Corneli HM, Dean JM. Against all advice: an analysis of refusals of care. *Ann. Emerg. Med.* 2003;42: pp. 689-696.

- Hyde LK, Cook LJ, Olson LM, Weiss HB, Dean JM. Effect of motor vehicle crashes on adverse fetal outcomes. *Obstet. Gynecol.* 2003;102:pp. 279 - 86.

- Vernon DD, Diller EM, Cook LJ, Reading JC, Suruda AJ, Dean JM: Evaluating the crash and citation rates of Utah drivers licensed with medical conditions, 1992 - 1996. *Accid. Anal. Prev.* 2002;34:pp. 237 - 46.

- Skokan EG, Olson LM, Cook LJ, Corneli HM. Snowmobile injuries in Utah. *Acad. Emerg. Med.* 2001;8:pp. 1173-7.

- Cvijanovich NZ, Cook LJ, Mann NC, Dean JM. Pediatric all-terrain vehicle injuries. *Pediatrics.* 2001;108:pp. 631-5.

- Cook LJ, Olson LM, Dean JM. Probabilistic record linkage: Relationships between file sizes, identifiers, and match weights. *Methods Inf. Med.* 2001;40:pp. 196-203.

- Dean JM, Vernon DD, Cook LJ, Nechodom PJ, Reading JC, Suruda A. Probabilistic linkage of computerized ambulance and inpatient hospital discharge records: A potential tool for evaluation of emergency medical services. *Ann. Emerg. Med.* 2001;37:pp. 616-26.

- Cvijanovich NZ, Cook LJ, Mann NC, Dean JM, Graduated driver licensing restrictions. *Pediatrics.* 2001;107:pp. 632-7.

- Knight S, Cook LJ, Nechodom PJ, Olson LM, Reading JC, Dean JM. Shoulder belts in motor vehicle crashes: a statewide analysis of restraint efficacy. *Accid. Anal. Prev.* 2001;33:pp. 65-71.

- Corneli HM, Cook LJ, Dean JM. Adults and children in severe motor vehicle crashes: A matched-pairs study. *Ann. Emerg. Med.* 2000;36:pp. 340-5.

- Cook LJ, Knight S, Olson LM, Nechodom PJ, Dean JM. Crash characteristics and medical outcomes of older drivers in motor vehicle crashes in Utah, 1992 - 1995. *Ann. Emerg. Med.* 2000;35:pp. 585-591.

- Berg M, Cook LJ, Corneli H, Vernon D, Dean JM. Effect of seating position and restraint use on injuries to children in motor vehicle crashes. *Pediatrics.* 2000;105:pp. 831-835.

- Suruda AJ, Vernon DD, Reading J, Cook LJ, Nechodom PJ, Leonard D, Dean JM. Pre-hospital emergency medical services: A population-based study of pediatric utilization. *Inj. Prev.* 1999;5:pp. 294-297.

- Cvijanovich NZ, Cook LJ, Nechodom PJ, Dean JM. A population-based study of teenage drivers: 1992-1996. *43rd Annual Proceedings Association for the Advancement of Automotive Medicine.* 1999:pp. 175-186.

- Leonard DR, Suruda AJ, Cook LJ, Reading J, Mobasher H, Dean JM. Distinctive emergency department usage for injury for worker's compensation cases in Utah in 1996. *Occup. Med.* 1999;41:pp. 686-692.

Technical Reports

- Diller EM, Cook LJ, Leonard DR, Dean M, Reading JM, Vernon DD. Evaluating Drivers Licensed with Medical Conditions Licensed with Medical Conditions

in Utah, 1992 - 1996. National Highway Traffic Safety Administration 1999; Report No. DOT HS 809 023.

- Vernon DD, Diller E, Cook LJ, Reading J, Dean JM. Further Analysis of Drivers Licensed with Medical Conditions in Utah. National Highway Traffic Association. 2001; Report No. DOT HS 809 211.

Published Abstracts

- Cook LJ, Olson LM, Dean JM. Usefulness of name information in probabilistic record linkage. *Pediatr. Emerg. Care*. 2000;16(1):p. 66.
- Cook LJ, Olson LM, Mann NC, Dean JM. Analysis of pediatric emergency department visits. *Pediatr. Emerg. Care*. 2000;16(1):p. 66.
- Cvijanovich NZ, Mann NC, Cook LJ, Dean JM. A study of ATV injuries in Utah. *Pediatr. Emerg. Care*. 2000;16:p. 66.
- Knight S, Cook LJ, Olson LM, Mann NC. Validity and reliability of death information on hospital records. *Pediatr. Emerg. Care*. 2000;16:p. 66.