UNIVERSITY OF CALIFORNIA
Los Angeles

Statistical Methods for Multivariate Genetic Analysis

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Biostatistics

by

Soo Min Ji

2022

ABSTRACT OF THE DISSERTATION

Statistical Methods for Multivariate Genetic Analysis

by

Soo Min Ji
Doctor of Philosophy in Biostatistics
University of California, Los Angeles, 2022
Professor Hua Zhou, Co-Chair
Professor Kenneth L. Lange, Co-Chair

This dissertation develops statistical and computational methods for human genetics. We consider modern solutions to estimate the power of proposed genetic studies, and propose an alternative to the mixed model framework for analysis on non-Gaussian distributions. The methods we develop are designed for multivariate simulation and analysis in high dimensions. We implement our methods in individual, open-sourced Julia packages. They are freely available to the scientific community through the OpenMendel platform.

The dissertation of Soo Min Ji is approved.

Eric M. Sobel

Donatello Telesca

Kenneth L. Lange, Committee Co-Chair

Hua Zhou, Committee Co-Chair

University of California, Los Angeles

2022

To my mom and dad, who loved me so much

TABLE OF CONTENTS

1 Introduction **1**

2 Realistic Trait Simulation: TraitSimulation.jl **3**

2.1 Motivation 3

2.2 Implementation 4

2.2.1 Julia 4

2.2.2 SNP Data 5

2.2.3 Trait Simulation 5

2.3 Results 10

2.3.1 Statistical Power 12

2.3.2 Benchmarks 15

2.4 Conclusions 15

2.5 Availability of source code 17

3 A Flexible Quasi-Copula Distribution for Statistical Modeling: QuasiCopula.jl **19**

3.1 Motivation 19

3.1.1 GLMM Framework 21

3.1.2 Tonda's Framework 21

3.2 Definitions 22

3.3 Moments 24

3.4 Marginal and Conditional Distributions 26

3.5 Generation of Random Deviates 27

3.6 Parameter Estimation . . . 29
3.6.1 Mean Components . . . 29
3.6.2 Structured Covariance . . . 30
3.6.3 MM Algorithm for the VC Model Parameters . . . 31
3.6.4 Initialization . . . 33
3.7 Statistical Properties . . . 33
3.7.1 Compound Symmetric Covariance . . . 33
3.8 Results . . . 34
3.8.1 Simulation Studies . . . 34
3.8.2 Bivariate Mixed Outcome Model . . . 44
3.8.3 NHANES Data Example . . . 46
3.9 Discussion . . . 47
3.10 Supplemental Material . . . 49
3.10.1 Tonda's Approximation Details . . . 49
3.10.2 Generate Random Deviates . . . 50
3.10.3 Parameter Estimation: . . . 64
3.10.4 Quasi-Newton Algorithm . . . 69
3.10.5 Negative Binomial . . . 71
3.10.6 Compound Symmetric Covariance . . . 74
3.10.7 Gaussian Base . . . 75
3.10.8 Additional Simulation Study Results . . . 82
3.11 Availability of source code . . . 104

4 Genome-Wide Association Analysis with Quasi-Copula . . . 105

4.1 Motivation . . . 105

4.2 Genetic Model . . . 105

4.3 Score Test for Individual SNPs or SNP-sets . . . 106

4.3.1 Score and Approximate Observed Information . . . 106

4.3.2 Score Test Statistic . . . 108

5 Conclusions and Future Research . . . 109

5.1 Quasi-Copula model for Random Sample GWAS . . . 110

5.1.1 Steepest Ascent Estimation of an Unstructured Covariance . . . 110

5.2 GWAS on Pedigree and Structured Population Data . . . 111

5.2.1 Quasi-Copula model for Pedigree GWAS . . . 112

5.2.2 Pedigree GWAS Example: Bivariate Trait . . . 112

5.2.3 An MM-Algorithm for Variance Component Matrices . . . 113

5.2.4 Surrogate Function . . . 114

LIST OF FIGURES

2.1 `OpenMendel` Pipeline Example: Illustrates how `TraitSimulation.jl` fits within the `OpenMendel` software pipeline to assess the statistical power of different association studies. 10

2.2 Case Study 1: Power under an Ordinal Multinomial Model. This example shows the power to detect a single causal SNP in UK Biobank data with four outcome categories for disease status. Using an ordinal multinomial simulation model and the `OpenMendel` module for ordinal trait regression [11], we assume a single SNP as a fixed effect and control for sex and standardized age. The figure compares analysis results for three SNPs of varying MAF over 1000 simulation replicates each. For each SNP, the graph depicts the power to detect that SNP at significance level $\alpha = 5 \times 10^{-8}$. For each SNP, the effect size varies from 0 to 0.05 in increments of 0.001. On the x-axis, we exponentiate effect sizes to covert to odds ratios. See the text for a detailed description of the model. 11

2.3 Case Study 2: Power under Univariate and Bivariate Variance Components Models. This example shows the power to detect a single causal SNP using both univariate and bivariate variance components simulation models and the `OpenMendel` module for variance components analysis [34]. For each anlysis, each line in the graph depicts the power to detect a SNP with MAF $= 0.23$ using 1000 simulations at significance level $\alpha = 5 \times 10^{-8}$. The SNP effect size varies from 0 to 0.065 in increments of 0.002 in the center range (0.016 – 0.032) and increments of 0.005 in the two end ranges. On the x-axis, we convert the SNP MAF and effect sizes into the proportion of variation explained by the SNP. See the text for a detailed description of the model. 11

3.1 Simulation I: Mean squared errors (MSE) of parameter estimates β and θ under the Poisson base distribution with log link function and a single VC versus a random intercept GLMM fit via `MixedModels.jl`. Each scenario result includes 100 replicates. . . . 36

3.2 Simulation I: Mean squared errors (MSE) of parameter estimates β and θ under the negative binomial base distribution with log link function and a single VC versus a random intercept GLMM fit via `MixedModels.jl`. Each scenario result includes 100 replicates. . . . 37

3.3 Simulation II: Mean squared errors (MSE) of parameter estimates β and $\theta = 0.05$ under the Poisson base distribution with log link function and a single VC versus a random intercept GLMM fit via `MixedModels.jl`. Each scenario reports involves 100 replicates. 38

3.4 Simulation II: Mean squared errors (MSE) of parameter estimates β and $\theta = 0.05$ under the negative binomial base distribution with log link function and a single VC versus a random intercept GLMM fit via `MixedModels.jl`. Each scenario result includes 100 replicates. . . . 39

3.5 Simulation II: Mean squared errors (MSE) of parameter estimates β and $\theta = 0.01$ under the Poisson base distribution with log link function and a single VC versus a random intercept GLMM fit via `MixedModels.jl`. Each scenario result includes 100 replicates. . 40

3.6 Simulation II: Mean squared errors (MSE) of parameter estimates β and $\theta = 0.01$ under the negative binomial base distribution with log link function and a single VC versus a random intercept GLMM fit via `MixedModels.jl`. Each scenario result includes 100 replicates. . . . 41

3.7 Simulation I: Mean squared errors (MSE) of MLE estimates β and $\theta = 0.1$ under the Bivariate Mixed Outcome model with Poisson and Bernoulli base distributions and their canonical link functions. Each scenario reports involves 100 replicates. 45

3.8 Mean squared errors (MSE) of parameter estimates β and θ under the AR(1) covariance for the Poisson base distribution with log link function. Each scenario reports involves 100 replicates. 84

3.9 Mean squared errors (MSE) of parameter estimates β and θ under the AR(1) covariance for the negative binomial base distribution with log link function. Each scenario reports involves 100 replicates. 85

3.10 Mean squared errors (MSE) of parameter estimates β and θ under the AR(1) covariance for the Bernoulli base distribution with logit link function. Each scenario reports involves 100 replicates. 86

3.11 Mean squared errors (MSE) of parameter estimates β and θ under the AR(1) covariance for the Normal base distribution with Identity link function. Each scenario reports involves 100 replicates. 87

3.12 Mean squared errors (MSE) of parameter estimates β and θ under the CS covariance for the Poisson base distribution with log link function. Each scenario reports involves 100 replicates. 88

3.13 Mean squared errors (MSE) of parameter estimates β and θ under the CS covariance for the negative binomial base distribution with log link function. Each scenario reports involves 100 replicates. 89

3.14 Mean squared errors (MSE) of parameter estimates β and θ under the CS covariance for the Bernoulli base distribution with logit link function. Each scenario reports involves 100 replicates. 90

3.15 Mean squared errors (MSE) of parameter estimates β and θ under the CS covariance for the Normal base distribution with Identity link function. Each scenario reports involves 100 replicates. 91

3.16 Simulation I: Mean squared errors (MSE) of parameter estimates β and θ under the Bernoulli base distribution with logit link function and a single VC versus a random intercept GLMM fit via `MixedModels.jl`. Each scenario reports involves 100 replicates. 92

3.17 Simulation I: Mean squared errors (MSE) of parameter estimates β and θ under the Normal base distribution with Identity link function and a single VC versus a random intercept LMM fit via `MixedModels.jl`. Each scenario reports involves 100 replicates. . 93

3.18 Simulation II: Mean squared errors (MSE) of parameter estimates β and $\theta = 0.05$ under the Bernoulli base distribution with logit link function and a single VC versus a random intercept GLMM fit via `MixedModels.jl`. Each scenario reports involves 100 replicates. 94

3.19 Simulation II: Mean squared errors (MSE) of parameter estimates β and $\theta = 0.01$ under the Bernoulli base distribution with logit link function and a single VC versus a random intercept GLMM fit via `MixedModels.jl`. Each scenario reports involves 100 replicates. 95

3.20 Simulation II: Mean squared errors (MSE) of parameter estimates β and $\theta = 0.05$ under the Normal base distribution with Identity link function and a single VC versus a random intercept LMM fit via `MixedModels.jl`. Each scenario reports involves 100 replicates. . 96

3.21 Simulation II: Mean squared errors (MSE) of parameter estimates β and $\theta = 0.01$ under the Normal base distribution with Identity link function and a single VC versus a random intercept LMM fit via `MixedModels.jl`. Each scenario reports involves 100 replicates. . 97

LIST OF TABLES

2.1 Simulation Models Included in `TraitSimulation.jl`. 6

2.2 Syntax for Model Construction and for the `simulate` Function. (See text for variable definitions.) 6

2.3 For the Ordered Multinomial Model, Power Calculation Runtimes in Seconds. A total of 1000 replications were performed for each combination (k,n) of the number of causal SNPs and the sample size. This was repeated for several SNP effect sizes (see Figure 2.2) and the median runtimes are recorded here. 14

2.4 For the Univariate and Bivariate Variance Components Model, Power Calculation Runtimes in Seconds. A total of 1000 replications were performed for each combination (k,n) of the number of causal SNPs and the sample size. This was repeated for several SNP effect sizes (see Figure 2.3) and the median runtimes are recorded here. 15

3.1 Run times and (standard error of run times) in seconds based on 100 replicates under simulation II with Poisson and negative binomial (NB) Base, $\theta_{\text{true}} = 0.01$, sampling unit size d_i and sample size n. 43

3.2 MLE's and (confidence intervals) based on a single replicate under simulation II with negative binomial Base, $\theta_{\text{true}} = 0.01$, sampling unit size $d_i = 5$ and sample size $n = 10000$. 44

3.3 Run times and (confidence interval run times) in seconds based on a single replicate under simulation II with negative binomial Base, $\theta_{\text{true}} = 0.01$, sampling unit size $d_i = 5$ and sample size $n = 10000$. 44

3.4 Random intercept MLEs, loglikelihoods, and run times for for the NHEFS data under the quasi-copula (QC) model and GLMM. All $n = 1537$ sampling units are of size $d_i = 2$. 47

3.5 Run times and (standard error of run times) in seconds based on 100 replicates for Poisson Base under AR(1) and CS covariance structure with sampling unit size d_i and sample size n. 98

3.6 Run times and (standard error of run times) in seconds based on 100 replicates for negative binomial (NB) Base under AR(1) and CS covariance structure with sampling unit size d_i and sample size n. 99

3.7 Run times and (standard error of run times) in seconds based on 100 replicates for Bernoulli Base under AR(1) and CS covariance structure with sampling unit size d_i and sample size n. 100

3.8 Run times and (standard error of run times) in seconds based on 100 replicates for Gaussian Base under AR(1) and CS covariance structure with sampling unit size d_i and sample size n. 101

3.9 Run times and (standard error of run times) in seconds based on 100 replicates under simulation II with Bernoulli Base, $\theta_{\text{true}} = 0.01$, sampling unit size d_i and sample size n. . 102

3.10 Run times and (standard error of run times) in seconds based on 100 replicates under simulation II with Gaussian Base, $\theta_{\text{true}} = 0.01$, sampling unit size d_i and sample size n. . 103

ACKNOWLEDGMENTS

Let me begin by expressing my immeasurable gratitude to my advisors Janet Sinsheimer, Hua Zhou and Kenneth Lange. Janet has been the best academic advisor and role model for the entirety of my time at graduate school. Her dedication to science across a broad range of disciplines, charisma, creativity, and enthusiasm are both inspiring and infectious. Hua has been the best technical mentor I could have hoped for, knowledgeable, patient, caring, and leads by example. He has taught me everything I know about scientific computing and julia, and has set a lasting precedent for proper coding practices and reproducible research. Ken has been equally supportive to me, brilliant, charming, humorous, and undeniably talented in mathematics. Despite his numerous academic contributions and achievements, I admire his wit and humility the most. Their academic expertise and compassion have provided a safe learning environment for me to grow both personally and professionally, for which I am forever grateful.

I also met wonderful mentors who profoundly guided my growth. I thank Donatello Telesca for his enthusiasm and support in and out of classes. I thank Eric Sobel and Jeanette Papp not only for their mentorship as directors under the Genomic Analysis Training Program and faculty members of the `OpenMendel` group, but also for believing in me and encouraging me through difficult times.

During my time as a PhD student, I was also lucky to have met many of my closest friends with whom I hope to remain lifelong friends. I have met great companions throughout my graduate school journey in my cohort, and I also thoroughly enjoyed the research banter with students I met in the Biomath department. I also want to thank the student members of the `OpenMendel` group, for giving me such a wonderful group to belong to and learn from. Your company has not only inspired me to be a better student and researcher, but also made this journey enjoyable.

In the end, let me thank my family and partner for their never-ending encouragement and support. Mom and Dad, thank you for all the sacrifices you both made for the sake of our education.

VITA

2011–2015 B.S. Applied Mathematics, University of California, San Diego

2016–2020 Graduate student researcher

2020–2022 Genomic Analysis Training Program (T32 HG002536), UCLA

PUBLICATIONS

Ji SS, Chu BB, Sinsheimer JS, Zhou H, Zhou JJ, Lange K. "A flexible quasi-copula distribution for statistical modeling" *Journal of Multivariate Analysis*. To be submitted 2022 May

Ji SS, German CA, Lange K, Sinsheimer JS, Zhou H, Zhou JJ, Sobel EM. "Modern simulation utilities for genetic analysis" *BMC Bioinformatics*. 2021 May

Levine AJ, Soontornniyomkij V, Masliah E, Sinsheimer JS, **Ji SS**, Horvath S, Singer EJ, Kallianpur A, Moore DJ. "A candidate gene study of intermediate histopathological phenotypes in HIV-associated neurocognitive disorders" *J Neurovirol.* 2020 Aug

Zhou H, Sinsheimer JS, Bates DM, Chu BB, German CA, **Ji SS**, Keys K, Mosher G, Papp J, Sobel EM, Zhai J, Zhou J, Lange K (2020). "OpenMendel: a cooperative programming project for statistical genetics" *Human genetics* 139.1 2020 Jan

vonHoldt BM, **Ji SS**, Aardema ML, Stahler DR, Udell MAR, Sinsheimer JS. "Activity of Genes

with Functions in Human Williams-Beuren Syndrome Is Impacted by Mobile Element Insertions in the Gray Wolf Genome" *Genome Biol Evol* 2018 June

CHAPTER 1

Introduction

It is important for geneticists to consider new computational tools that meet all the challenges of contemporary studies, especially when modeling random effects in genetic data. Statistical geneticists employ simulation to estimate the power of proposed studies, test new analysis tools, and evaluate the implications of different model specifications. Although there are existing trait simulators, there is ample room to modernize available simulation models and computing platforms. Project I aims to address this challenge by giving users the ability to easily simulate phenotypic traits under generalized linear models (GLMs), linear mixed models (LMMs), and generalized linear mixed models (GLMMs), conditional on PLINK formatted genotype data. Project I provides customized simulation utilities that accompany specific genetic analysis options in OpenMendel; for example, ordered, multinomial traits. For random effects, we provide simulation utilities both under the LMM and GLMM framework. This project has been completed and approved for publication with BMC Bioinformatics on March 25, 2021.

The last two projects are motivated by the genetic community's pressing need for an alternative to the mixed model framework for analysis on non-Gaussian distributions. Many genome-wide association study (GWAS) tools are limited to independent observations and single phenotype analysis. As genomic data sets are growing in size and complexity, more flexible GWAS tools are needed to better accommodate modern problems. Existing algorithms for fitting GLMM in modern genomic studies are not scalable since integration over the random effects has proven intractable [5] [8]. Generalized estimating equations (GEEs) have since become the go-to alternative to GLMMs, despite their drawbacks which our second project aims to address: (1) GEEs lack a well-defined likelihood and (2)

GEE estimation searches often fail to converge. Project II aims to introduce a new class of models as a viable alternative to GLMMs and GEEs for efficient analysis in the presence of non-Gaussian distributions and random effects. For this project, I am working on a methods article "A Flexible Quasi-Copula Distribution for Statistical Modeling" that will provide the theoretical underpinnings as well as show the computational advantages of the approach. This project has been completed and submitted for publication to Annals of Applied Statistics on April 13, 2022. Project III will expand upon Project II, demonstrating how the new class of models has useful qualities in a contemporary genomic context. Specifically, we aim to find an extension of the model in Project II to allow for multivariate GWAS using the computationally efficient score test. In all of these projects, we focus on ensuring the methods developed scale to large sample sizes so they can be applied to modern scale problems. Each method is implemented in individual open-source Julia packages, freely available to the scientific community through the `OpenMendel` statistical genetics ecosystem [35].

CHAPTER 2

Realistic Trait Simulation: TraitSimulation.jl

2.1 Motivation

There is a lack of software available to geneticists who wish to calculate power and sample sizes in designing a study on genetics data. Typically, the study power depends on assumptions about the underlying disease model. Many power calculating software tools operate as a black box and do not allow for customization. To develop custom tests, researchers can develop their own simulation procedures to carry out power calculations. One limitation with many existing methods for simulating traits conditional on genotypes is that these methods are limited to normally distributed traits and to fixed effects. The TraitSimulation.jl package gives users the ability to easily simulate phenotypic traits under GLMs or LMMs conditional on PLINK formatted genotype data [5]. We include customized simulation utilities that accompany specific genetic analysis options in `OpenMendel`; for example, ordered, multinomial traits. For random effects, we currently limit the user to linear mixed models although we are exploring the possibility of generalized linear mixed models as these will be needed in my copula work. A single simulate function allows for the simulation of both independent traits and multiple correlated traits under a fixed effect model or a mixed effect model. The program also uses standard PLINK data format.

Many power calculating software tools operate as a black box and do not allow for customization. Another limitation with many existing methods for simulating traits conditional on genotypes is that these methods are limited to normally distributed traits and to fixed effects. For example, most phenotype simulators are limited to Gaussian traits or traits transformable to normality, ignoring

qualitative traits and realistic, non-normal trait distributions.

In the sections below, we first explain the advantages of using Julia to develop the `OpenMendel` project and demonstrate some of its key language features. We then present `OpenMendel`'s efficient SNP data management tool `SnpArrays`. Finally, we outline the trait simulation procedure with example data and results in various case studies that critically employ the simulated phenotypes in downstream analyses. The realistic trait simulation models used in these case studies are not all available in any other simulation software that we are aware of. Users are not limited to the analysis options provided in `OpenMendel` or Julia. After simulating the desired trait, they can call other analysis program, including popular R, C++ or Python packages, while staying within the Julia or Jupyter notebook environments. Alternatively, they can output the simulated traits to files for more customizable downstream analyses.

2.2 Implementation

This section sketches our implementation of the `TraitSimulation.jl` package.

2.2.1 Julia

The Julia programming language provides an excellent computing environment for genetic association studies. Among Julia's many features is its just-in-time compiler that allows the language to combine the speed and efficiencies of low-level languages such as C or C++ with the ease-of-use and understandable syntax of high-level languages such as R or Python. Julia's speed is also enhanced by its automatic use of the tremendous parallelization built into modern CPUs. For example, Julia includes automatic instruction-level parallelism, vectorization (to carry out many mathematical operations simultaneously), and multi-threading (to have whole sections of code run in parallel); Julia even includes tools for distributed computing across massive computing clusters [16, 15, 14]. To make coding easier and more efficient, Julia also has automatic type checking (to ensure variable consistency) and multiple dispatch (in which a single function can be used with different types and

amounts of data as input and still have optimal efficiency). In addition, Julia ships with a native package manager, which improves portability, ease of deployment, and reproducibility. Julia also allows users to easily maneuver between shared and distributed memory environments, including graphics processing units (GPUs). Julia's efficiency and versatility solves the long-standing two-language problem, in which developers could quickly prototype software in higher-level languages such as R or Python but then must rewrite their prototype code in lower-level languages such as C or C++ to handle larger, real-world data sets. As genetic data evolves and grows, and more resource-intensive tools are required to perform analyses, these design features make Julia a compelling language for computational genetics.

2.2.2 SNP Data

Our Julia-based package `SnpArrays.jl` [30] is a versatile interface to SNP data for all of our `OpenMendel` modules, and potentially for other packages. Users can specify SNPs of interest by name, position, minor allele frequency (MAF), or other filtering criterion provided by `SnpArrays.jl`. A remarkable feature is that after reading in compressed SNP data files, `SnpArrays.jl`keeps all of the genotype data compressed during its computations, such as estimation of genetic relationship matrices (GRMs) and principal components (PCs). This feature reduces RAM requirements by orders of magnitude while maintaining extremely fast performance. This is all possible because Julia allows for operations such as matrix-matrix multiplication to be defined on BitArrays, which are arrays where each element is one or two bits. This permits real analysis of biobank-scale data on commodity-level computers, which was accomplished with our software in [11].

2.2.3 Trait Simulation

Our new `TraitSimulation.jl` program provides the broad range of underlying models listed in Table 2.1, including ordered multinomial models, generalized linear models (GLMs), and generalized linear mixed models (GLMM). `TraitSimulation.jl` allows users to easily modify the models

	Simulation Model	Relatedness Data	Typical Application
(1)	Generalized Linear Models	-	Exponential-Family Traits
(2)	Case/Control Models	-	Disease Status Traits
(3)	Proportional Hazards/Odds Models	-	Ordinal Traits
(4)	Variance Component Models	GRM	Correlated Normal Traits
(5)	Generalized Linear Mixed Models	GRM	Correlated Non-Normal Traits

Table 2.1: Simulation Models Included in `TraitSimulation.jl`.

Simulation Model in Table 2.1	Model Construction Syntax
(1)	model = GLMTrait(X, β, dist, link)
	model = GLMTrait(X, β, G, γ, dist, link)
(2, 3)	model = OrderedMultinomialTrait(X, β, θ, link)
(4)	model = VCMTrait(X, β, vc)
	model = VCMTrait(formula, df, vc)
	model = VCMTrait($X, \beta, G, \gamma, \Sigma, V$)
(5)	model = GLMMTrait(X, β, vc, dist, link)
Simulation Model in Table 2.1	**`simulate` Function Syntax**
(1, 3, 4, 5)	`simulate`(model)
(2)	`simulate`(model, Logistic = true, cutoff = 2)

Table 2.2: Syntax for Model Construction and for the `simulate` Function. (See text for variable definitions.)

in Table 2.1 to fit their needs or to create entirely new simulation models. Table 2.2, whose variables are defined below, conveys the basic syntax for model construction and running the simulation under that model. This flexibility allows users to relax strict distributional assumptions imposed by many existing packages and simulate traits that do not conform to normality restrictions. Greater fidelity to trait distributions is bound to improve analysis results. Users interested in studying the robustness of their model can assess the effects of model misspecification by simulating the trait data under the hypothesized model and then analyzing the data under different models. An explicit example of the use of our software for this purpose can be seen in [11] where they found a decrease in power when analyzing ordered multinomial phenotypes under a linear or logistic regression model.

To run our `TraitSimulation.jl` package the typical five steps are:

1. Load the required packages: `SnpArrays.jl`and `TraitSimulation.jl`.
2. Read in PLINK data files via `SnpArrays.jl`and estimate the GRM, if applicable.
3. Construct the simulation model, including relevant parameters such as the genetic and non-genetic predictors, the variance components, etc.
4. Call the simulation routine to sample from the constructed model.
5. Output the simulated phenotypes to a file or pass them to other analyses.

The following Julia code snippet is an example of commands used to perform the above steps for model (4) in Table 2.1, based on genotype data in an existing compressed file. Prospective users may interact with a comprehensive online tutorial with step-by-step instructions and sample code at [13].

```
# Load the packages
using SnpArrays, TraitSimulation

# Read the genetic data into a SNPArray
snps = SnpArray("genotypefile.bed", n) # sample size n
GRM = grm(snps, minmaf = 0.05) # filter on frequency to
        # estimate the Genetic Relationship Matrix (GRM)
locus = convert(Vector{Float64},  # load model genotypes
                @view snps[:, snp_index]; impute = true)

# Specify the genetic and non-genetic predictors
X = [intercept sex locus] # predictor matrix for model
B = [[12.0; 2.0; 0.005;] [20.0; 2.0; 0.01]]
          # corresponding regression coefficients matrix

```

```
# Specify the variance components for simulation
I_n = Matrix(I, n, n) # the n x n identity matrix
variance_formula = @vc Σ_A ⊗ 2GRM + Σ_E ⊗ I_n;

# Construct the model variable via the desired framework
trait_model = VCMTrait(X, B, variance_formula)
              # see Table 2 for more details on VCMTrait

# Simulate the trait under the constructed model
y = simulate(trait_model) # create a single replicate
```

Here X is the matrix of predictors and B is the corresponding matrix of regression coefficients. The **@vc** macro provides a convenient way to specify the variance components of the model. Σ_A is the additive genetic covariance matrix, Σ_E is the environmental covariance matrix, $\otimes$ denotes a Kronecker product and I_n is the $n \times n$ identity matrix.

Table 2.2 uses the same variables definitions as the above code. For each model in Table 2.2 the simulation procedure is also similar to the above code. Julia implements multiple dispatch that allows our simulate function to run the appropriate simulation routine even when we specify dramatically different models.

More details on running `TraitSimulation.jl` under various settings, and additional step-by-step instructions for model specification, can be found in our interactive `Jupyter` notebooks at [13]. `TraitSimulation.jl` provides users with a variety of different ways to specify the simulation model of choice. The following alternative commands to specify the genetic model may be more convenient.

```
# Alternate code for model specification
mean_formula = ["12 + 2.0*sex + 0.005*locus",
                "20 + 2.0*sex + 0.01*locus"]
```

```
trait_model = VCMTrait(mean_formula, DataFrame(X),
                       variance_formula)
```

In this alternative specification, the regression coefficients in B are provided as a 2-element vector of formulas, one for each trait. The matrix of predictors is specified as a DataFrame with column names coordinated with those appearing in the formulas.

Another model specification mechanism provides greater flexibility for users who wish to include many SNPs without having to convert the model genotypes from the compressed SnpArray. Here, the genetic and non-genetic predictors (G,X) and the corresponding regression coefficients (γ,B) are provided separately.

```
# Alternate code for model specification
G = SnpArray("snps_for_simulation.bed", n) # load SNPs
γ = [0.005 0.01] # regression coefficients for SNPs
trait_model = VCMTrait(X, B, G, γ, variance_formula)
```

Users with many variance components may choose not to use the **@vc** macro and instead provide a list of the variance components and variance/covariance matrices:

```
# Alternate code for model specification
G = SnpArray("snps_for_simulation.bed", n) # load SNPs
γ = [0.005 0.01] # regression coefficients for SNPs
Σ = [Σ_A Σ_E] # collect all variance components
V = [V_A V_E] # collect all variance/covariance matrices
trait_model = VCMTrait(X, B, G, γ, Σ, V)
```

2.3 Results

Readers can reproduce our results by accessing the software, documentation, and Jupyter notebooks at:

`https://github.com/OpenMendel/TraitSimulation.jl`

In the two case studies we present below, all the generative models for trait simulation that we employ, univariate and bivariate variance components models and an ordinal multinomial model, could not currently have been built into any other trait simulation package we know. As we describe in the case studies, these are clearly the correct models for simulation given the respective data sets. Thus, `TraitSimulation.jl`'s flexible model specification permits analyses that would not otherwise be available. Step-by-step, interactive Jupyter notebooks that walk the user through these case studies are available at [13]. We begin by describing the statistics behind these power analysis studies and how `TraitSimulation.jl` fits into the software pipeline in concert with other `OpenMendel` modules.

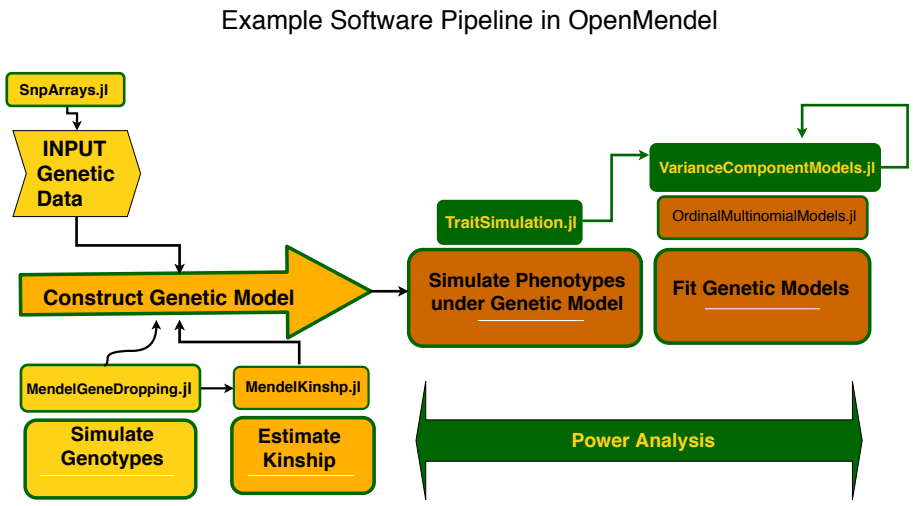

Figure 2.1: `OpenMendel` Pipeline Example: Illustrates how `TraitSimulation.jl` fits within the `OpenMendel` software pipeline to assess the statistical power of different association studies.

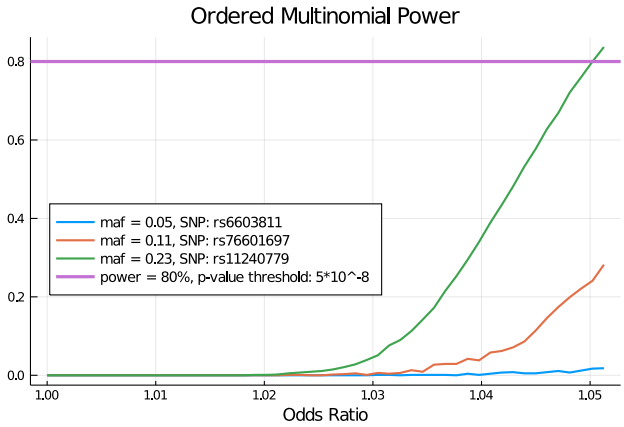

Figure 2.2: Case Study 1: Power under an Ordinal Multinomial Model. This example shows the power to detect a single causal SNP in UK Biobank data with four outcome categories for disease status. Using an ordinal multinomial simulation model and the `OpenMendel` module for ordinal trait regression [11], we assume a single SNP as a fixed effect and control for sex and standardized age. The figure compares analysis results for three SNPs of varying MAF over 1000 simulation replicates each. For each SNP, the graph depicts the power to detect that SNP at significance level $\alpha = 5 \times 10^{-8}$. For each SNP, the effect size varies from 0 to 0.05 in increments of 0.001. On the x-axis, we exponentiate effect sizes to covert to odds ratios. See the text for a detailed description of the model.

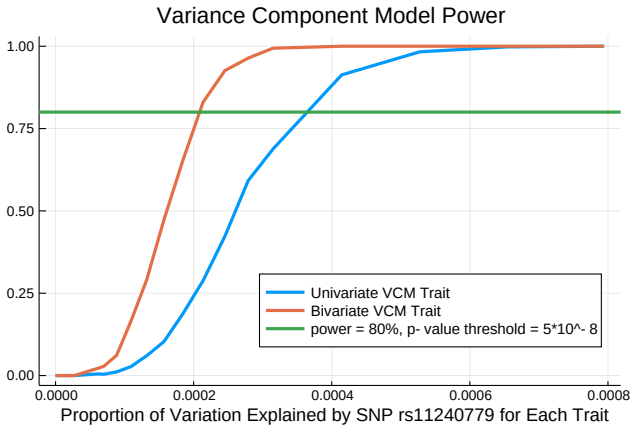

Figure 2.3: Case Study 2: Power under Univariate and Bivariate Variance Components Models. This example shows the power to detect a single causal SNP using both univariate and bivariate variance components simulation models and the `OpenMendel` module for variance components analysis [34]. For each anlysis, each line in the graph depicts the power to detect a SNP with MAF $= 0.23$ using 1000 simulations at significance level $\alpha = 5 \times 10^{-8}$. The SNP effect size varies from 0 to 0.065 in increments of 0.002 in the center range (0.016 – 0.032) and increments of 0.005 in the two end ranges. On the x-axis, we convert the SNP MAF and effect sizes into the proportion of variation explained by the SNP. See the text for a detailed description of the model.

2.3.1 Statistical Power

For a trait Y with predictor matrix $\mathbf{X}$ and genotype vector $\mathbf{G_s}$, we now illustrate how to estimate the power to detect an associated SNP with effect size γ at the pre-specified significance level α. Specifically, we set $\alpha = 5 \times 10^{-8}$ and test the hypothesis

$$\mathbf{H}_0 : \gamma = 0 \qquad \text{versus} \qquad \mathbf{H}_A : \gamma \neq 0$$

in our two subsequent case studies. The user also needs to specify the number of simulation replicates. In the examples presented in this article, we commence by simulating 1000 replicates of an n-vector of phenotypes Y with the specified SNP effect size γ. For each simulated trait vector, we perform a likelihood ratio test of the above hypothesis test and reject the null when the p-value falls below α. The power for the model is estimated as the proportion of the 1000 tests rejecting the null.

2.3.1.1 Case Study 1: Power Analysis for an Ordinal Disease

When modeling complex diseases where a binary phenotype for disease status is suboptimal, an ordered multinomial model is a powerful alternative. Our group recently demonstrated the application of an ordinal multinomial model to assess markers for association to diabetes and hypertension in the UK Biobank data [11]. Ordinal phenotypes were simulated and then fit using one of three analyses models, linear regression, logistic regression and ordered multinomial regression, to assess effects of model misspecification and show increased power under the ordered multinomial model. For the current case study, we determine the power to detect a SNP that influences an ordered categorical phenotype representing the stages of disease progression in the UK Biobank data with n = 185,565 subjects after data cleaning. Specifically, consider a trait y that takes ordered discrete values at one of J = 4 levels:

$$(1) \text{ undiagnosed } < (2) \text{ mild } < (3) \text{ moderate } < (4) \text{ severe } .$$

Under the GLM framework, the cumulative probabilities $\alpha_{ij} = \Pr(y_i \leq j)$ are linked to the linear predictors via the logit link $g(\alpha_{ij}) = \eta = \log\left(\frac{\alpha_{ij}}{1-\alpha_{ij}}\right)$. The link itself is determined by the formula

$$g(\alpha_{ij}) = \theta_j - (\mathbf{X}_i^T \beta + \gamma \mathbf{G}_s), \quad j = 1, \ldots, J-1,$$

where the intercept parameters $\theta_1 \leq \cdots \leq \theta_{J-1}$ enforce the order between the categories and β reflects the effects of the linear predictors under the proportional hazards model. The effect sizes can be interpreted as the expected change of the response variable on an ordered log-odds scale for each unit increase in the predictor. Figure 2.2 shows the resulting power curves for three SNPs with varying MAF.

2.3.1.2 Case Study 2: Power Analysis for Multivariate Continuous Traits

In this case study, we carry out heritability estimation on simulated data with two variance components, one for the additive genetic variance and one for the environmental variance. `TraitSimulation.jl` allows users to simulate multiple traits and more than two variance components by changing a few pertinent commands. For multivariate traits, two theoretical covariance matrices must be substituted for the additive genetic and environmental variances. Here we demonstrate how power calculations for the mixed model scale on a subset of n = 20,000 individuals from the same UK Biobank data used in Case Study 1. For both the univariate model ($d = 1$) and multivariate ($d > 1$) mixed effect model (listed as model type (4) in Table 2.1), we invoke `SnpArrays.jl` to estimate the kinship matrix $\widehat{\Phi}_{GRM}$ via the standard GRM formula

$$\widehat{\Phi}_{GRMij} = \frac{1}{2S} \sum_{k=1}^{S} \frac{(G_{ik} - 2p_k)(G_{jk} - 2p_k)}{2p_k(1-p_k)}.$$

Here i and j are two generic individuals, S is the number of typed SNPs in the data, p_k is the MAF of the k^{th} SNP, and $G_{ik} \in \{0,1,2\}$ is the number of copies of the minor allele at the k^{th} SNP of individual i. Missing genotypes are simplistically imputed on the fly as the most likely genotype given a SNP's MAF. Finally, we make the common assumption that the residual covariance between two relatives

	n = 185,565
$k = 1$	707.8
$k = 20$	14350.2

Table 2.3: For the Ordered Multinomial Model, Power Calculation Runtimes in Seconds. A total of 1000 replications were performed for each combination (k,n) of the number of causal SNPs and the sample size. This was repeated for several SNP effect sizes (see Figure 2.2) and the median runtimes are recorded here.

is well approximated via the additive genetic variance times twice their kinship coefficient. The latter is taken as the corresponding entry of the GRM matrix.

Our univariate and bivariate power calculation results under the variance components model (VCM) framework appear in Figure 2.3. In the univariate model, β and γ represent the non-genetic and genetic regression coefficients, respectively. We assigned 20 different values to the effect size γ of the associated SNP or SNPs during phenotype simulation. At each γ value, for each of 1000 replications, we tested for association using a likelihood ratio test (LRT) with significance level $\alpha = 5 \times 10^{-8}$. Symbolically, the univariate and bivariate models are

$$\mathbf{Y}_{n\times 1} = \mathbf{X}\beta + \mathbf{G}_s\gamma + \mathbf{g} + \varepsilon; \qquad \begin{aligned} \mathbf{g} &\sim N(\mathbf{0}, \sigma_A^2 \times \Phi) \\ \varepsilon &\sim N(\mathbf{0}, \sigma_E^2 \times \mathbf{I_n}) \end{aligned}$$

$$vec(\mathbf{Y}_{n\times d}) = vec(\mathbf{XB} + \mathbf{G}_s\gamma) + \mathbf{g} + \varepsilon; \qquad \begin{aligned} \mathbf{g} &\sim N(\mathbf{0}, \Sigma_A \otimes \Phi) \\ \varepsilon &\sim N(\mathbf{0}, \Sigma_E \otimes \mathbf{I}_n) \end{aligned}$$

Here σ_A^2 and Σ_A are the additive genetic variance and matrix, σ_E^2 and Σ_E are the environmental variance and matrix, Φ is the kinship matrix, and $\mathbf{I_n}$ is the $n \times n$ identity matrix. The multivariate trait model is presented in its vectorized form using the multivariate normal density, where $\mathbf{B}$ and γ are the matrix of regression coefficients for the non-genetic and genetic predictors, respectively. Kronecker products $\otimes$ are required as explained in [17].

		$n = 5{,}000$	$n = 10{,}000$	$n = 20{,}000$
Univariate	$k = 1$	72.4	202.7	815.4
	$k = 20$	1422.6	4122.1	16018.4
Bivariate	$k = 1$	215.5	354.7	978.7
	$k = 20$	4207.8	7007.2	19644.8

Table 2.4: For the Univariate and Bivariate Variance Components Model, Power Calculation Runtimes in Seconds. A total of 1000 replications were performed for each combination (k, n) of the number of causal SNPs and the sample size. This was repeated for several SNP effect sizes (see Figure 2.3) and the median runtimes are recorded here.

2.3.2 Benchmarks

Tables 2.3 and 2.4 record the median total runtimes in seconds over all 1000 replications across k specified SNP predictors for a sample size of n people, for Case Studies 1 and 2, respectively, as reported by the Julia `BenchmarkTools.jl`package. All computer runs were performed on a standard 3.5 GHz Intel i9 CPU with 12 cores; they were run under Linux but we find the operating system has no appreciable effect on runtimes. As mentioned above, these power calculation runtimes are dominated by the post-simulation analyses. Thus, for variance component analyses, the runtimes scale linearly in k, but not in n, as is usual for a variance components statistical analysis. Of course, overall runtimes are linear in the number of replications chosen to perform. However, since each replication is an independent process and our programs can easily be distributed across multiple machines, using even extremely large numbers of replications, for example, for precise type 1 error estimation, is certainly feasible on a computational cluster or in the cloud.

2.4 Conclusions

Genetic epidemiology and computational statistics are inexorably linked. The increasing size and complexity of genetic data drive improvements in algorithm design, and statistical advances push new genetic analyses. To continue this progress, we have introduced `TraitSimulation.jl`, a

software package that employs the Julia language to achieve impressive computational efficiencies and easy coding for a broad range of trait simulation models, including many unavailable in other simulation packages.

Simulation is a vital step in estimating the power of a proposed study to map genetic influences. To obtain the best power estimates, one must exploit all available study subjects (unrelated, sibships, parent-offspring pairs, and extended pedigrees), impute realistic genotypes (based on ethnically correct MAF, linkage disequilibrium (LD), and possibly recombination events), incorporate pertinent non-genetic predictors, and critically, simulate realistic trait values.

For example, if one is planning a family-based study and wanted to do a power analysis before collecting any data, then one would start with a collection of pedigree structures, including possibly singletons, that mimicked to the best of one's knowledge the potential sample collection. At the founders of each pedigree one would want to simulate the genetic data using ethnic-specific allelic frequencies based on the admixture of the target population. The correct LD structure should also be maintained within these founder genomes. One way to accomplish this is to find a real genotyping or sequencing study, for example, the International Genome Sample Resource (IGSR)[12], that includes subjects in the specified ethnicities, and use the real genomes of unrelated individuals as the data for the founders of the pedigrees. Then use gene-dropping software, for example, from the `OpenMendel` suite, and the real human recombination map to mimic recombination events that would occur as genomes are passed from parent to child through the pedigrees. The result will be simulated but realistic genetic data for all individuals in all pedigrees, because the data reflects the appropriate allelic frequencies, LD patterns, recombination map, and relationship structure. Our `TraitSimulation.jl` package can then use this data and whichever trait model you wish to study to repeatedly generate trait values. Finally, each set of simulated data would be subject to the statistical analyses that constitutes the power analysis.

The model generality, ease of use, and speed of `TraitSimulation.jl`, and indeed `OpenMendel` as a whole, promote the agenda of modern epidemiology. `TraitSimulation.jl`'s wide range of generating models improves model realism and therefore power estimation. This generality allows

statistical analysis to escape the straitjacket of the Gaussian assumption by allowing case/control and ordinal disease models, and more profoundly, any GLM or GLMM structure. Our choice of the Julia computer language makes it straightforward to code software and for users to adapt existing code to fit their modeling needs. Julia enhances the speed, flexibility, and overall ease-of-use of `TraitSimulation.jl`. Julia's speed stems from its just-in-time compiler, thorough use of parallelization, and its promotion of bit-wise linear algebra operations.

`TraitSimulation.jl` is part of the `OpenMendel` family of Julia packages [34]. `OpenMendel` provides an integrated suite of genetic analysis tools that rely on the standard data structure provided by `SnpArrays.jl`. `TraitSimulation.jl` can access other downstream analysis packages in estimating parameters and the power of new statistical tests. However, such pipeline strategies introduce extra layers of complexity and ultimately hamper analysis reproducibility. All of `OpenMendel`'s packages are fast, memory efficient, and user- and developer-friendly. The open-source nature of `OpenMendel` encourages other statisticians to extend its code base. In adding `TraitSimulation.jl` to the `OpenMendel` family, we enable trait simulation within an integrated robust analysis pipeline. In our view, `OpenMendel` represents a unique and unified state-of-the-art environment for statistical genetics. We ask for your feedback and the help of the entire genetics community in perfecting `OpenMendel`. It or something very similar will be necessary as we face ever more massive and complex modern data sets.

2.5 Availability of source code

Project name: TraitSimulation

Project home page:

https://github.com/OpenMendel/TraitSimulation.jl

Operating systems: Mac OS, Linux, Windows

Programming language: Julia 1.0, 1.2

License: MIT

The code to generate simulated data, as well as their subsequent power analyses, are available in our github repository. Notably, `TraitSimulation.jl` interfaces with the `OpenMendel` [35] packages `SnpArrays.jl` [33], `OrdinalMultinomialModels.jl` [11], `VarianceComponentModels.jl` [34], and JuliaStats's packages `Distribution.jl` [4] and `GLM.jl` [22]. We used the Julia package `Plots.jl` to obtain all our graphs.

CHAPTER 3

A Flexible Quasi-Copula Distribution for Statistical Modeling: QuasiCopula.jl

3.1 Motivation

The analysis of correlated data is stymied by the lack of flexible multivariate distributions with fixed margins. Once one ventures beyond the confines of multivariate Gaussian distributions, analysis choices are limited. [21] launched the highly influential method of generalized estimating equations (GEEs). This advance allows generalized linear models (GLMs) to accommodate the correlated traits encountered in panel and longitudinal data and effectively broke the stranglehold of Gaussian distributions in analysis. The competing method of statistical copulas introduced earlier by Sklar is motivated by the same consideration [26]. Finally, generalized linear mixed models (GLMMs) [6, 29] attacked the same problem. GLMMs are effective tools for modeling overdispersion and capturing the correlations of multivariate discrete data.

However, none of these three modeling approaches is a panacea. GEEs lack a well-defined likelihood, and estimation searches can fail to converge. For copula models, likelihoods exist, but are unwieldy, particularly for discrete outcomes. Copula calculations scale extremely poorly in high dimensions. Computing with GLMMs is problematic since their densities have no closed form and require evaluation of multidimensional integrals. Gaussian quadratures scale exponentially in the dimension of the parameter space. Markov Chain Monte Carlo (MCMC) can be harnessed in Bayesian versions of GLMMs, but even MCMC can be costly. For these reasons alone, it is worth pursuing alternative modeling approaches.

This brings us to an obscure paper by the Japanese mathematical statistician Tonda. Working within the framework of Gaussian copulas [27] and generalized linear models, Tonda introduces a device for relaxing independence assumptions while preserving computable likelihoods [28]. He succeeds brilliantly except for the presence of an annoying constraint on the parameter space of the new distribution class. The fact that his construction perturbs marginal distributions is forgivable. The current paper has several purposes. First, by adopting a slightly different working definition, we show how to extend his construction to lift the awkward parameter constraint. Our new definition allows explicit calculation of (a) moments, (b) marginal and conditional distributions, and (c) the score and observed information of the loglikelihood and allows (d) generation of random deviates. Tonda tackles item (a), omits items (b) and (c), and mentions item (d) only in passing. For maximum likelihood estimation (MLE), he relies on a non-standard derivative-free algorithm [23] that scales poorly in high dimensions. We present two gradient-based algorithms designed for high-dimensional MLEs. The first is a block ascent algorithm that updates fixed effects by Newton's method and updates variance components by a minorization-maximization (MM) algorithm. The second is a standard quasi-Newton algorithm that updates fixed effects and variance components jointly.

In contrast to other multivariate outcome models, our loglikelihoods contain no determinants or matrix inverses. These features resolve computational bottlenecks in parameter estimation. We advocate gradient based estimation methods that avoid computationally intensive second derivatives. Approximate Hessians can be computed after estimation to provide asymptotic standard errors and confidence intervals. The range of potential applications of our quasi-copula model is enormous. Panel, longitudinal, time series, and all of GLM modeling stand to benefit. In addition to relaxing independence assumptions, our models offer a simple way to capture over-dispersion. Our simulation studies and real data examples highlight not only the virtues of the quasi-copula model but also its limitations. For reasons to be explained, we find that the model reflects reality best when the size of the independent sampling units is low or the correlations between responses within a unit are small.

We first address the inspiration from Tonda's paper to illustrate the GLMM approximation pro-

cedure and address why we call the model a "Quasi-Copula" model.

3.1.1 GLMM Framework

Let $\mathbf{Y} = (Y_1, ..., Y_d)$ be a d-dimensional multivariate random vector response, $\theta = (\theta_1, ..., \theta_d)$ denote a d-dimensional unknown parameter vector. Recall that for a GLM outcome, the distribution of Y is from the exponential family of distributions of the form

$$f_{exp}(Y|\theta) = \prod_{j=1}^{d} \exp\left[\frac{Y_j\theta_j - b_j(\theta_j)}{\phi_j} + c_j(y_j, \phi_j)\right].$$

where if we use the canonical link function, $\theta = g(\mu) = \eta + \mathbf{z}$ where η is the fixed systematic component and $\mathbf{z} = (z_1, ..., z_d) \sim h(\mathbf{z}|\Sigma)$ is the random effects. Typically we assume the random effects follow a multivariate normal with mean 0 and covariance matrix Σ.

$$h(\mathbf{z}|\Sigma) \sim N_d(0, \Sigma)$$

Then the likelihood is

$$f(Y|\eta, \Sigma) = \int_{\mathbb{R}^d} f_{exp}(Y|\theta) h(\mathbf{z} \mid \Sigma)\, dz$$

3.1.2 Tonda's Framework

In order to avoid calculating the multi-dimensional integral, Tonda derives an approximation of the GLMM density based on a Taylor Series expansion about the mean of the random effect, $\mathbf{z_i} = 0$, up to the second order. Detailed derivations are found in the text [28] and in the supplemental material at the end of this chapter.

$$\begin{aligned} f(\mathbf{Y}; \tilde{\eta}, \Sigma) &\approx E_{\mathbf{z}}\left[f_{exp}(\mathbf{Y}|\eta + \mathbf{z}) + \left(\frac{\partial f_{exp}(\mathbf{Y}|\eta + \mathbf{z})}{\partial \mathbf{z}}|_{\mathbf{z}=0}\right)^t \mathbf{z} + \frac{1}{2}\mathbf{z}^t\left(\frac{\partial^2 f_{exp}(\mathbf{Y}|\eta + \mathbf{z_i})}{\partial \mathbf{z}^2}|_{\mathbf{z}=0}\right)^t \mathbf{z}\right] \\ &= \left[1 + \frac{1}{2}tr(\mathbf{W}\Sigma)\right] f_{exp}(\mathbf{Y}|\eta) \end{aligned}$$

where $\mathbf{W} = (w_{ab})$ is a $p \times p$ matrix whose elements are functions of $\mathbf{Y}$ given by

$$\begin{aligned} w_{aa} &= \frac{\partial^2}{\partial z_a^2} f_{exp}(\mathbf{Y}|\eta + \mathbf{z})|_{\mathbf{z}=0} \\ w_{ab} &= \frac{\partial^2}{\partial z_a \partial z_b} f_{exp}(\mathbf{Y}|\eta + \mathbf{z})|_{\mathbf{z}=0} \end{aligned}$$

Tonda notes in Theorem 3.1 of the paper is only a proper pdf if Σ is a positive definite matrix and the following condition holds

$$\sum_{j=1}^{d} \frac{1}{\phi_j} v_j(\mu_j)\sigma_{jj} \leq 2.$$

Recall the form of the copula loglikelihood for continuous distributions with parametric models $F_{Y_1}(y_1|\theta_1), ..., F_{Y_d}(y_d|\theta_d)$ as the marginal CDFs.

$$f_{\mathbf{Y}}(\mathbf{y}) = f_{\mathbf{Y}}(y_1, ..., y_d) = c_{\mathbf{Y}}(F_{Y_1}(y_1), ..., F_{Y_d}(y_d)) \prod_{j=1}^{d} f_{Y_j}(y_j)$$

We call this denisty a "Quasi-Copula" model because it takes a similar form as the general copula density above. The density uses a scalar valued function of $\mathbf{Y}$ as the "copula" to induce covariation between the independent exponential family densities. We note the main difference is that this approximation is more general than the traditional copula framework in that it does not depend on CDF's or inverse quantiles of the marginal distributions.

3.2 Definitions

Consider d independent random variables $X_1, \ldots, X_d$ with densities $f_i(x_i)$ relative to measures α_i, with means μ_i, variances σ_i^2, third central moments c_{i3}, and fourth central moments c_{i4}. Let $\mathbf{\Gamma} = (\gamma_{ij})$ be an $d \times d$ positive semidefinite matrix, and α be the product measure $\alpha_1 \times \cdots \times \alpha_d$. Inspired by [28], we let $\mathbf{D}$ be the diagonal matrix with ith diagonal entry σ_i and consider the nonnegative function

$$1 + \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^t \mathbf{D}^{-1} \mathbf{\Gamma} \mathbf{D}^{-1} (\mathbf{x} - \boldsymbol{\mu}).$$

Its average value is

$$\begin{aligned} & \int \prod_{i=1}^{d} f_i(x_i) \Big[1 + \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^t \mathbf{D}^{-1} \boldsymbol{\Gamma} \mathbf{D}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \Big] d\alpha(\mathbf{x}) \\ = \; & 1 + \frac{1}{2} \sum_i \sum_j E \Big[\frac{(x_i - \mu_i)(x_j - \mu_j)}{\sigma_i \sigma_j} \Big] \gamma_{ij} \\ = \; & 1 + \frac{1}{2} \sum_i \gamma_{ii}. \end{aligned}$$

It follows that the function

$$g(\mathbf{x}) \; = \; \Big[1 + \frac{1}{2} \operatorname{tr}(\boldsymbol{\Gamma}) \Big]^{-1} \prod_{i=1}^{d} f_i(x_i) \Big[1 + \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^t \mathbf{D}^{-1} \boldsymbol{\Gamma} \mathbf{D}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \Big] \tag{3.1}$$

is a probability density with respect to the measure α. The virtue of the density is that it overcomes the independence restriction and steers the sample matrix of the residuals toward the target covariance matrix $\boldsymbol{\Gamma}$. Note that $g(\mathbf{x})$ is technically not a copula since it fails to preserve the marginal distributions $f_i(x_i)$. For instance, we will see later that $g(\mathbf{x})$ tends to inflate marginal variances.

Tonda replaces the ith diagonal entry of $\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^t \mathbf{D}^{-1} \boldsymbol{\Gamma} \mathbf{D}^{-1} (\mathbf{x} - \boldsymbol{\mu})$ by $\frac{1}{2}[\frac{(x_i - \mu_i)^2}{\sigma_i^2} - 1]\gamma_{ii}$. This yields

$$\int \prod_{i=1}^{m} f_i(x_i) \Big[1 + \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^t \mathbf{D}^{-1} \boldsymbol{\Gamma} \mathbf{D}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \Big] d\alpha(\mathbf{x}) \;\; = \;\; 1 + \frac{1}{2} \sum_i (1 - 1) \gamma_{ii} \;\; = \;\; 1.$$

As long as $1 - \frac{1}{2} \sum_i \gamma_{ii} \geq 0$, we have a proper nonnegative density, and no normalization is necessary. Unfortunately, this sufficient condition for nonnegativity is restrictive and awkward to maintain in maximum likelihood estimation.

3.3 Moments

Let $\mathbf{Y} = (Y_1, \ldots, Y_d)^t$ be a random vector distributed as $g(\mathbf{x})$. To calculate the mean of Y_k, note that our independence assumption implies

$$\begin{aligned} & \int (x_k - \mu_k) g(\mathbf{x}) \alpha(\mathbf{x}) \\ = & \left[1 + \frac{1}{2}\mathrm{tr}(\mathbf{\Gamma})\right]^{-1} \frac{1}{2} \sum_i \sum_j \mathbb{E}\left[(x_k - \mu_k) \frac{(x_i - \mu_i)(x_j - \mu_j)}{\sigma_i \sigma_j}\right] \gamma_{ij} \\ = & \left[1 + \frac{1}{2}\mathrm{tr}(\mathbf{\Gamma})\right]^{-1} \frac{c_{k3}\gamma_{kk}}{2\sigma_k^2}. \end{aligned}$$

Hence, if κ_{k3} is the skewness of X_k, then

$$\begin{aligned} \mathbb{E}(Y_k) &= \mu_k + \left[1 + \frac{1}{2}\mathrm{tr}(\mathbf{\Gamma})\right]^{-1} \frac{c_{k3}\gamma_{kk}}{2\sigma_k^2} \\ &= \mu_k + \left[1 + \frac{1}{2}\mathrm{tr}(\mathbf{\Gamma})\right]^{-1} \frac{\sigma_k \kappa_{k3} \gamma_{kk}}{2} \\ &= \mu_k + \frac{\sigma_k \kappa_{k3} \gamma_{kk}}{2} + O(\|\mathbf{\Gamma}\|^2) \end{aligned}$$

for any matrix norm $\|\mathbf{\Gamma}\|$. The mean $E(Y_k)$ is close to μ_k when the diagonal entries of $\mathbf{\Gamma}$ and, hence $|\mathbf{\Gamma}\|$ itself, are small.

To calculate the covariance matrix of $\mathbf{Y}$, note that

$$\begin{aligned} & \int (x_k - \mu_k)(x_l - \mu_l) g(\mathbf{x}) d\alpha(\mathbf{x}) \\ = & \left[1 + \frac{1}{2}\mathrm{tr}(\mathbf{\Gamma})\right]^{-1} 1_{\{k=l\}} \sigma_k^2 + \left[1 + \frac{1}{2}\mathrm{tr}(\mathbf{\Gamma})\right]^{-1} \\ & \times \frac{1}{2} \sum_i \sum_j E\left[(x_k - \mu_k)(x_l - \mu_l) \frac{(x_i - \mu_i)(x_j - \mu_j)}{\sigma_i \sigma_j}\right] \gamma_{ij}. \end{aligned}$$

The indicated expectations relative to $\prod_{i=1}^{d} f_i(x_i)$ reduce to

$$\begin{aligned}
&\mathbb{E}[(x_k-\mu_k)(x_l-\mu_l)(x_i-\mu_i)(x_j-\mu_j)] \\
&= \begin{cases} c_{k4} & k=l=i=j \\ \sigma_k^2\sigma_i^2 & k=l\neq i=j \\ \sigma_k^2\sigma_l^2 & k=i\neq l=j \\ \sigma_k^2\sigma_l^2 & k=j\neq l=i \\ 0 & \text{otherwise .} \end{cases}
\end{aligned}$$

When $k=l$ and κ_{k4} is the kurtosis of X_k,

$$\begin{aligned}
&\int (x_k-\mu_k)^2 g(\mathbf{x})\,d\alpha(\mathbf{x}) \\
&= \left[1+\frac{1}{2}\mathrm{tr}(\boldsymbol{\Gamma})\right]^{-1}\left[\sigma_k^2+\frac{1}{2}\frac{c_{k4}\gamma_{kk}}{\sigma_k^2}+\frac{1}{2}\sigma_k^2\sum_{i\neq k}\gamma_{ii}\right] \\
&= \left[1+\frac{1}{2}\mathrm{tr}(\boldsymbol{\Gamma})\right]^{-1}\sigma_k^2\left[1+\frac{\kappa_{k4}\gamma_{kk}}{2}+\frac{1}{2}\sum_{i\neq k}\gamma_{ii}\right] \\
&= \sigma_k^2+\frac{\sigma_k^2\kappa_{k4}\gamma_{kk}}{2}+\frac{\sigma_k^2}{2}\sum_{i\neq k}\gamma_{ii}+O(\|\boldsymbol{\Gamma}\|^2) \\
&= \sigma_k^2\left[1+\frac{(\kappa_{k4}-1)\gamma_{kk}}{2}+\frac{1}{2}\sum_{i}\gamma_{ii}\right]+O(\|\boldsymbol{\Gamma}\|^2) \\
&= \sigma_k^2\left[1+\frac{(\kappa_{k4}-1)\gamma_{kk}}{2}\right]+O(\|\boldsymbol{\Gamma}\|^2).
\end{aligned}$$

Because $[E(Y_k-\mu_k)]^2 = O(\|\boldsymbol{\Gamma}\|^2)$, we find that

$$\begin{aligned}
Var(Y_k) &= \mathbb{E}[(Y_k-\mu_k)^2]-[\mathbb{E}(Y_k-\mu_k)]^2 \\
&= \sigma_k^2\left[1+\frac{(\kappa_{k4}-1)\gamma_{kk}}{2}\right]+O(\|\boldsymbol{\Gamma}\|^2).
\end{aligned}$$

Because the kurtosis $\kappa_{k4}\geq 1$, the multiplier $\kappa_{k4}-1$ of γ_{kk} is nonnegative, and the variance is inflated for $\|\boldsymbol{\Gamma}\|$ small.

When $k \neq l$,

$$\begin{aligned}\int (x_k-\mu_k)(x_l-\mu_l)g(\mathbf{x})d\alpha(\mathbf{x}) &= \left[1+\frac{1}{2}\mathrm{tr}(\mathbf{\Gamma})\right]^{-1}\frac{1}{2}2\sigma_k\sigma_l\gamma_{kl} \\ &= \sigma_k\sigma_l\gamma_{kl}+O(\|\mathbf{\Gamma}\|^2).\end{aligned}$$

Hence, the covariance and correlation satisfy

$$\begin{aligned}Cov(Y_k,Y_l) &= Cov(Y_k-\mu_k,Y_l-\mu_l) \\ &= \mathbb{E}[(Y_k-\mu_k)(Y_l-\mu_l)]-\mathbb{E}(Y_k-\mu_k)\,\mathbb{E}(Y_l-\mu_l) \\ &= \sigma_k\sigma_l\gamma_{kl}+O(\|\mathbf{\Gamma}\|^2) \\ \mathrm{Corr}(Y_k,Y_l) &= \frac{\gamma_{kl}}{\sqrt{1+\frac{(\kappa_{k4}-1)\gamma_{kk}}{2}+O(|\mathbf{\Gamma}\|^2)}\sqrt{1+\frac{(\kappa_{l4}-1)\gamma_{ll}}{2}+O(|\mathbf{\Gamma}\|^2)}}.\end{aligned}$$

As a check, the quantities $\mathbb{E}(Y_k)$, $Var(Y_k)$, and $Cov(Y_k,Y_l)$ reduce to the correct values μ_k, σ_k^2, and 0, respectively, when $\mathbf{\Gamma}=\mathbf{0}$.

3.4 Marginal and Conditional Distributions

Let S be a subset of $\{1,\ldots,d\}$ with complement T. To simplify notation, suppose $S=\{1,2,\ldots,s\}$. Now write

$$\mathbf{Y} = \begin{pmatrix}\mathbf{Y}_S\\ \mathbf{Y}_T\end{pmatrix},\ \mathbf{r} = \begin{pmatrix}\mathbf{r}_S\\ \mathbf{r}_T\end{pmatrix},\ \mathbf{\Gamma} = \begin{pmatrix}\mathbf{\Gamma}_S & \mathbf{\Gamma}_{ST}\\ \mathbf{\Gamma}_{ST}^t & \mathbf{\Gamma}_T\end{pmatrix},\ \alpha = \alpha_S\times\alpha_T,$$

where $\mathbf{r}$ is the vector $\mathbf{D}^{-1}(\mathbf{Y}-\boldsymbol{\mu})$ of standardized residuals. The marginal density of $\mathbf{Y}_S$ is

$$\begin{aligned}&\left[1+\frac{1}{2}\mathrm{tr}(\mathbf{\Gamma})\right]^{-1}\prod_{i\in S}f_i(y_i)\int\prod_{i\in T}f_i(y_i)\left[1+\frac{1}{2}\mathbf{r}^t\mathbf{\Gamma}\mathbf{r}\right]d\alpha_T(\mathbf{y}_T) \\ =\ &\left[1+\frac{1}{2}\mathrm{tr}(\mathbf{\Gamma})\right]^{-1}\prod_{i\in S}f_i(y_i)\left[1+\frac{1}{2}\mathbf{r}_S^t\mathbf{\Gamma}_S\mathbf{r}_S+\frac{1}{2}\mathrm{tr}(\mathbf{\Gamma}_T)\right].\end{aligned}$$

To derive the conditional density of $\mathbf{Y}_S$ given by $\mathbf{Y}_T$, we divide the joint density by the marginal density of $\mathbf{Y}_T$. This action produces the conditional density

$$d_S \prod_{i \in S} f_i(y_i) \left[1 + \frac{1}{2} \mathbf{r}^t \mathbf{\Gamma} \mathbf{r} \right]$$

with normalizing constant $d_S = \left[1 + \frac{1}{2} \mathbf{r}_T^t \mathbf{\Gamma}_T \mathbf{r}_T + \frac{1}{2} \mathrm{tr}(\mathbf{\Gamma}_S) \right]^{-1}$. From this density, our well-rehearsed arguments lead to the conditional mean

$$\mathbb{E}(Y_k \mid \mathbf{Y}_T) = \mu_k + d_S \left[\frac{c_{k3} \gamma_{kk}}{2\sigma_k^2} + \frac{1}{\sigma_k} \sum_{j \in T} r_j \gamma_{jk} \right] \;=\; \mu_k + \frac{c_{k3} \gamma_{kk}}{2\sigma_k^2} + O(\|\mathbf{\Gamma}\|^2)$$

for $k \in S$. The corresponding conditional variance is

$$Var(Y_k \mid \mathbf{Y}_T) \;=\; \sigma_k^2 + \frac{1}{2} \left(\frac{c_{k4}}{\sigma_k^2} - \sigma_k^2 \right) \gamma_{kk} + \sum_{j \in T} \frac{c_{k3} r_j \gamma_{kj}}{\sigma_k} + O(\|\mathbf{\Gamma}\|^2).$$

and the corresponding conditional covariances are

$$Cov(Y_k, Y_l \mid \mathbf{Y}_T) \;=\; \sigma_k \sigma_l \gamma_{kl} + O(\|\mathbf{\Gamma}\|\|^2)$$

for $k \in S$, $l \in S$, and $k \neq l$. It is noteworthy that to order $O(\|\mathbf{\Gamma}\|^2)$, the conditional and marginal means agree, and the conditional and marginal covariances agree.

3.5 Generation of Random Deviates

To generate a random vector from the density (3.1), we first sample Y_1 from its marginal density

$$\left[1 + \frac{1}{2} \mathrm{tr}(\mathbf{\Gamma}) \right]^{-1} f_1(y_1) \left(1 + \frac{\gamma_{11}}{2} r_1^2 + \frac{1}{2} \sum_{j=2}^{d} \gamma_{jj} \right),$$

and then sample the subsequent components Y_i from their conditional distributions, $Y_i \mid Y_1, \ldots, Y_{i-1}$ for all $i \in [1,d]$. If we denote the set $\{1,\ldots,i-1\}$ by $[i-1]$, then the conditional density of y_i given the previous components is

$$d_{[i-1]}^{-1} f_i(y_i)\Big[d_{[i-1]} + r_i \sum_{j=1}^{i-1} r_j \gamma_{ij} + \frac{\gamma_{ii}}{2}(r_i^2 - 1)\Big],$$

where $d_{[i-1]} = 1 + \frac{1}{2}\mathbf{r}_{[i-1]}^t \mathbf{\Gamma}_{[i-1]} \mathbf{r}_{[i-1]} + \frac{1}{2}\sum_{j=i}^{d} \gamma_{jj}$.

When the densities $f_i(y_i)$ are discrete, each stage of sampling is straightforward. Consider any random variable Z with nonnegative integer values, discrete density $p_i = \Pr(Z = i)$, and mean ν. The inverse method of random sampling reduces to a sequence of comparisons. We partition the interval $[0,1]$ into subintervals with the ith subinterval of length p_i. To sample Z, we draw a uniform random deviate U from $[0,1]$ and return the deviate j determined by the conditions $\sum_{i=1}^{j-1} p_i \leq U < \sum_{i=1}^{j} p_i$. The process is most efficient when the largest p_i occur first. This suggests that we let k denote the least integer $\lfloor \nu \rfloor$ and rearrange the probabilities in the order $p_k, p_{k+1}, p_{k-1}, p_{k+2}, p_{k-2}, \ldots$ This tactic is apt put most of the probability mass first and render sampling efficient.

When the densities $f_i(y_i)$ are continuous, each stage of sampling is probably best performed by inverse transform sampling. This requires calculating distribution functions and forming their inverses, either analytically or by Newton's method. The required distribution functions assume the form

$$\int_{-\infty}^{x} f(y)[a_0 + a_1(y-\mu) + a_2(y-\mu)^2]\,dy \;=\; \int_{-\infty}^{x} f(y)[b_0 + b_1 y + b_2 y^2]\,dy.$$

The integrals $\int_{-\infty}^{x} f(y) y^j\,dy$ are available as special functions for Gaussian, beta, and gamma densities $f(y)$. For instance, if $\phi(y) = \frac{1}{2\pi} e^{-y^2/2}$ is the standard normal density and $\Phi(x)$ is the standard normal distribution, then

$$\int_{-\infty}^{x} y\phi(y)\,dy \;=\; -\phi(x) \text{ and } \int_{-\infty}^{x} y^2\phi(y)\,dy = \Phi(x) - x\phi(x).$$

To avoid overburdening the text with classical mathematics, we omit further details. Additional derivations can be found in the supplemental material.

3.6 Parameter Estimation

3.6.1 Mean Components

Consider n independent realizations $\mathbf{y}_i$ from the quasi-copula density (3.1). Each of these may be of a different dimension d_i and possess a different mean vector $\boldsymbol{\mu}_i(\boldsymbol{\beta})$, covariance matrice $\boldsymbol{\Gamma}_i(\boldsymbol{\theta}) = [\gamma_{ijk}(\boldsymbol{\theta})]$, and component densities $f_{ij}(y_{ij} \mid \boldsymbol{\beta})$. If $\mathbf{r}_i(\boldsymbol{\beta})$ denotes the vector $\mathbf{D}_i^{-1}(\mathbf{y}_i - \boldsymbol{\mu}_i)$ of standardized residuals for sampling unit i, then the loglikelihood of the sample is

$$\begin{aligned} \mathcal{L}(\boldsymbol{\beta},\boldsymbol{\theta}) &= -\sum_{i=1}^{n} \ln\left[1+\frac{1}{2}\operatorname{tr}(\boldsymbol{\Gamma}_i(\boldsymbol{\theta}))\right] + \sum_{i=1}^{n}\sum_{j=1}^{d_i} \ln f_{ij}(y_{ij} \mid \boldsymbol{\beta}) \\ &\quad + \sum_{i=1}^{n} \ln\left\{1+\frac{1}{2}\mathbf{r}_i(\boldsymbol{\beta})^t\boldsymbol{\Gamma}_i(\boldsymbol{\theta})\mathbf{r}_i(\boldsymbol{\beta})\right\}. \end{aligned}$$

The score (gradient of the loglikelihood) with respect to $\boldsymbol{\beta}$ is clearly

$$\nabla_{\boldsymbol{\beta}}\mathcal{L}(\boldsymbol{\beta},\boldsymbol{\theta}) = \sum_{i=1}^{n}\sum_{j=1}^{d_i} \nabla \ln f_{ij}(y_{ij} \mid \boldsymbol{\beta}) + \sum_{i=1}^{n} \frac{\nabla \mathbf{r}_i(\boldsymbol{\beta})^t\boldsymbol{\Gamma}_i(\boldsymbol{\theta})\mathbf{r}_i(\boldsymbol{\beta})}{1+\frac{1}{2}\mathbf{r}_i(\boldsymbol{\beta})^t\boldsymbol{\Gamma}_i(\boldsymbol{\theta})\mathbf{r}_i(\boldsymbol{\beta})},$$

where $\nabla \mathbf{r}_i(\boldsymbol{\beta})^t = d\mathbf{r}_i(\boldsymbol{\beta})$ is the differential (Jacobi matrix) of the vector $\mathbf{r}_i(\boldsymbol{\beta})$. An easy calculation shows that $\nabla \mathbf{r}_i(\boldsymbol{\beta})$ has entries

$$\nabla r_{ij}(\boldsymbol{\beta}) = -\frac{1}{\sigma_{ij}(\boldsymbol{\beta})}\nabla\mu_{ij}(\boldsymbol{\beta}) - \frac{1}{2}\frac{y_{ij}-\mu_{ij}(\boldsymbol{\beta})}{\sigma_{ij}^3(\boldsymbol{\beta})}\nabla\sigma_{ij}^2(\boldsymbol{\beta}).$$

In searching the likelihood surface, it is best to approximate the observed information by a positive definite matrix. This suggests replacing $-d^2 \ln f_{ij}(y_{ij} \mid \boldsymbol{\beta})$ by the expected information matrix $\mathbf{J}_{ij}(\boldsymbol{\beta})$ under the independence model and dropping indefinite matrices in the exact Hessian. These

steps give the approximate Hessian

$$d^2_{\boldsymbol{\beta}}\mathcal{L} \;\approx\; -\sum_{i=1}^{n}\sum_{j=1}^{d_i}\mathbf{J}_{ij}(\boldsymbol{\beta}) - \sum_{i=1}^{n}\frac{[\nabla\mathbf{r}_i(\boldsymbol{\beta})\boldsymbol{\Gamma}_i(\boldsymbol{\theta})\mathbf{r}_i(\boldsymbol{\beta})][\nabla\mathbf{r}_i(\boldsymbol{\beta})\boldsymbol{\Gamma}_i(\boldsymbol{\theta})\mathbf{r}_i(\boldsymbol{\beta})]^t}{\left[1+\frac{1}{2}\mathbf{r}_i(\boldsymbol{\beta})^t\boldsymbol{\Gamma}_i\mathbf{r}_i(\boldsymbol{\beta})\right]^2},$$

which is clearly negative semidefinite. As partial justification for this approximation, we expect residuals to be small on average. The score and approximate Hessian provide the ingredients for an approximate Newton's method for improving $\boldsymbol{\beta}$.

3.6.2 Structured Covariance

Maximization of the loglikelihood also involves finding optimal values for the covariance parameters $\boldsymbol{\theta}$ determining the structured covariance matrices $\boldsymbol{\Gamma}_i$. Assuming there are no shared mean and covariance parameters, the relevant part of the loglikelihood is

$$-\sum_{i=1}^{n}\ln\left[1+\frac{1}{2}\mathrm{tr}(\boldsymbol{\Gamma}_i(\boldsymbol{\theta}))\right]+\sum_{i=1}^{n}\ln\left[1+\frac{1}{2}\mathbf{r}_i(\boldsymbol{\beta})^t\boldsymbol{\Gamma}_i(\boldsymbol{\theta})\mathbf{r}_i(\boldsymbol{\beta})\right].$$

To simplify estimation of $\boldsymbol{\Gamma}_i$, we investigate just three covariance scenarios, namely, an autoregressive AR(1) model, a compound symmetric (CS) model, and a variance components (VC) model. Under the AR(1) and CS models, $\boldsymbol{\Gamma}_i(\boldsymbol{\theta})$ is parameterized by $\boldsymbol{\theta}=(\sigma^2,\rho)$, a total variance σ^2 and a correlation ρ. For the AR(1) model this leads to the representation

$$\begin{aligned}\boldsymbol{\Gamma}_i(\boldsymbol{\theta}) &= \sigma^2\times\begin{bmatrix} 1 & \rho & \rho^2 & \rho^3 & \ldots & \rho^{d_i-1}\\ \rho & 1 & \rho & \rho^2 & \ldots & \\ & & \ldots & & & \\ & & \ldots & \rho & 1 & \rho \\ \rho^{d_i-1} & \rho^{d_i-2} & \ldots & \rho^2 & \rho & 1\end{bmatrix}\\ &= \sigma^2\times\mathbf{V}_i(\rho).\end{aligned}$$

For the CS model this leads to the representation

$$\begin{aligned}\boldsymbol{\Gamma}_i &= \sigma^2 \times \left[\rho \mathbf{1}_{d_i}\mathbf{1}_{d_i}^t + (1-\rho)I_{d_i}\right] \\ &= \sigma^2 \times \mathbf{V}_i(\rho)\end{aligned}$$

The relevant part of the loglikelihood can be rewritten as

$$f(\sigma^2,\rho) = -\sum_{i=1}^n \ln\left[1+\frac{d_i\sigma^2}{2}\right] + \sum_{i=1}^n \ln\left[1+\frac{\sigma^2}{2}\mathbf{r}_i(\boldsymbol{\beta})^t\mathbf{V}_i(\rho)\mathbf{r}_i(\boldsymbol{\beta})\right]$$

A typical variance components problem depends on the decomposition $\boldsymbol{\Gamma}_i(\boldsymbol{\theta}) = \sum_{k=1}^m \theta_k \Omega_{ik}$ of $\boldsymbol{\Gamma}_i$ into a linear combination of known covariance matrices $\Omega_{ik} = (\omega_{ikjl})$ against unknown nonnegative variance components θ_k arranged in a vector $\boldsymbol{\theta} = (\theta_k)$. Now the relevant part of the loglikelihood amounts to

$$f(\boldsymbol{\theta}) = \sum_{i=1}^n \ln(1+\boldsymbol{\theta}^t\mathbf{b}_i) - \sum_{i=1}^n \ln(1+\boldsymbol{\theta}^t\mathbf{c}_i),$$

where the vectors $\mathbf{b}_i$ and $\mathbf{c}_i$ have the nonnegative components

$$b_{ik} = \frac{1}{2}\mathbf{r}_i(\boldsymbol{\beta})^t\Omega_{ik}\mathbf{r}_i(\boldsymbol{\beta}) \quad \text{and} \quad c_{ik} = \frac{1}{2}\operatorname{tr}(\Omega_{ik}).$$

Derivation of the scores $\nabla_\theta \mathcal{L}$ and approximate observed information matrices $-d_\theta^2\mathcal{L}$ for the AR(1), CS and VC models is relegated to the Supplemental Material. The gradients alone provide the raw material for a quasi-Newton search of parameter space.

3.6.3 MM Algorithm for the VC Model Parameters

One can construct an iterative MM algorithm for updating $\boldsymbol{\theta}$ holding $\boldsymbol{\beta}$ fixed. There exists a substantial literature on the MM principle for optimization [20, 19, 32]. The idea in maximization is to concoct a surrogate function $g(\boldsymbol{\theta} \mid \boldsymbol{\theta}_r)$ that is easy to maximize and hugs the objective $f(\boldsymbol{\theta})$ tightly.

Here θ_r is the current value of θ. Construction of the surrogate is guided by two minorization requirements:

$$\begin{aligned} f(\theta) &\geq g(\theta \mid \theta_r) \quad \forall\, \theta \quad \text{(dominance condition)} \\ f(\theta_r) &= g(\theta_r \mid \theta_r) \quad \text{(tangent condition)}. \end{aligned}$$

The next iterate is determined by $\theta_{r+1} = \text{ argmax } g(\theta \mid \theta_r)$.

The MM principle guarantees that $f(\theta_{r+1}) \geq f(\theta_r)$, with strict inequality being the rule. In practice, minorization is carried out piecemeal on a sum of terms defining the objective.

For our particular problem we capitalize on the convexity of the function $-\ln(s)$. The supporting hyperplane inequality implies the linear minorization

$$-\sum_{i=1}^{n} \ln(1+\theta^t \mathbf{c}_i) \;\geq\; -\sum_{i=1}^{n} \frac{1}{1+\theta_r^t \mathbf{c}_i}(1+\theta^t \mathbf{c}_i - 1 - \theta_r^t \mathbf{c}_i).$$

On the other hand, Jensen's inequality gives the minorization

$$\begin{aligned} \sum_{i=1}^{n} \ln(1+\theta^t \mathbf{b}_i) \;\geq\; & \sum_{i=1}^{n} \frac{1}{1+\theta_r^t \mathbf{b}_i} \ln\left(\frac{1+\theta_r^t \mathbf{b}_i}{1} 1\right) \\ & + \sum_{i=1}^{n} \sum_{j} \frac{\theta_{rj} b_{ij}}{1+\theta_r^t \mathbf{b}_i} \ln\left(\frac{1+\theta_r^t \mathbf{b}_i}{\theta_{rj} b_{ij}} \theta_j b_{ij}\right). \end{aligned}$$

The sum of these two minorizations constitutes the overall minorization $g(\theta \mid \theta_r)$. The stationary condition $\nabla g(\theta \mid \theta_r) = \mathbf{0}$ can be solved to yield the updates

$$\theta_{r+1,j} \;=\; \theta_{rj} \frac{\sum_{i=1}^{n} \frac{b_{ri}}{1+\theta_r^t \mathbf{b}_i}}{\sum_{i=1}^{n} \frac{c_{ij}}{1+\theta_r^t \mathbf{c}_i}}.$$

Note that the update $\theta_{r+1,j}$ remains nonnegative if θ_{rj} is nonnegative and equals 0 if and only if $\theta_{rj} = 0$. However, convergence of θ_{rj} to 0 is possible. More importantly, the MM updates drive the loglikelihood uphill.

3.6.4 Initialization

Most optimization algorithms benefit from good starting values. The obvious candidate for $\boldsymbol{\beta}$ is the maximum likelihood estimate delivered by the independence model using `GLM.jl`. Under the VC framework, we use the MM algorithm to initialize variance components. Under the CS and AR(1) framework, we initialize the variance component σ^2 by the crude estimate from the MM algorithm treating $\rho = 1$.

3.7 Statistical Properties

Because the likelihood is a smooth function of the parameters in the quasi-copula model, we expect the maximum likelihood estimates $(\widehat{\boldsymbol{\beta}}, \widehat{\boldsymbol{\theta}})$ to be consistent and asymptotically efficient. One can estimate the asymptotic covariance matrix by the inverse of the observed information matrix. The expected information matrix is probably unavailable in closed form. It is straightforward to implement likelihood ratio tests on the mean components $\boldsymbol{\beta}$. Likelihood ratio testing on the variance components $\boldsymbol{\theta}$ is complicated by the same nonnegativity constraints implicit in all variance components models.

3.7.1 Compound Symmetric Covariance

Under the Compound Symmetric (CS) parameterization of $\boldsymbol{\Gamma}_i$, one can test hypotheses involving the correlation parameter ρ. To ensure that the covariance matrix Γ_i is positive semi-definite, we bound $\rho \in (-\frac{1}{d_i - 1}, 1)$. Additional details on this derivation can be found in the supplemental materials. For example, in the bivariate case, $d_i = 2$, $\rho \in (-1, 1)$ and

$$\begin{aligned} \boldsymbol{\Gamma}_i &= \sigma^2 \times \left[\rho \mathbf{1}_2 \mathbf{1}_2^t + (1 - \rho) I_2 \right] \\ &= \sigma^2 \times \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix} \end{aligned}$$

We are interested in the hypothesis $\mathbf{H_0} : \rho = \mathbf{0}$, which represents an independent univariate generalized linear mixed model with with a single variance component proportional to the identity matrix. The additional noise component captures overdispersion.

3.8 Results

3.8.1 Simulation Studies

To assess estimation accuracy of the quasi-copula model, we first present simulation studies for the Poisson and negative binomial base distributions with log link function, under the VC parameterization of $\mathbf{\Gamma}_i$. We then demonstrate the flexibility of the model in analyzing mixed discrete outcomes under a bivariate model with Poisson and Bernoulli base distributions and canonical link functions. Additional simulation studies with different base distributions under the AR(1), CS and VC parameterizations of $\mathbf{\Gamma}_i$ are included in the supplemental material.

In each simulation scenario, the non-intercept entries of the predictor matrix $\mathbf{X}_i$ are independent standard normal deviates. True regression coefficients $\boldsymbol{\beta}_{\text{true}} \sim \text{Uniform}(-0.2, 0.2)$. For the negative binomial base, all dispersion parameters are $\mathbf{r}_{\text{true}} = 10$. Each simulation scenario was run on 100 replicates for each sample size $n \in \{100, 1000, 10000\}$ and number of observations $d_i \in \{2, 5, 10, 15, 20, 25\}$ per independent sampling unit.

Under the VC parameterization of $\mathbf{\Gamma}_i$, the choice $\mathbf{\Gamma}_{i,\text{true}} = \theta_{\text{true}} \times \mathbf{1}_{d_i}\mathbf{1}_{d_i}^t$ allows us to compare to the random intercept GLMM fit using `MixedModels.jl`. When the random effect term is a scalar, `MixedModels.jl` uses Gaussian quadrature for parameter estimation. We compare estimates and run-times to the random intercept GLMM fit of `MixedModels.jl` with 25 Gaussian quadrature points. We conduct simulation studies under two scenarios (simulation I and II). In simulation I, it is assumed that the data are generated by the quasi-copula model with $\theta_{\text{true}} = 0.1$, and in simulation II, it is assumed that the true distribution is the random intercept GLMM with $\theta_{\text{true}} = 0.01, 0.05$.

Simulation I: In this scenario, we simulate datasets under the quasi-copula model as outlined in Section 3.5 and compare MLE fits under the quasi-copula model and GLMM. Figures 3.1 - 3.2 help us assess estimation accuracy and how well the GLMM density approximates the quasi-copula density. As anticipated, the MSE's across all base distributions decrease as sample size increases. For data simulated under the quasi-copula model, quasi-copula mean squared errors (MSE) are generally lower than GLMM MSE's. GLMM estimated variance components are often zero and stay relatively constant across sample sizes. This confirms the fact that the two models are different in how they handle random effects, particularly with larger sampling units ($d_i > 2$).

Simulation II: In the second simulation scenario, we generate datasets under the random intercept Poisson GLMM and compare MLE fits delivered by the two models. Figures 3.3 - 3.6 now shed light on how well the quasi-copula density approximates the GLMM density under different magnitudes of the variance components. As expected, MSE's under GLMM analysis are now generally lower than those under quasi-copula analysis. For the Poisson and negative binomial base distributions with $\theta_{\text{true}} = 0.05$, Figures 3.3 - 3.4 indicate biases for the quasi-copula estimates of $(\boldsymbol{\beta}, \boldsymbol{\theta})$ for larger sampling units $(d_i > 2)$ up to sample size $n = 10,000$. However in Figures 3.5 - 3.6, where the variance component $\theta_{\text{true}} = 0.01$ is smaller, we no longer observe biased estimates for $(\boldsymbol{\beta}, \boldsymbol{\theta})$.

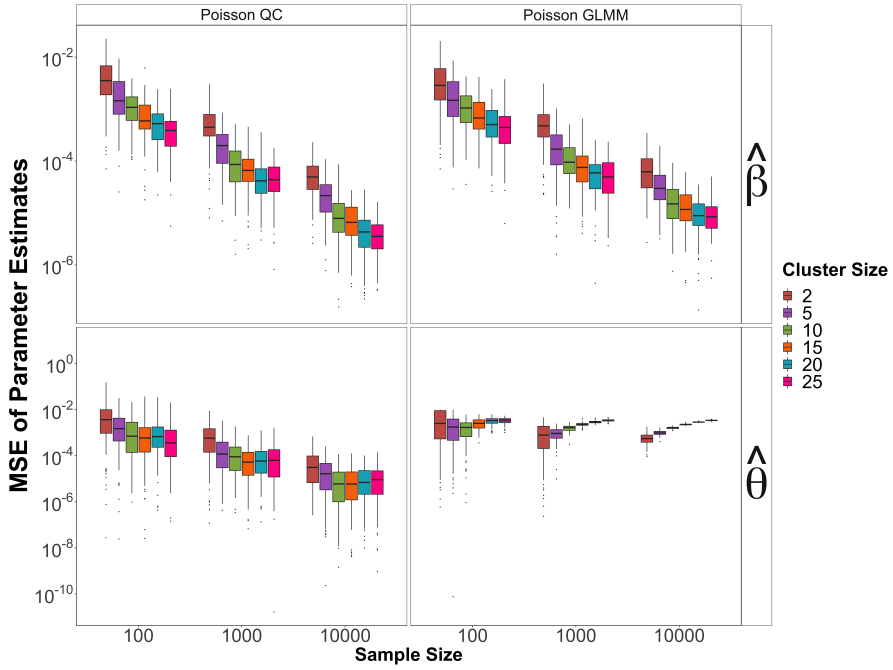

Figure 3.1: Simulation I: Mean squared errors (MSE) of parameter estimates $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ under the Poisson base distribution with log link function and a single VC versus a random intercept GLMM fit via `MixedModels.jl`. Each scenario result includes 100 replicates.

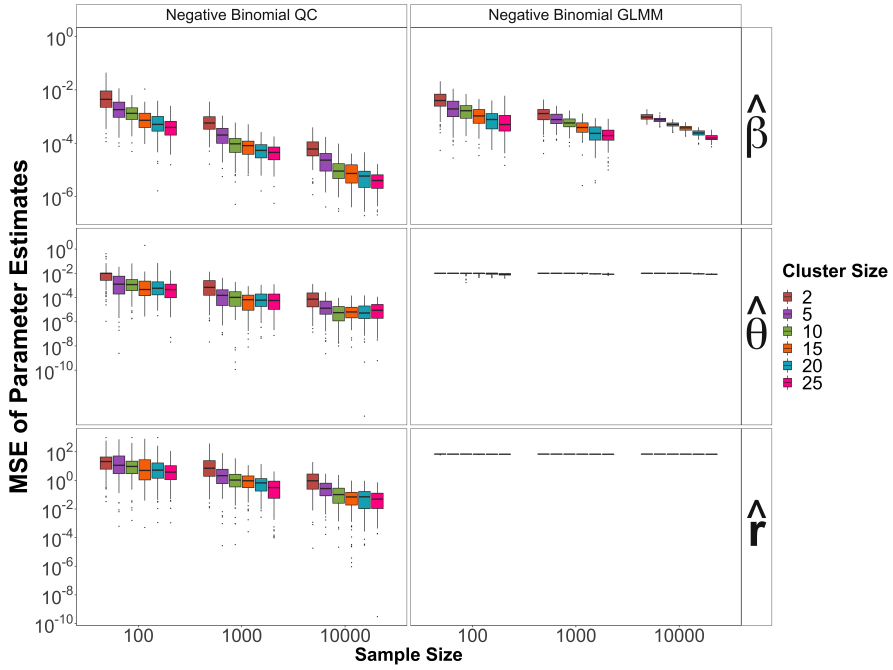

Figure 3.2: Simulation I: Mean squared errors (MSE) of parameter estimates $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ under the negative binomial base distribution with log link function and a single VC versus a random intercept GLMM fit via `MixedModels.jl`. Each scenario result includes 100 replicates.

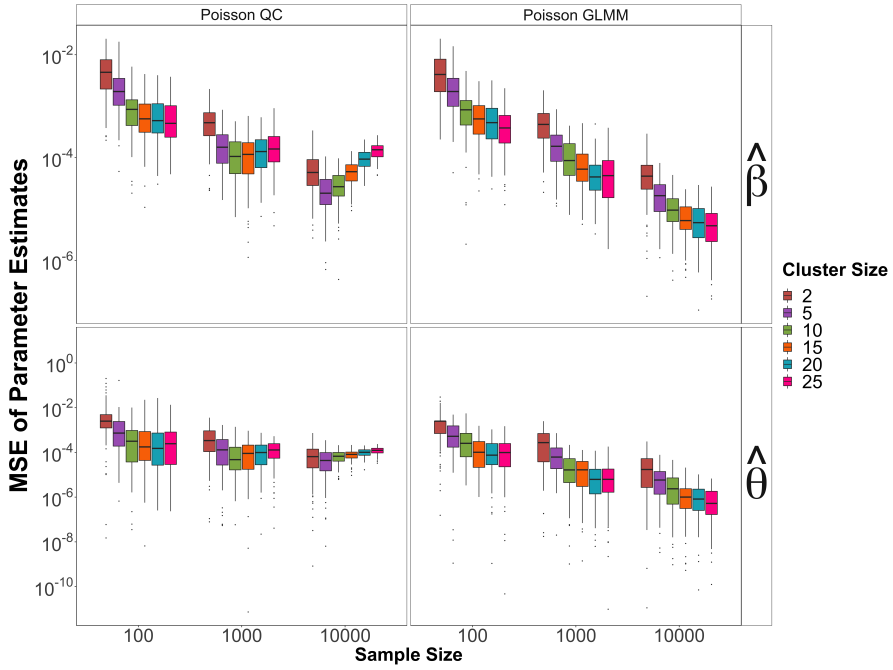

Figure 3.3: Simulation II: Mean squared errors (MSE) of parameter estimates $\boldsymbol{\beta}$ and $\boldsymbol{\theta} = 0.05$ under the Poisson base distribution with log link function and a single VC versus a random intercept GLMM fit via `MixedModels.jl`. Each scenario reports involves 100 replicates.

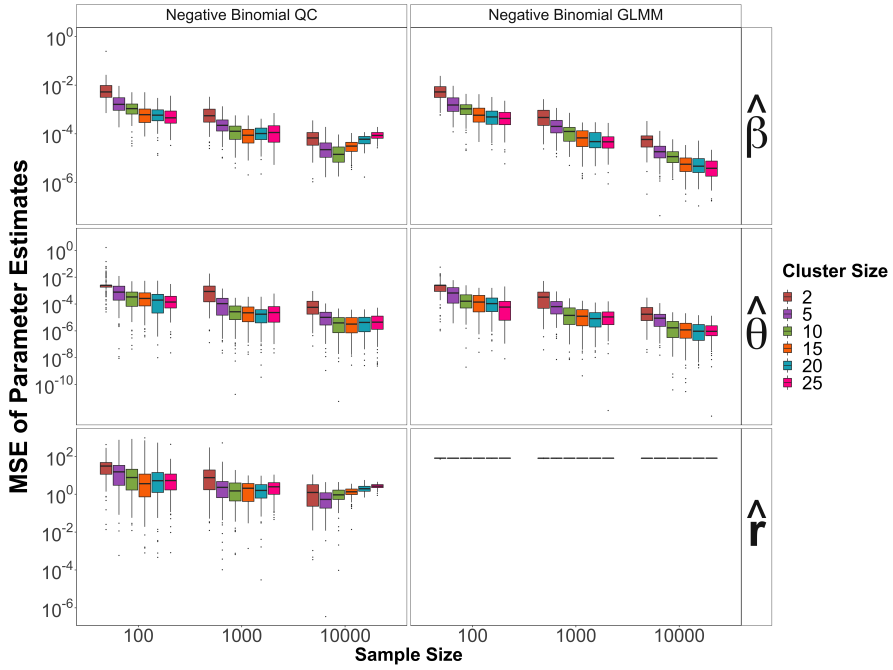

Figure 3.4: Simulation II: Mean squared errors (MSE) of parameter estimates $\boldsymbol{\beta}$ and $\boldsymbol{\theta} = 0.05$ under the negative binomial base distribution with log link function and a single VC versus a random intercept GLMM fit via `MixedModels.jl`. Each scenario result includes 100 replicates.

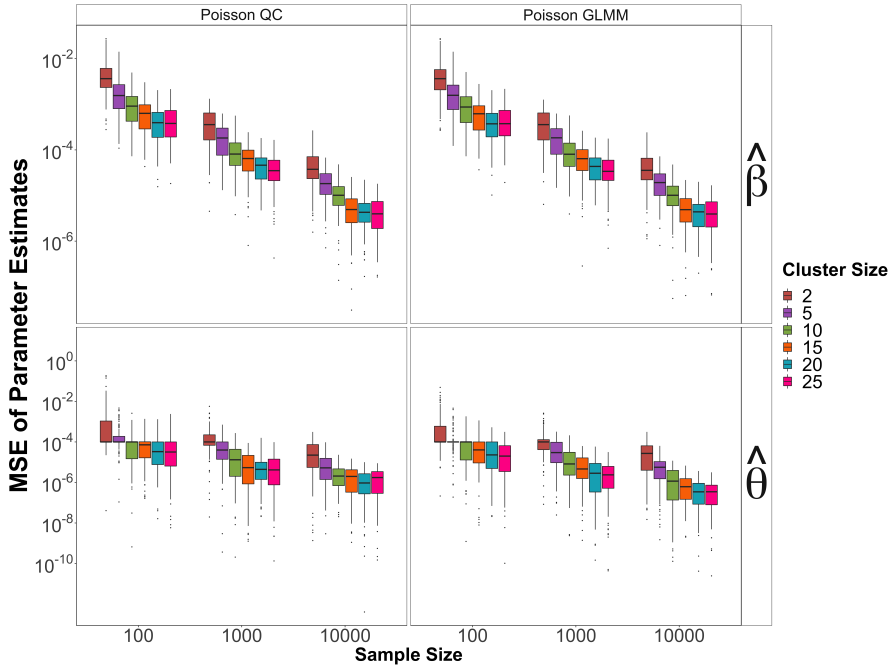

Figure 3.5: Simulation II: Mean squared errors (MSE) of parameter estimates $\boldsymbol{\beta}$ and $\boldsymbol{\theta} = 0.01$ under the Poisson base distribution with log link function and a single VC versus a random intercept GLMM fit via `MixedModels.jl`. Each scenario result includes 100 replicates.

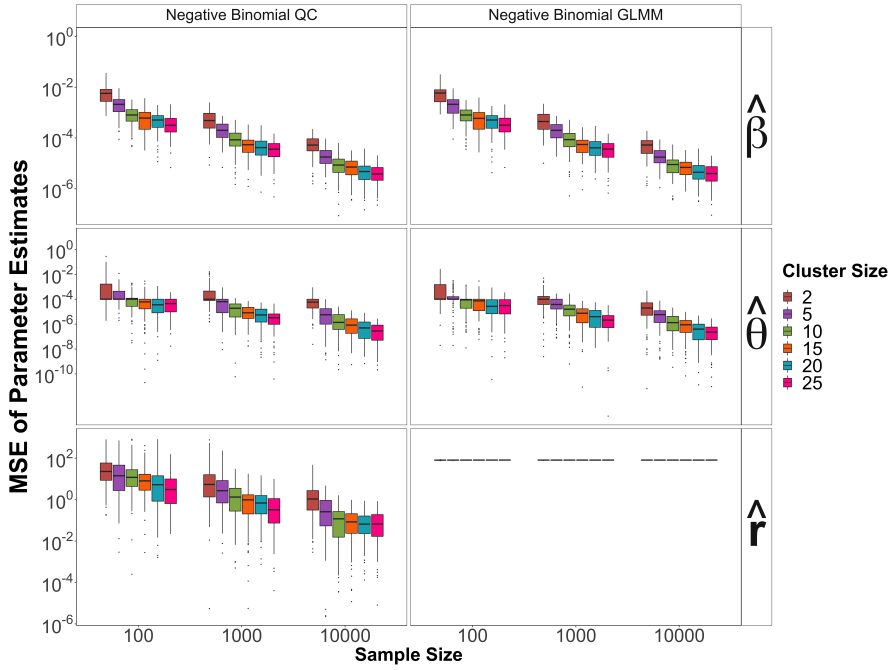

Figure 3.6: Simulation II: Mean squared errors (MSE) of parameter estimates $\boldsymbol{\beta}$ and $\boldsymbol{\theta} = 0.01$ under the negative binomial base distribution with log link function and a single VC versus a random intercept GLMM fit via `MixedModels.jl`. Each scenario result includes 100 replicates.

3.8.1.1 Run Times

Run times under simulation I and II are comparable. Table 3.1 presents average run times and their standard errors in seconds for 100 replicates under simulation II with $\theta_{\text{true}} = 0.01$. All computer runs were performed on a standard 2.3 GHz Intel i9 CPU with 8 cores. Runtimes for the quasi-copula model are presented given multi-threading across 8 cores. We note the current version of `MixedModels.jl` does not allow for multi-threading across multiple cores. Because in contrast to `MixedModels.jl` the quasi-copula loglikelihoods contain no determinants or matrix inverses, `QuasiCopula.jl` experiences less pronounced increases in computation time as sample and sampling unit sizes grow. Run times for the quasi-copula model are faster than those of `MixedModels.jl` for discrete outcomes (Table 3.1, Supplementary Table 3.9) and slower for Gaussian distributed outcomes (Supplementary Table 3.10). This general trend also holds on a per core

basis. This discrepancy is hardly surprising since `MixedModels.jl` takes into account the low-rank structure of the covariance matrix Ω_i in the random intercept linear mixed model (LMM). This tactic reduces the computational complexity per sample from $\mathbf{O}(d_i^3)$ to $\mathbf{O}(d_i^2)$. More detailed comparisons appear in the supplement.

For the negative binomial base distribution, `MixedModels.jl` explicitly warns the user against fitting GLMM's with unknown dispersion parameter r. Our software updates r iteratively by Newton's method, holding the other parameters $(\boldsymbol{\beta}, \boldsymbol{\theta})$ fixed. Our restriction to `MixedModels.jl` makes for a fair comparison within the Julia language universe. We also compared our negative binomial fits with those delivered by the three popular R packages for GLMM estimation in Table 3.2. On a single dataset with $d_i = 5$, and $n = 10,000$ simulated under simulation II, the `lme4` package [2] takes an inordinately long time to fit the model. Obtaining confidence intervals takes a significant amount of additional time, and inference of r is impossible. The `glmmTMB` package [7] allows for inference of r and takes much less time to form confidence intervals than `lme4`, but it is still significantly slower than quasi-copula fitting. Both `lme4` and `glmmTMB` fit the negative binomial GLMM using Laplace Approximation, while the `GLMMadaptive` package [24] uses adaptive Gaussian quadrature. In Tables 3.2 and 3.3, we use `GLMMadaptive` to fit the data with 25 Gaussian quadrature points. `GLMMadaptive` allows for inference of r and takes no additional time to form confidence intervals, but is still significantly slower than quasi-copula fitting. Run times in seconds for obtaining the estimates and confidence intervals in Table 3.2 appear in Table 3.3.

n	d_i	Poisson QC time	Poisson GLMM time	NB QC time	NB GLMM time
100	2	0.021 (<0.001)	0.022 (0.003)	0.125 (0.008)	0.037 (0.003)
100	5	0.020 (<0.001)	0.045 (0.003)	0.095 (0.005)	0.068 (0.004)
100	10	0.023 (0.001)	0.080 (0.004)	0.105 (0.004)	0.187 (0.011)
100	15	0.024 (0.001)	0.148 (0.006)	0.105 (0.004)	0.282 (0.017)
100	20	0.025 (0.001)	0.186 (0.007)	0.112 (0.002)	0.394 (0.017)
100	25	0.026 (<0.001)	0.265 (0.009)	0.119 (0.003)	0.461 (0.019)
1000	2	0.025 (<0.001)	0.192 (0.007)	0.163 (0.009)	0.365 (0.013)
1000	5	0.030 (<0.001)	0.516 (0.016)	0.167 (0.004)	0.857 (0.033)
1000	10	0.035 (0.001)	1.011 (0.022)	0.243 (0.003)	1.972 (0.050)
1000	15	0.040 (<0.001)	1.402 (0.030)	0.303 (0.002)	2.854 (0.064)
1000	20	0.042 (<0.001)	1.887 (0.036)	0.371 (0.002)	3.722 (0.077)
1000	25	0.051 (0.001)	2.531 (0.046)	0.435 (0.002)	4.815 (0.089)
10000	2	0.128 (0.001)	1.896 (0.032)	1.169 (0.040)	3.902 (0.079)
10000	5	0.154 (0.001)	4.333 (0.075)	1.375 (0.020)	8.598 (0.140)
10000	10	0.232 (0.002)	9.545 (0.143)	2.154 (0.007)	20.499 (0.303)
10000	15	0.272 (0.002)	14.844 (0.249)	2.78 (0.007)	29.003 (0.465)
10000	20	0.336 (0.002)	21.423 (0.356)	3.314 (0.007)	42.952 (0.679)
10000	25	0.429 (0.003)	29.324 (0.528)	4.111 (0.011)	54.676 (0.861)

Table 3.1: Run times and (standard error of run times) in seconds based on 100 replicates under simulation II with Poisson and negative binomial (NB) Base, $\theta_{\text{true}} = 0.01$, sampling unit size d_i and sample size n.

Parameter	Truth	QC fit	lme4 fit	glmmTMB fit	GLMMadaptive fit
β_1	0.036	0.033	0.032	0.033	0.032
		(0.028, 0.037)	(0.022, 0.042)	(0.023, 0.043)	(0.023, 0.042)
β_2	0.107	0.106	0.106	0.106	0.106
		(0.101, 0.111)	(0.097, 0.115)	(0.097, 0.115)	(0.097, 0.115)
β_3	0.026	0.026	0.026	0.026	0.026
		(0.017, 0.035)	(0.017, 0.035)	(0.017, 0.035)	(0.017, 0.035)
θ	0.01	0.007	0.009	0.008	0.009
		(0.003, 0.011)	(0.002, 0.015)	(0.003, 0.019)	(0.005, 0.018)
r	10	10.002	10.147	10.101	9.996
		(9.094, 10.910)	(NA, NA)	(8.640, 11.809)	(8.612, 11.602)

Table 3.2: MLE's and (confidence intervals) based on a single replicate under simulation II with negative binomial Base, $\theta_{\text{true}} = 0.01$, sampling unit size $d_i = 5$ and sample size $n = 10000$.

n	d_i	QC time	lme4 time	glmmTMB time	GLMMadaptive time
10000	5	1.247 (0.046)	75.774 (158.034)	98.944 (0.472)	84.471 (<0.001)

Table 3.3: Run times and (confidence interval run times) in seconds based on a single replicate under simulation II with negative binomial Base, $\theta_{\text{true}} = 0.01$, sampling unit size $d_i = 5$ and sample size $n = 10000$.

3.8.2 Bivariate Mixed Outcome Model

Let $\mathbf{y}_i = (y_{i1}, y_{i2})^t$ denote the ith bivariate mixed discrete outcome from n bivariate sampling units. For purposes of illustration we assume that y_{i1} follows a Poisson base distribution and y_{i2} follows a Bernoulli base distribution under their canonical link functions. Thus,

$$\begin{aligned} y_{i1} &\sim \text{Poisson}\,[\mu_{i1}(\boldsymbol{\beta}_1)], \quad \text{where} \quad \text{Log}[\mu_{i1}(\boldsymbol{\beta}_1)] = \mathbf{x}_i^t \boldsymbol{\beta}_1 \\ y_{i2} &\sim \text{Bernoulli}\,[\mu_{i1}(\boldsymbol{\beta}_2)], \quad \text{where} \quad \text{Logit}[\mu_{i2}(\boldsymbol{\beta}_2)] = \mathbf{x}_i^t \boldsymbol{\beta}_2. \end{aligned}$$

For each independent realization $\mathbf{y}_i$, we postulate a vector of covariates $\mathbf{x}_i$ and a corresponding vector of fixed effects, both of length p. An intercept is included among the fixed effects. The fixed

effects $\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_1 \\ \boldsymbol{\beta}_2 \end{bmatrix}$ for both responses are jointly estimated under the design matrix $\mathbf{X}_i = \begin{bmatrix} \mathbf{x}_i^t & 0_p^t \\ 0_p^t & \mathbf{x}_i^t \end{bmatrix}$. Estimation of the variance components $\boldsymbol{\theta}$ is unchanged. As expected, Figure 3.7 shows that all MSEs decrease as the sample size n increases.

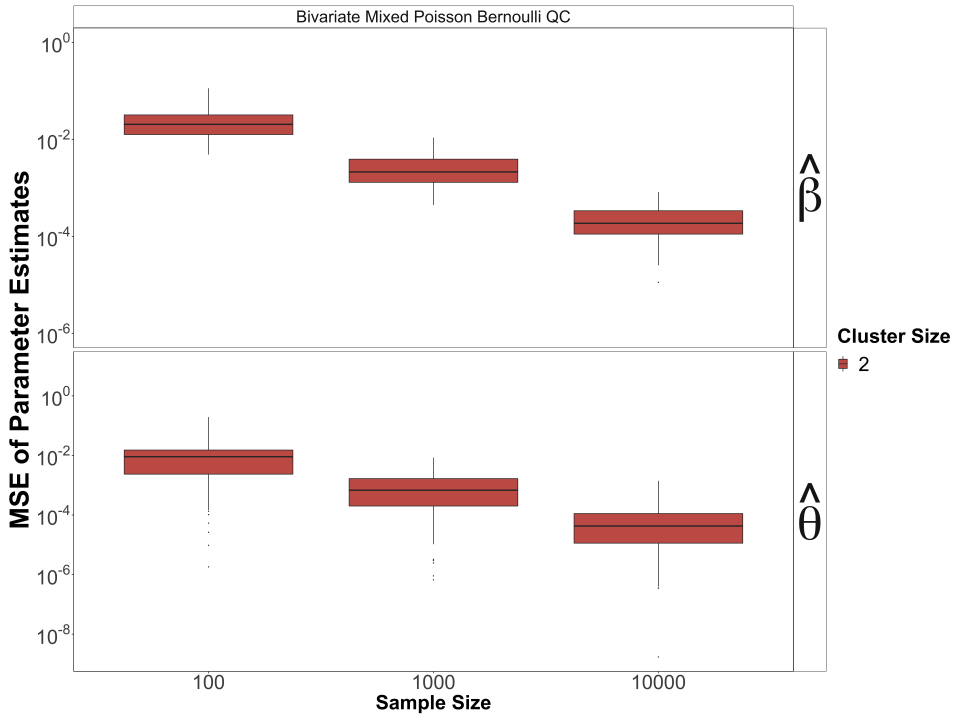

Figure 3.7: Simulation I: Mean squared errors (MSE) of MLE estimates $\boldsymbol{\beta}$ and $\boldsymbol{\theta} = 0.1$ under the Bivariate Mixed Outcome model with Poisson and Bernoulli base distributions and their canonical link functions. Each scenario reports involves 100 replicates.

3.8.3 NHANES Data Example

For many repeated measurement problems, a simple random intercept model is sufficient to account for correlations between different responses on the same subject. To illustrate this point and the performance of the quasi-copula model, we now turn to a bivariate example from the NHANES I Epidemiologic Followup Study (NHEFS) dataset [9]. In this example, we group the data by subject ID and jointly model the number of cigarettes smoked per day in 1971 and the number of cigarettes smoked per day in 1982 as a bivariate outcome. For fixed effects, we include an intercept and control for sex, age in 1971, and the average price of tobacco in the state of residence. The average price of tobacco is a time-dependent covariate that is adjusted for inflation using the 2008 U.S. consumer price index (CPI). Participants with missing responses or predictors were excluded from the model cohort. A total of $n = 1537$ NHANES I participants constitute the cohort. Table 3.4 compares the estimates, loglikelihoods and run times in seconds of the random intercept regression model with Poisson, negative binomial, and Bernoulli base distributions under `QuasiCopula.jl` and `MixedModels.jl`. For the Bernoulli base distribution, we transformed each count outcome to a binary indicator with value 1 if the number of cigarettes smoked per day is greater than the sample average and value 0 otherwise.

Because overdispersion is a feature of this dataset, the Poisson base distribution represents a case of model misspecification; the negative binomial base distribution is a better choice for analysis. Under the Poisson base distribution, the quasi-copula model inflates the variance component to account for the overdispersion. Under the negative binomial base distribution, both `QuasiCopula.jl` and `MixedModels.jl` estimate the variance component to be 0. This suggests that no additional overdispersion exists in the data. The estimates for $\boldsymbol{\beta}$ under the quasi-copula model with Poisson base are closer to the more realistic estimates under the negative binomial base than those of GLMM. The maximum loglikelihood of the quasi-copula model is lower than that of GLMM for the Poisson base and higher than that of GLMM for the negative binomial and Bernoulli bases. Run times favor the quasi-copula model.

Parameter	QC Poisson	GLMM Poisson	QC NB	GLMM NB	QC Bernoulli	GLMM Bernoulli
$\beta_{\text{Intercept}}$	2.509	2.039	2.580	2.580	-1.768	-1.411
β_{sex}	-0.210	-0.225	-0.187	-0.187	-0.793	-0.761
β_{age}	-0.009	-0.009	-0.009	-0.009	-0.040	-0.034
β_{price}	0.434	0.597	0.402	0.402	2.238	1.891
θ	7.080	0.458	0.0	0.0	0.666	3.461
r	-	-	1.141	1.395	-	-
loglikehood	-20690.797	-15499.537	-12037.587	-12047.504	-1938.712	-1980.893
time (seconds)	0.268	0.749	0.160	0.978	0.109	1.030

Table 3.4: Random intercept MLEs, loglikelihoods, and run times for for the NHEFS data under the quasi-copula (QC) model and GLMM. All $n = 1537$ sampling units are of size $d_i = 2$.

3.9 Discussion

We propose a new model for analyzing multivariate data based on Tonda's Gaussian copula approximation. Our quasi-copula model enables the analysis of correlated responses and handles random effects needed in applications such as panel and repeated measures data. The quasi-copula model trades Tonda's awkward parameter space constraint for a simple normalizing constant. This allows one to engage in full likelihood analysis under a tractable probability density function with no implicit integrations or matrix inverses. The quasi-copula model is relatively easy to fit and friendly to likelihood ratio hypothesis testing. Additionally, it easily extends to accommodate mixtures of different base distributions.

For maximum likelihood estimation, we recommend a combination of two numerical methods. The first is a block ascent algorithm that alternates between updating the mean parameters $\boldsymbol{\beta}$ by a version of Newton's method and updating the variance components by a minorization-maximization (MM) algorithm. The second method jointly updates $\boldsymbol{\beta}$ and the variances components by a standard quasi-Newton algorithm. The MM algorithm converges quickly to a neighborhood of the MLE but then slows. In contrast, the quasi-Newton struggles at first and then converges quickly. Thus, we start with the block ascent algorithm and then switch to the quasi-Newton algorithm. Both algorithms and their combination are available in our `QuasiCopula.jl` Julia package.

On balance our numerical tests suggest limitations of the quasi-copula model in handling strongly correlated responses and large sampling units. The presence and size of the normalizing constant $1 + \text{tr}(\boldsymbol{\Gamma})$ in the quasi-copula density may well be the culprit. When the true distribution follows the random intercept GLMM, the quasi-copula estimates are most accurate for small sampling units. When sampling units are large, the quasi-copula estimates are reasonably accurate for smaller magnitudes of variance components. In actual practice many statisticians simply assume the validity of their underlying statistical model.

3.10 Supplemental Material

3.10.1 Tonda's Approximation Details

Let $\mathbf{x}$ be a random vector with exponential density $f(\mathbf{x} \mid \nu) = e^{\mathbf{T}(\mathbf{x})^t \nu - A(\nu)}$. Note that $\mathbf{T}(\mathbf{x})$ has mean $\boldsymbol{\mu}(\nu) = \nabla A(\nu)$ and covariance matrix $d^2 A(\nu)$. Let us shift ν by adding a random Gaussian $\mathbf{z}$ with mean $\mathbf{0}$ and covariance $\boldsymbol{\Sigma}$. The new density $E[e^{\mathbf{T}(\mathbf{x})^t(\nu+\mathbf{z})-A(\nu+\mathbf{z})}]$ can be approximated by expanding the integrand to second order around $\mathbf{z} = \mathbf{0}$ and integrating. This yields

$$\begin{aligned}
\mathbb{E}[e^{\mathbf{T}(\mathbf{x})^t(\nu+\mathbf{z})-A(\nu+\mathbf{z})}] &\approx \mathbb{E}\Big(e^{\mathbf{T}(\mathbf{x})^t(\nu)-A(\nu)}\{1+[\mathbf{T}(\mathbf{x})-\nabla A(\nu)]^t\mathbf{z} \\
&\quad +\frac{1}{2}\mathbf{z}^t[\mathbf{T}(\mathbf{x})-\nabla A(\nu)][\mathbf{T}(\mathbf{x})-\nabla A(\nu)]^t\mathbf{z} \\
&\quad -\frac{1}{2}\mathbf{z}^t d^2A(\nu)\mathbf{z}\}\Big) \\
&= e^{\mathbf{T}(\mathbf{x})^t(\nu)-A(\nu)}\Big(1+\frac{1}{2}\mathrm{tr}\{[\mathbf{T}(\mathbf{x})-\nabla A(\nu)][\mathbf{T}(\mathbf{x})-\nabla A(\nu)]^t\boldsymbol{\Sigma}\} \\
&\quad -\frac{1}{2}\mathrm{tr}[d^2A(\nu)\boldsymbol{\Sigma}]\Big) \\
&= e^{\mathbf{T}(\mathbf{x})^t(\nu)-A(\nu)}\Big(1+\frac{1}{2}\mathrm{tr}\{[\mathbf{T}(\mathbf{x})-\boldsymbol{\mu}(\nu)][\mathbf{T}(\mathbf{x})-\boldsymbol{\mu}(\nu)]^t\boldsymbol{\Sigma}\} \\
&\quad -\frac{1}{2}\mathrm{tr}[d^2A(\nu)\boldsymbol{\Sigma}]\Big) \\
&= e^{\mathbf{T}(\mathbf{x})^t(\nu)-A(\nu)}\Big\{1+\frac{1}{2}\mathbf{W}^t\sqrt{d^2A(\nu)}\boldsymbol{\Sigma}\sqrt{d^2A(\nu)}\mathbf{W} \\
&\quad -\frac{1}{2}\mathrm{tr}[\sqrt{d^2A(\nu)}\boldsymbol{\Sigma}\sqrt{d^2A(\nu)}]\Big\},
\end{aligned}$$

where $\mathbf{W}$ is the standardized version $[\mathbf{T}(\mathbf{x})-\boldsymbol{\mu}(\nu)]d^2A(\nu)^{-1/2}$ of the base sufficient statistic $\mathbf{T}(\mathbf{x})$. The condition $1 - -\frac{1}{2}\mathrm{tr}[\sqrt{d^2A(\nu)}\boldsymbol{\Sigma}\sqrt{d^2A(\nu)}] > 0$ is sufficient but not necessary for the approximate density to be nonnegative. When this condition holds, the approximate density has mass 1. In our quasi-copula density, we drop the offending term $-\frac{1}{2}\mathrm{tr}[\sqrt{d^2A(\nu)}\boldsymbol{\Sigma}\sqrt{d^2A(\nu)}]$, replace $\sqrt{d^2A(\nu)}\boldsymbol{\Sigma}\sqrt{d^2A(\nu)}$ by $\boldsymbol{\Gamma}$, assume $T(\mathbf{x}) = \mathbf{x}$, and normalize.

3.10.2 Generate Random Deviates

We can construct the d dimensional multivariate vector, $\mathbf{y}$ from the multivariate density $g_{\mathbf{y}}(\mathbf{y})$ element wise using conditional densities. We recognize the joint density can be represented as a product of conditional densities:

$$g_{\mathbf{y}}(\mathbf{y}) = g_{y_1}(y_1) \times g_{y_2|y_1}(y_2|y_1) \times ... \times g_{y_d|y_1,...,y_{d-1}}(y_d|y_1,...,y_{d-1})$$

Thus we can first sample y_1 from its marginal density $g_{y_1}(y_1)$, and then sample y_2 from the conditional density $g_{y_2|y_1}(y_2|y_1)$. The resulting set is a sample from the joint density of $g_{y_1,y_2}(y_1,y_2)$. Continuing this process for all n values of the multivariate vector, $\mathbf{y}$, we can sample from it's joint density $g_{\mathbf{y}}(\mathbf{y})$. First we derive the form of the marginal densities $g_{y_1}(y_1)$, and then show the derivation of the conditional density $g_{y_d|y_1,...,y_{d-1}}(y_d|y_1,...,y_{d-1})$.

3.10.2.1 Marginal Distribution

For every univariate base distribution, the required probability density functions (PDFs) $g_y(y)$ are of the same form, where c_0, c_1 and c_2 are constants that depend on the parameters of the specified base distribution $f_y(y)$.

$$g_y(y) = cf_y(y)[a_0 + a_1(y-\mu) + a_2(y-\mu)^2] \tag{3.2}$$

$$= cf_y(y) \times \left[c_0 + c_1 y + c_2 y^2\right], \tag{3.3}$$

We can re-arrange the PDF to derive the constants c_0, c_1, c_2 in the marginal PDF $g_y(y)$ as follows:

$$\begin{aligned}
g_y(y) &= \left(1+\frac{1}{2}\text{tr}\Gamma\right)^{-1} f_y(y)\Big[1+\frac{\gamma_{11}}{2}\Big(\frac{y-\mu}{\sigma}\Big)^2+\frac{1}{2}\sum_{j=2}^{d}\gamma_{jj}\Big] \\
&= \left(1+\frac{1}{2}\text{tr}\Gamma\right)^{-1} f_y(y)\Big[1+\frac{\gamma_{11}}{2}\left(\frac{y^2-2y\mu+\mu^2}{\sigma^2}\right)+\frac{1}{2}\sum_{j=2}^{d}\gamma_{jj}\Big] \\
&= \left(1+\frac{1}{2}\text{tr}\Gamma\right)^{-1} f_y(y)\Big[1+\frac{\gamma_{11}}{2}\left(\frac{y^2}{\sigma^2}\right)+\frac{\gamma_{11}}{2}\left(\frac{-2y\mu}{\sigma^2}\right)+\frac{\gamma_{11}}{2}\left(\frac{\mu^2}{\sigma^2}\right)+\frac{1}{2}\sum_{j=2}^{d}\gamma_{jj}\Big] \\
&= \left(1+\frac{1}{2}\text{tr}\Gamma\right)^{-1} f_y(y)\left[\left(1+\frac{\gamma_{11}}{2}\Big(\frac{\mu^2}{\sigma^2}\Big)+\frac{1}{2}\sum_{j=2}^{d}\gamma_{jj}\right)+\left(\frac{\gamma_{11}}{2}\Big(\frac{-2\mu}{\sigma^2}\Big)\right)y+\left(\frac{\gamma_{11}}{2}\Big(\frac{1}{\sigma^2}\Big)\right)y^2\right] \\
&= c\times f_y(y)\Big[\Big(c_0\Big)+\Big(c_1\Big)y+\Big(c_2\Big)y^2\Big],
\end{aligned}$$

- $c=\left[1+\frac{1}{2}\text{tr}(\Gamma)\right]^{-1}$, for all base distributions $f_y(y)$.
- $c_0=\left(1+\frac{\gamma_{11}}{2}\left(\frac{\mu^2}{\sigma^2}\right)+\frac{1}{2}\sum_{j=2}^{d}\gamma_{jj}\right)$,
- $c_1=\left(\frac{\gamma_{11}}{2}\left(\frac{-2\mu}{\sigma^2}\right)\right)$,
- $c_2=\left(\frac{\gamma_{11}}{2}\left(\frac{1}{\sigma^2}\right)\right)$

Thus, the required marginal Cumulative Distribution Function (CDF) $G_y(x)$ takes the following form. We will derive the CDF by finding the appropriate scaled cumulative distributions of the three

terms.

$$\begin{aligned} G_y(x) &= \int_0^{\infty} g_y(y)\,dy \\ &= c\int_{-\infty}^{x} f(y)[c_0 + c_1 y + c_2 y^2]\,dy \\ &= c \times c_0 \int_{-\infty}^{x} f_y(y)\,dy \\ &+ c \times c_1 \int_{-\infty}^{x} y f_y(y)\,dy \\ &+ c \times c_2 \int_{-\infty}^{x} y^2 f_y(y)\,dy \\ &= \textbf{term1} + \textbf{term2} + \textbf{term3} \end{aligned}$$

The first term is a scalar multiple of the base distribution CDF, $F_y(y)$, and d_1, d_2 are normalizing constants for random variables v_1, v_2 from named distributions $f_{v_1}(v_1), f_{v_2}(v_2)$ with CDFs $F_{v_1}(x)$ and $F_{v_2}(x)$ in terms 2 and 3, respectively.

- **term1** $= c \times c_0 \int_{-\infty}^{x} f_y(y)\,dy = c \times (c_0) \times F_y(x)$,
- **term2** $= c \times c_1 \int_{-\infty}^{x} y * f_y(y)\,dy = c \times c_1 \times d_1 \times \int_{-\infty}^{x} f_{v_1}(y)\,dy = c \times (c_1) \times d_1 \times F_{v_1}(x)$
- **term3** $= c \times c_2 \int_{-\infty}^{x} y^2 f_y(y)\,dy = c \times c_2 \times d_2 \times \int_{-\infty}^{x} f_{v_2}(y)\,dy = c \times (c_2) \times d_2 \times F_{v_2}(x)$

For every base distribution, to satisfy properties of a proper distribution function we require

$$c \times [c_0 + c_1 \times d_1 + c_2 \times d_2] = 1.$$

3.10.2.2 Conditional Distribution

Let $\mathbf{y}_{[\mathbf{i-1}]}$ indicate elements $y_1, ..., y_{i-1}, \forall i \in [1, d]$. Then the conditional density of y_i given the previous components $\mathbf{y}_{[\mathbf{i-1}]}$ is:

$$
\begin{aligned}
g_{y_i|\mathbf{y}_{[\mathbf{i-1}]}}(y_i|\mathbf{y}_{[\mathbf{i-1}]}) &= d_{[i-1]}^{-1} f_i(y_i)\Big[d_{[i-1]} + r_i \sum_{j=1}^{i-1} r_j \gamma_{ij} + \frac{\gamma_{ii}}{2}(r_i^2 - 1)\Big] \\
&= d_{[i-1]}^{-1} f_i(y_i)\Big[d_{[i-1]} + \Big(\frac{y_i - \mu_i}{\sigma_i}\Big) \sum_{j=1}^{i-1} r_j \gamma_{ij} + \frac{\gamma_{ii}}{2}(r_i^2 - 1)\Big] \\
&= d_{[i-1]}^{-1} f_i(y_i)\Big[\Big(d_{[i-1]}^{-1} - \frac{\gamma_{ii}}{2}\Big) + \Big(\frac{\sum_{j=1}^{i-1} r_j \gamma_{ij}}{\sigma_i}\Big) y_i - \mu_i \Big(\frac{\sum_{j=1}^{i-1} r_j \gamma_{ij}}{\sigma_i}\Big) + \frac{\gamma_{ii}}{2}\frac{(y_i - \mu_i)^2}{\sigma_i^2}\Big] \\
&= d_{[i-1]}^{-1} f_i(y_i)\Big[\Big(d_{[i-1]}^{-1} - \frac{\gamma_{ii}}{2} - \mu_i \Big(\frac{\sum_{j=1}^{i-1} r_j \gamma_{ij}}{\sigma_i}\Big)\Big) + \Big(\frac{\sum_{j=1}^{i-1} r_j \gamma_{ij}}{\sigma_i}\Big) y_i + \frac{\gamma_{ii}}{2}\frac{(y_i - \mu_i)^2}{\sigma_i^2}\Big] \\
&= d_{[i-1]}^{-1} f_i(y_i)\Big[\Big(d_{[i-1]}^{-1} - \frac{\gamma_{ii}}{2} - \mu_i \Big(\frac{\sum_{j=1}^{i-1} r_j \gamma_{ij}}{\sigma_i}\Big) + \frac{\gamma_{ii}}{2}\frac{\mu_i^2}{\sigma_i^2}\Big) \\
&+ \Big(\frac{\sum_{j=1}^{i-1} r_j \gamma_{ij}}{\sigma_i} + \frac{\gamma_{ii}}{2}\Big(\frac{-2\mu_i}{\sigma_i^2}\Big)\Big) y_i \\
&+ \Big(\frac{\gamma_{ii}}{2}\Big(\frac{1}{\sigma_i^2}\Big)\Big) y_i^2\Big] \\
&= c f_i(y_i)\Big[c_0 + c_1 y_i + c_2 y_i^2\Big]
\end{aligned}
$$

where $d_{[i-1]} = 1 + \frac{1}{2}\mathbf{r}_{[i-1]}^t \mathbf{\Gamma}_{[i-1]} \mathbf{r}_{[i-1]} + \frac{1}{2}\sum_{j=i}^{d} \gamma_{jj}$.

- $c = d_{[i-1]}^{-1}$
- $c_0 = \Big(d_{[i-1]}^{-1} - \frac{\gamma_{ii}}{2} - \mu_i \Big(\frac{\sum_{j=1}^{i-1} r_j \gamma_{ij}}{\sigma_i}\Big) + \frac{\gamma_{ii}}{2}\frac{\mu_i^2}{\sigma_i^2}\Big)$
- $c_1 = \Big(\frac{\sum_{j=1}^{i-1} r_j \gamma_{ij}}{\sigma_i} + \frac{\gamma_{ii}}{2}\Big(\frac{-2\mu_i}{\sigma_i^2}\Big)\Big)$
- $c_2 = \Big(\frac{\gamma_{ii}}{2}\Big(\frac{1}{\sigma_i^2}\Big)\Big)$

We can construct each conditional density given the previously sampled elements as a combination of three constants, just as in the marginal density.

3.10.2.3 Continuous Outcomes

When marginal densities $f(y)$ are continuous, each stage of sampling is probably best performed by inverse transform sampling.

Gamma distribution

This note considers the special case of Gamma base in the copula framework outlined in Ken's notes, where $f_1(y) \sim \Gamma(\alpha, \theta)$. We will simulate y directly from its marginal density, $g_y(y)$, which can also be represented as a mixture distribution.

$$y \sim \Gamma(\alpha, \theta); f_y(y) = \frac{1}{\Gamma(\alpha)\theta^{\alpha}} y^{\alpha-1} e^{\frac{-y}{\theta}}$$

- $\mu = E[y] = \alpha\theta$, and $\sigma^2 = Var(y) = \alpha\theta^2$.

$$\begin{aligned} g_y(y) &= \Big[1 + \frac{1}{2}\text{tr}(\boldsymbol{\Gamma})\Big]^{-1} \Big[\frac{1}{\Gamma(\alpha)\theta^{\alpha}} y^{\alpha-1} e^{\frac{-y}{\theta}}\Big] \Big(1 + \frac{\gamma_{11}}{2}\left[\frac{(y-\mu)^2}{\sigma^2}\right] + \frac{1}{2}\sum_{j=2}^{d}\gamma_{jj}\Big) \\ &= \Big[1 + \frac{1}{2}\text{tr}(\boldsymbol{\Gamma})\Big]^{-1} \Big[\frac{1}{\Gamma(\alpha)\theta^{\alpha}} y^{\alpha-1} e^{\frac{-y}{\theta}}\Big] \left(\Big(1 + \frac{1}{2}\sum_{j=2}^{d}\gamma_{jj}\Big) + \frac{\gamma_{11}}{2}\left[\frac{y^2 - 2y\mu + \mu^2}{\sigma^2}\right]\right) \\ &= c \times f_y(y)\Big[\Big(c_0\Big) + \Big(c_1\Big)y + \Big(c_2\Big)y_2^2\Big] \end{aligned}$$

- Here $c_0 = \left(1 + \frac{\gamma_{11}}{2}\left(\frac{\mu^2}{\sigma^2}\right) + \frac{1}{2}\sum_{j=2}^{d}\gamma_{jj}\right) = \left(1 + \frac{\gamma_{11}}{2}\Big(\alpha\Big) + \frac{1}{2}\sum_{j=2}^{d}\gamma_{jj}\right)$
- $c_1 = \left(\frac{\gamma_{11}}{2}\left(\frac{-2\mu}{\sigma^2}\right)\right) = \left(\frac{\gamma_{11}}{2}\left(\frac{-2}{\theta}\right)\right)$
- $c_2 = \left(\frac{\gamma_{11}}{2}\left(\frac{1}{\sigma^2}\right)\right) = \left(\frac{\gamma_{11}}{2}\left(\frac{1}{\alpha\theta^2}\right)\right)$

We will use the given information here: $y \sim \Gamma(\alpha, \theta)$, where $F_Y(x) = P(y \leq x)$ to derive the CDF

$G_Y(x)$ term by term.

$$\begin{aligned}
\textbf{term1} &= c \times (c_0) \int_0^x f_y(y)dy \\
&= c \times (c_0) \times F_Y(Y = x) \\
&= c \times \left(1 + \frac{\gamma_{11}}{2}\left(\alpha\right) + \frac{1}{2}\sum_{j=2}^{m}\gamma_{jj}\right) \times F_Y(Y = x)
\end{aligned}$$

Define a new random variable $v_1 \sim \Gamma(\alpha+1, \theta)$, where $F_{v_1}(x) = P(v_1 \leq x)$.

$$\begin{aligned}
\textbf{term2} &= c \times (c_1) \times \int_0^x y f_y(y)dy \\
&= c \times \left(\frac{\gamma_{11}}{2}\left(\frac{-2}{\theta}\right)\right) \times \int_0^x y f_y(y)dy \\
&= c \times \left(\frac{\gamma_{11}}{2}\left(\frac{-2}{\theta}\right)\right) \times \theta \int_0^x \frac{y}{\theta} f_y(y)dy \\
&= c \times \left(\frac{\gamma_{11}}{2}\left(\frac{-2}{\theta}\right)\right) \times \theta \times \int_0^x \left[\frac{1}{\Gamma(\alpha)\theta^{\alpha+1}} y^{(\alpha+1)-1} e^{\frac{-y}{\theta}}\right] dy \\
&= c \times \left(\frac{\gamma_{11}}{2}\left(\frac{-2}{\theta}\right)\right) \times \theta \times \frac{\Gamma(\alpha+1)}{\Gamma(\alpha)} \times \int_0^x \left[\frac{1}{\Gamma(\alpha+1)\theta^{\alpha+1}} y^{(\alpha+1)-1} e^{\frac{-y}{\theta}}\right] dy \\
&= c \times \left(\frac{\gamma_{11}}{2}\left(\frac{-2}{\theta}\right)\right) \times \theta \times \frac{\Gamma(\alpha+1)}{\Gamma(\alpha)} \times F_{v_1}(x)
\end{aligned}$$

Define another random variable $v_2 \sim \Gamma(\alpha+2,\theta)$; with CDF $F_{v_2}(x) = P(v_2 \leq x)$.

$$
\begin{aligned}
\textbf{term 3} &= c \times (c_2) \times \int_0^x y^2 f_y(y)\,dy \\
&= c \times \left(\frac{\gamma_{11}\left(\frac{1}{\alpha\theta^2}\right)}{2} \right) \times \int_0^x y^2 f_y(y)\,dy \\
&= c \times \left(\frac{\gamma_{11}\left(\frac{1}{\alpha\theta^2}\right)}{2} \right) \times \theta^2 \int_0^x \frac{y^2}{\theta^2} f_y(y)\,dy \\
&= c \times \left(\frac{\gamma_{11}\left(\frac{1}{\alpha\theta^2}\right)}{2} \right) \times \theta^2 \times \int_0^x \left[\frac{1}{\Gamma(\alpha)\theta^{\alpha+2}} y^{(\alpha+2)-1} e^{\frac{-y}{\theta}} \right] dy \\
&= c \times \left(\frac{\gamma_{11}\left(\frac{1}{\alpha\theta^2}\right)}{2} \right) \times \theta^2 \times \frac{\Gamma(\alpha+2)}{\Gamma(\alpha)} \times \int_0^x \left[\frac{1}{\Gamma(\alpha+2)\theta^{\alpha+2}} y^{(\alpha+2)-1} e^{\frac{-y}{\theta}} \right] dy \\
&= c \times \left(\frac{\gamma_{11}\left(\frac{1}{\alpha\theta^2}\right)}{2} \right) \times \theta^2 \times \frac{\Gamma(\alpha+2)}{\Gamma(\alpha)} \times F_{v_2}(x)
\end{aligned}
$$

Exponential distribution

Next, we consider when $f_y(y) \sim$ Exponential$(\frac{1}{\theta})$. To find the appropriate CDF function under this exponential base, we make note of the relationship between the exponential and gamma densities.

$$y \sim \text{Exponetial}\left(\frac{1}{\theta}\right) \iff y \sim \Gamma(\alpha = 1, \theta \geq 0);$$

$$f_y(y) = \frac{1}{\theta} e^{\frac{-y}{\theta}} = \frac{1}{\Gamma(1)\theta^1} y^{1-1} e^{\frac{-y}{\theta}}; y, \theta \geq 0$$

- $\mu = E[y] = \theta$, and $\sigma^2 = Var(y) = \theta^2$.

$$\begin{aligned} g_y(y) &= \left[1+\frac{1}{2}\text{tr}(\mathbf{\Gamma})\right]^{-1}\left[\frac{1}{\Gamma(1)\theta^1}y^{1-1}e^{\frac{-y}{\theta}}\right]\left(1+\frac{\gamma_{11}}{2}\left[\frac{(y-\mu)^2}{\sigma^2}\right]+\frac{1}{2}\sum_{j=2}^{d}\gamma_{jj}\right) \\ &= \left[1+\frac{1}{2}\text{tr}(\mathbf{\Gamma})\right]^{-1}\left[\frac{1}{\Gamma(1)\theta^1}y^{1-1}e^{\frac{-y}{\theta}}\right]\left(\left(1+\frac{1}{2}\sum_{j=2}^{d}\gamma_{jj}\right)+\frac{\gamma_{11}}{2}\left[\frac{y^2-2y\mu+\mu^2}{\sigma^2}\right]\right) \\ &= c\times f_y(y)\left[\left(c_0\right)+\left(c_1\right)y+\left(c_2\right)y_2^2\right] \end{aligned}$$

- Here $c_0 = \left(1+\frac{\gamma_{11}}{2}\left(\frac{\mu^2}{\sigma^2}\right)+\frac{1}{2}\sum_{j=2}^{d}\gamma_{jj}\right) = \left(1+\frac{\gamma_{11}}{2}+\frac{1}{2}\sum_{j=2}^{d}\gamma_{jj}\right)$
- $c_1 = \frac{\gamma_{11}}{2}\left(\frac{-2\mu}{\sigma^2}\right) = \frac{\gamma_{11}}{2}\left(\frac{-2}{\theta}\right)$
- $c_2 = \frac{\gamma_{11}}{2}\left(\frac{1}{\sigma^2}\right) = \frac{\gamma_{11}}{2}\left(\frac{1}{\theta^2}\right)$

$y \sim \text{Exponential}(\frac{1}{\theta}) = \Gamma(\alpha = 1, \theta)$, with CDF $F_Y(Y = x) = P(Y \leq x)$

$$\begin{aligned} \textbf{term1} &= c\times(c_0)\int_0^x f_y(y)dy \\ &= c\times(c_0)\times F_y(x) \\ &= c\times\left(1+\frac{\gamma_{11}}{2}+\frac{1}{2}\sum_{j=2}^{d}\gamma_{jj}\right)\times F_y(x) \end{aligned}$$

Define a new random variable $v_1 \sim \Gamma(\alpha+1,\theta) = \Gamma(2,\theta)$, with CDF $F_{v_1}(x) = P(v_1 \leq x)$.

$$\begin{aligned}
\textbf{term2} &= c \times (c_1) \times \int_0^x y f_y(y) dy \\
&= c \times \left(\frac{\gamma_{11}}{2} \left(\frac{-2}{\theta} \right) \right) \times \int_0^x y f_y(y) dy \\
&= c \times \left(\frac{\gamma_{11}}{2} \left(\frac{-2}{\theta} \right) \right) \times \int_0^x \left[\frac{1}{\Gamma(1)\theta^1} y^{(1+1)-1} e^{\frac{-y}{\theta}} \right] dy \\
&= c \times \left(\frac{\gamma_{11}}{2} \left(\frac{-2}{\theta} \right) \right) \times \frac{\theta^2}{\theta} \times \frac{\Gamma(2)}{\Gamma(1)} \times \int_0^x \left[\frac{1}{\Gamma(2)\theta^2} y^{(2)-1} e^{\frac{-y}{\theta}} \right] dy \\
&= c \times \left(\frac{\gamma_{11}}{2} \left(\frac{-2}{\theta} \right) \right) \times \theta \times \frac{\Gamma(2)}{\Gamma(1)} \times F_{v_1}(x)
\end{aligned}$$

Define another random variable $v_2 \sim \Gamma(1+2,\theta) = \Gamma(3,\theta)$; with CDF $F_{v_2}(x) = P(v_2 \leq x)$.

$$\begin{aligned}
\textbf{term 3} &= c \times (c_2) \times \int_0^x y^2 f_y(y)\, dy \\
&= c \times \left(\frac{\gamma_{11}}{2} \left(\frac{1}{\theta^2} \right) \right) \times \int_0^x y^2 f_y(y)\, dy \\
&= c \times \left(\frac{\gamma_{11}}{2} \left(\frac{1}{\theta^2} \right) \right) \times \int_0^x \left[\frac{1}{\Gamma(1)\theta^1} y^{(1+2)-1} e^{\frac{-y}{\theta}} \right] dy \\
&= c \times \left(\frac{\gamma_{11}}{2} \left(\frac{1}{\theta^2} \right) \right) \times \frac{\theta^3}{\theta} \times \frac{\Gamma(3)}{\Gamma(1)} \times \int_0^x \left[\frac{1}{\Gamma(3)\theta^3} y^{(3)-1} e^{\frac{-y}{\theta}} \right] dy \\
&= c \times \left(\frac{\gamma_{11}}{2} \left(\frac{1}{\theta^2} \right) \right) \times \theta^2 \times \frac{\Gamma(3)}{\Gamma(1)} \times F_{v_2}(x)
\end{aligned}$$

Beta distribution

Next, we consider when $f_y(y) \sim \text{Beta}(\alpha,\beta)$.

$$y \sim f(y;\alpha,\beta) = \frac{1}{B(\alpha,\beta)} y^{\alpha-1}(1-y)^{\beta-1} = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} y^{\alpha-1}(1-y)^{\beta-1}, \quad y \in [0,1]$$

- $\mu = E[y] = \frac{\alpha}{\alpha+\beta}$, and $\sigma^2 = Var(y) = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$

$$\begin{aligned} g_y(y) &= \Big[1+\frac{1}{2}\text{tr}(\Gamma)\Big]^{-1}\left[\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}y^{\alpha-1}(1-y)^{\beta-1}\right]\Big(1+\frac{\gamma_{11}}{2}\left[\frac{(y-\mu)^2}{\sigma^2}\right]+\frac{1}{2}\sum_{j=2}^{m}\gamma_{jj}\Big) \\ &= \Big[1+\frac{1}{2}\text{tr}(\Gamma)\Big]^{-1}\left[\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}y^{\alpha-1}(1-y)^{\beta-1}\right]\left(\Big(1+\frac{1}{2}\sum_{j=2}^{d}\gamma_{jj}\Big)+\frac{\gamma_{11}}{2}\left[\frac{y^2-2y\mu+\mu^2}{\sigma^2}\right]\right) \\ &= c\times f_y(y)\Big[\Big(c_0\Big)+\Big(c_1\Big)y+\Big(c_2\Big)y_2^2\Big] \end{aligned}$$

- Here $c_0 = \left(1+\frac{\gamma_{11}}{2}\left(\frac{\mu^2}{\sigma^2}\right)+\frac{1}{2}\sum_{j=2}^{d}\gamma_{jj}\right) = \left(1+\frac{\gamma_{11}}{2}\left(\frac{\alpha(\alpha+\beta+1)}{\beta}\right)+\frac{1}{2}\sum_{j=2}^{d}\gamma_{jj}\right)$
- $c_1 = \frac{\gamma_{11}}{2}\left(\frac{-2\mu}{\sigma^2}\right) = \frac{\gamma_{11}}{2}\left(\frac{-2(\alpha+\beta)(\alpha+\beta+1)}{\beta}\right)$
- $c_2 = \frac{\gamma_{11}}{2}\left(\frac{1}{\sigma^2}\right) = \frac{\gamma_{11}}{2}\left(\frac{(\alpha+\beta)^2(\alpha+\beta+1)}{\alpha\beta}\right)$

$y \sim \text{Beta}(\alpha,\beta)$ with CDF $F_Y(Y=x) = P(Y \le x)$

$$\begin{aligned} \textbf{term1} &= c\times(c_0)\int_0^x f_y(y)dy \\ &= c\times(c_0)\times F_y(x) \\ &= c\times\left(1+\frac{\gamma_{11}}{2}\left(\frac{\alpha(\alpha+\beta+1)}{\beta}\right)+\frac{1}{2}\sum_{j=2}^{d}\gamma_{jj}\right)\times F_y(x) \end{aligned}$$

Define a new random variable $v_1 \sim \text{Beta}(\alpha+1,\beta)$ with CDF $F_{v_1}(x) = P(v_1 \le x)$.

$$\begin{aligned} \textbf{term2} &= c\times(c_1)\times\int_0^x y f_y(y)dy \\ &= c\times\frac{\gamma_{11}}{2}\Big(\frac{-2(\alpha+\beta)(\alpha+\beta+1)}{\beta}\Big)\times\int_0^x y f_y(y)dy \\ &= c\times\frac{\gamma_{11}}{2}\Big(\frac{-2(\alpha+\beta)(\alpha+\beta+1)}{\beta}\Big)\times\int_0^x\left[\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}y^{(\alpha+1)-1}(1-y)^{\beta-1}\right]dy \\ &= c\times\frac{\gamma_{11}}{2}\Big(\frac{-2(\alpha+\beta)(\alpha+\beta+1)}{\beta}\Big)\times\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)}\frac{\Gamma(\alpha+1)}{\Gamma(\alpha+\beta+1)}\times\int_0^x f_{v_1}(y)dy \\ &= c\times\frac{\gamma_{11}}{2}\Big(\frac{-2(\alpha+\beta)(\alpha+\beta+1)}{\beta}\Big)\times\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)}\frac{\Gamma(\alpha+1)}{\Gamma(\alpha+\beta+1)}\times F_{v_1}(x) \end{aligned}$$

Define another random variable $v_2 \sim \text{Beta}(\alpha+2,\beta)$ with CDF $F_{v_2}(x) = P(v_2 \le x)$.

$$\begin{aligned}
\textbf{term 3} &= c \times (c_2) \times \int_0^x y^2 f_y(y)\,dy \\
&= c \times \frac{\gamma_{11}}{2}\Big(\frac{(\alpha+\beta)^2(\alpha+\beta+1)}{\alpha\beta}\Big) \times \int_0^x y^2 f_y(y)\,dy \\
&= c \times \frac{\gamma_{11}}{2}\Big(\frac{(\alpha+\beta)^2(\alpha+\beta+1)}{\alpha\beta}\Big) \times \int_0^x \left[\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} y^{(\alpha+2)-1}(1-y)^{\beta-1}\right] dy \\
&= c \times \frac{\gamma_{11}}{2}\Big(\frac{-2(\alpha+\beta)(\alpha+\beta+2)}{\beta}\Big) \times \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)}\frac{\Gamma(\alpha+2)}{\Gamma(\alpha+2+\beta)} \times \int_0^x f_{v_2}(y)dy \\
&= c \times \frac{\gamma_{11}}{2}\Big(\frac{(\alpha+\beta)^2(\alpha+\beta+2)}{\alpha\beta}\Big) \times \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)}\frac{\Gamma(\alpha+2)}{\Gamma(\alpha+2+\beta)} \times F_{v_2}(x)
\end{aligned}$$

3.10.2.4 Discrete Outcomes

When the densities $f_i(y_i)$ are discrete, it may be necessary to compute infinite sums involving these probabilities. For example, such sums occur naturally in numerical algorithms developed for Poisson, Geometric, negative binomial variate generation. From a practical standpoint, it is necessary to truncate these infinite sums after a finite number of terms.

Consider any random variable Z with nonnegative integer values, discrete density $p_i = \Pr(Z = i)$, and mean ν. The inverse method of random sampling reduces to a sequence of comparisons. We partition the interval $[0,1]$ into subintervals with the ith subinterval of length p_i. To sample Z, we draw a uniform random deviate U from $[0,1]$ and return the deviate j determined by the conditions $\sum_{i=1}^{j-1} p_i \le U < \sum_{i=1}^{j} p_i$. There is no need to invoke the distribution of Z. The process is most efficient when the largest p_i occur first. This suggests that we let k denote the least integer $\lfloor \nu \rfloor$ and rearrange the probabilities in the order $p_k, p_{k+1}, p_{k-1}, p_{k+2}, p_{k-2}, \ldots$ This tactic is apt put most of the probability mass first and render sampling efficient.

Poisson Distribution

A Poisson distribution describes the number of independent events occurring within a unit time interval, given the average rate of occurrence θ.

$$y \sim \text{Poisson}(\theta); f_y(y) = \frac{\theta^y e^{-\theta y}}{y!}, y = 0,1,2,3,...$$

- $\mu = E[y] = \theta = \sigma^2 = Var(y)$

$$\begin{aligned} g_y(y) &= \left[1 + \frac{1}{2}\text{tr}(\Gamma)\right]^{-1}\left[\frac{\theta^y e^{-\theta y}}{y!}\right]\left(1 + \frac{\gamma_{11}}{2}\left[\frac{(y-\mu)^2}{\sigma^2}\right] + \frac{1}{2}\sum_{j=2}^{d}\gamma_{jj}\right) \\ &= \left[1 + \frac{1}{2}\text{tr}(\Gamma)\right]^{-1}\left[\frac{\theta^y e^{-\theta y}}{y!}\right]\left(\left(1 + \frac{1}{2}\sum_{j=2}^{d}\gamma_{jj}\right) + \frac{\gamma_{11}}{2}\left[\frac{y^2 - 2y\mu + \mu^2}{\sigma^2}\right]\right) \\ &= c \times f_y(y)\left[\left(c_0\right) + \left(c_1\right)y + \left(c_2\right)y_2^2\right] \end{aligned}$$

- Here $c_0 = \left(1 + \frac{\gamma_{11}}{2}\left(\frac{\mu^2}{\sigma^2}\right) + \frac{1}{2}\sum_{j=2}^{m}\gamma_{jj}\right) = \left(1 + \frac{\gamma_{11}}{2}\left(1\right) + \frac{1}{2}\sum_{j=2}^{d}\gamma_{jj}\right)$
- $c_1 = \left(\frac{\gamma_{11}}{2}\left(\frac{-2\mu}{\sigma^2}\right)\right) = \left(\frac{\gamma_{11}}{2}\left(-2\right)\right)$
- $c_2 = \left(\frac{\gamma_{11}}{2}\left(\frac{1}{\sigma^2}\right)\right) = \left(\frac{\gamma_{11}}{2}\left(\frac{1}{\theta}\right)\right)$

Binomial Distribution

A Binomial distribution characterizes the number of successes in a sequence of independent trials. It has two parameters: ‘n‘, the number of trials, and ‘p‘, the probability of success in an individual trial, with the distribution:

$$y \sim \text{Binomial}(N, p); f_y(y) = \binom{N}{y} p^y (1-p)^{N-y}, y = 0,1,2,...,N$$

- $\mu = E[y] = Np; \sigma^2 = Var(y) = Np(1-p)$

$$\begin{aligned} g_y(y) &= \left[1+\frac{1}{2}\text{tr}(\mathbf{\Gamma})\right]^{-1}\left[\binom{N}{y}p^y(1-p)^{N-y}\right]\left(1+\frac{\gamma_{11}}{2}\left[\frac{(y-\mu)^2}{\sigma^2}\right]+\frac{1}{2}\sum_{j=2}^{m}\gamma_{jj}\right) \\ &= \left[1+\frac{1}{2}\text{tr}(\mathbf{\Gamma})\right]^{-1}\left[\binom{N}{y}p^y(1-p)^{N-y}\right]\left(\left(1+\frac{1}{2}\sum_{j=2}^{m}\gamma_{jj}\right)+\frac{\gamma_{11}}{2}\left[\frac{y^2-2y\mu+\mu^2}{\sigma^2}\right]\right) \\ &= c\times f_y(y)\left[\left(c_0\right)+\left(c_1\right)y+\left(c_2\right)y_2^2\right] \end{aligned}$$

- Here $c_0 = \left(1+\frac{\gamma_{11}}{2}\left(\frac{\mu^2}{\sigma^2}\right)+\frac{1}{2}\sum_{j=2}^{m}\gamma_{jj}\right) = \left(1+\frac{\gamma_{11}}{2}\left(\frac{(Np)^2}{Np(1-p)}\right)+\frac{1}{2}\sum_{j=2}^{m}\gamma_{jj}\right)$
- $c_1 = \left(\frac{\gamma_{11}}{2}\left(\frac{-2\mu}{\sigma^2}\right)\right) = \left(\frac{\gamma_{11}}{2}\left(\frac{-2Np}{Np(1-p)}\right)\right)$
- $c_2 = \left(\frac{\gamma_{11}}{2}\left(\frac{1}{\sigma^2}\right)\right) = \left(\frac{\gamma_{11}}{2}\left(\frac{1}{Np(1-p)}\right)\right)$

Geometric Distribution

A Geometric distribution characterizes the number of failures before the first success in a sequence of independent Bernoulli trials with success rate p.

$$y \sim \text{Geometric}(p); f_y(y) = (1-p)^y p, y = 0,1,2,...$$

- $\mu = E[y] = \frac{1}{p}; \sigma^2 = Var(y) = \frac{1-p}{p^2}$

$$\begin{aligned} g_y(y) &= \left[1+\frac{1}{2}\text{tr}(\mathbf{\Gamma})\right]^{-1}\left[(1-p)^y p\right]\left(1+\frac{\gamma_{11}}{2}\left[\frac{(y-\mu)^2}{\sigma^2}\right]+\frac{1}{2}\sum_{j=2}^{d}\gamma_{jj}\right) \\ &= \left[1+\frac{1}{2}\text{tr}(\mathbf{\Gamma})\right]^{-1}\left[(1-p)^y p\right]\left(\left(1+\frac{1}{2}\sum_{j=2}^{d}\gamma_{jj}\right)+\frac{\gamma_{11}}{2}\left[\frac{y^2-2y\mu+\mu^2}{\sigma^2}\right]\right) \\ &= c\times f_y(y)\left[\left(c_0\right)+\left(c_1\right)y+\left(c_2\right)y_2^2\right] \end{aligned}$$

- Here $c_0 = \left(1 + \frac{\gamma_{11}}{2}\left(\frac{\mu^2}{\sigma^2}\right) + \frac{1}{2}\sum_{j=2}^{d}\gamma_{jj}\right) = \left(1 + \frac{\gamma_{11}}{2}\left(\frac{\frac{1}{p}}{\frac{1-p}{p^2}}\right) + \frac{1}{2}\sum_{j=2}^{d}\gamma_{jj}\right)$
- $c_1 = \left(\frac{\gamma_{11}}{2}\left(\frac{-2\mu}{\sigma^2}\right)\right) = \left(\frac{\gamma_{11}}{2}\left(\frac{\frac{-2}{p}}{\frac{1-p}{p^2}}\right)\right)$
- $c_2 = \left(\frac{\gamma_{11}}{2}\left(\frac{1}{\sigma^2}\right)\right) = \left(\frac{\gamma_{11}}{2}\left(\frac{p^2}{1-p}\right)\right)$

Negative Binomial Distribution

A negative binomial distribution describes the number of failures before the ‘r‘th success in a sequence of independent Bernoulli trials. It is parameterized by ‘r‘, the number of successes, and ‘p‘, the probability of success in an individual trial.

$$y \sim \text{Negative Binomial}(r,p); f_y(y) = \binom{y+r-1}{y} p^r (1-p)^y, y = 0,1,2,...$$

- $\mu = E[y] = \frac{pr}{1-p}; \sigma^2 = Var(y) = \frac{pr}{(1-p)^2}$

$$\begin{aligned} g_y(y) &= \left[1 + \frac{1}{2}\text{tr}(\mathbf{\Gamma})\right]^{-1}\left[\binom{y+r-1}{y} p^r (1-p)^y\right]\left(1 + \frac{\gamma_{11}}{2}\left[\frac{(y-\mu)^2}{\sigma^2}\right] + \frac{1}{2}\sum_{j=2}^{d}\gamma_{jj}\right) \\ &= \left[1 + \frac{1}{2}\text{tr}(\mathbf{\Gamma})\right]^{-1}\left[\binom{y+r-1}{y} p^r (1-p)^y\right]\left(\left(1 + \frac{1}{2}\sum_{j=2}^{d}\gamma_{jj}\right) + \frac{\gamma_{11}}{2}\left[\frac{y^2 - 2y\mu + \mu^2}{\sigma^2}\right]\right) \\ &= c \times f_y(y)\left[\left(c_0\right) + \left(c_1\right)y + \left(c_2\right)y_2^2\right] \end{aligned}$$

- Here $c_0 = \left(1 + \frac{\gamma_{11}}{2}\left(\frac{\mu^2}{\sigma^2}\right) + \frac{1}{2}\sum_{j=2}^{d}\gamma_{jj}\right) = \left(1 + \frac{\gamma_{11}}{2}\left(\frac{\frac{pr}{1-p}}{\frac{pr}{(1-p)^2}}\right) + \frac{1}{2}\sum_{j=2}^{d}\gamma_{jj}\right)$
- $c_1 = \left(\frac{\gamma_{11}}{2}\left(\frac{-2\mu}{\sigma^2}\right)\right) = \left(\frac{\gamma_{11}}{2}\left(\frac{\frac{-2pr}{1-p}}{\frac{pr}{(1-p)^2}}\right)\right)$
- $c_2 = \left(\frac{\gamma_{11}}{2}\left(\frac{1}{\sigma^2}\right)\right) = \left(\frac{\gamma_{11}}{2}\left(\frac{(1-p)^2}{pr}\right)\right)$

3.10.3 Parameter Estimation:

We extend the Gaussian Base Model to accommodate densities in exponential family of distributions under the generalized linear model (GLM) framework. In this note, we pay close attention to the density-specific quantities which facilitate parameter estimation, and illustrate using the Poisson and Bernoulli density.

3.10.3.1 Fisher Scoring to Estimate Beta

$$\mathcal{L}(\boldsymbol{\beta}) = -\sum_{i=1}^{n} \ln\left[1+\frac{1}{2}\operatorname{tr}(\boldsymbol{\Gamma}_i)\right] + \sum_{i=1}^{n} \ln\left\{1+\frac{1}{2}\mathbf{r}_i(\boldsymbol{\beta})^t \boldsymbol{\Gamma}_i \mathbf{r}_i(\boldsymbol{\beta})\right\} \quad + \quad \sum_{i=1}^{n}\sum_{j=1}^{n_i} \ln f_{ij}(y_{ij} \mid \boldsymbol{\beta}) \quad (3.4)$$

For each distribution, the objective function is the loglikelihood (1), and can be viewed as three separate pieces. The last term of the loglikelihood is specific to the hypothesized density, and has first derivative, $\sum_{i=1}^{n}\sum_j \nabla \ln f_{ij}(y_{ij} \mid \boldsymbol{\beta})$, and second derivative $\nabla^2 L_n(\boldsymbol{\beta})$, that generalize to the exponential family of distributions.

The score (gradient of the loglikelihood) with respect to $\boldsymbol{\beta}$ is:

$$\nabla L_n(\boldsymbol{\beta}) = \sum_{i=1}^{n}\sum_j \nabla \ln f_{ij}(y_{ij} \mid \boldsymbol{\beta}) + \sum_{i=1}^{n} \frac{\nabla \mathbf{r}_i(\boldsymbol{\beta}) \boldsymbol{\Gamma}_i \mathbf{r}_i(\boldsymbol{\beta})}{1+\frac{1}{2}\mathbf{r}_i(\boldsymbol{\beta})^t \boldsymbol{\Gamma}_i \mathbf{r}_i(\boldsymbol{\beta})}, \quad (3.5)$$

The first term in the gradient, $\sum_{i=1}^{n}\sum_j \nabla \ln f_{ij}(y_{ij} \mid \boldsymbol{\beta})$, corresponds to the first derivative of the piece of the loglikelihood, specific to the hypothesized density. We can write this first term as a function of $\mathbf{W_{1i}}$, a diagonal matrix of "working weights".

$$\sum_{i=1}^{n}\sum_j \nabla \ln f_{ij}(y_{ij} \mid \boldsymbol{\beta}) \quad = \quad \sum_{i=1}^{n}\sum_{j=1}^{n_i} \frac{(y_{ij}-\mu_{ij})\mu'_{ij}(\eta_{ij})}{\sigma^2_{ij}} \mathbf{x}_{ij} = \sum_{i=1}^{n} \mathbf{X_i}^T \mathbf{W_{1i}} (\mathbf{Y_i} - \mu_i)$$

$$W_{1i} = \mathbf{Diagonal}\left(\frac{\mathbf{g}'(\mathbf{X_i^T}\beta)}{\mathbf{var}(\mathbf{Y_i}|\mu_\mathbf{i})}\right) = \begin{pmatrix} \frac{\mu'_{i1}(\eta_{i1})}{\sigma^2_{i1}} & 0 & \cdots & 0 \\ 0 & \frac{\mu'_{i2}(\eta_{i2})}{\sigma^2_{i2}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\mu'_{in_i}(\eta_{in_i})}{\sigma^2_{in_i}} \end{pmatrix}$$

Instead of using the exact Hessian, we will use the expected Fisher Information to get an approximation of the Hessian, which is clearly negative semi-definite.

$$-\sum_{i=1}^{n} \mathbf{X_i}^T \mathbf{W_{2i}} \mathbf{X_i} - \sum_{i=1}^{n} \frac{[\nabla \mathbf{r}_i(\boldsymbol{\beta})^t \boldsymbol{\Gamma}_i \mathbf{r}_i(\boldsymbol{\beta})][\nabla \mathbf{r}_i(\boldsymbol{\beta})^t \boldsymbol{\Gamma}_i \mathbf{r}_i(\boldsymbol{\beta})]^t}{\left[1 + \frac{1}{2}\mathbf{r}_i(\boldsymbol{\beta})^t \boldsymbol{\Gamma}_i \mathbf{r}_i(\boldsymbol{\beta})\right]^2} \tag{3.6}$$

Specifically, we approximate the second derivative of the piece of the loglikelihood particular to the hypothesized density, $\nabla^2 L_n(\beta)$. Using the Expected Fisher Information Matrix, we present this term as a function of another diagonal weight matrix, $\mathbf{W_{2i}}$.

$$\begin{aligned} \nabla^2 L_n(\beta) &= \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{[\mu'_{ij}(\eta_{ij})]^2}{\sigma^2_{ij}} \mathbf{x}_{ij}\mathbf{x}_{ij}^T - \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{(y_{ij}-\mu_{ij})\mu''_{ij}(\eta_{ij})}{\sigma^2_{ij}} \mathbf{x}_{ij}\mathbf{x}_{ij}^T \\ &\quad + \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{(y_{ij}-\mu_{ij})[\mu'_{ij}(\eta_{ij})]^2 (d\sigma^2_{ij}/d\mu_{ij})}{\sigma^4_{ij}} \mathbf{x}_{ij}\mathbf{x}_{ij}^T \\ \mathbf{FIM}_n(\beta) &= \mathbf{E}[-\nabla^2 L_n(\beta)] = -\sum_{i=1}^{n}\sum_{j=1}^{n_i} \frac{[\mu'_{ij}(\eta_{ij})]^2}{\sigma^2_{ij}} \mathbf{x}_{ij}\mathbf{x}_{ij}^T = -\sum_{i=1}^{n} \mathbf{X_i}^T \mathbf{W_{2i}} \mathbf{X_i}. \end{aligned}$$

$$W_{2i} = \mathbf{Diagonal}\left(\frac{\mathbf{g}'(\mathbf{X_i^T}\beta)^\mathbf{2}}{\mathbf{var}(\mathbf{Y_i}|\mu_\mathbf{i})}\right) = \begin{pmatrix} \frac{(\mu'_{i1}(\eta_{i1}))^2}{\sigma^2_{i1}} & 0 & \cdots & 0 \\ 0 & \frac{(\mu'_{i2}(\eta_{i2}))^2}{\sigma^2_{i2}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{(\mu'_{in_i}(\eta_{in_i}))^2}{\sigma^2_{in_i}} \end{pmatrix}$$

The score and approximate Hessian provide the ingredients for a kind of scoring algorithm for im-

proving $\boldsymbol{\beta}$ in our model. For each Newton update of the fixed effect parameter, $\boldsymbol{\beta}$, in addition to updating the residual vector, $\mathbf{r}_i(\boldsymbol{\beta})$, we must also update these weight matrices, W_{1i} and W_{2i}, in the update of the Score and Hessian. We can find these quantities easily by making the appropriate calls to the GLM package, GLM.jl.

Let y_{ij} represent the j^{th} outcome for person i, hypothesized to come from a non-normal density in the exponential family of distributions, $f_{ij}(y_{ij} \mid \boldsymbol{\beta})$. For each hypothesized density under the GLM framework, we have mean parameter $\mu_{ij}(\boldsymbol{\beta}) = g^{-1}(\eta_{ij}(\boldsymbol{\beta})) = g^{-1}(\mathbf{x_{ij}}\boldsymbol{\beta})$, and variance parameter $\sigma^2_{ij}(\boldsymbol{\beta})$. Using these quantities, we define $r_{ij}(\boldsymbol{\beta}), j \in [1, d_i]$ as the j^{th} entry in the standardized residual vector for observation or group i.

$$r_{ij}(\boldsymbol{\beta}) = \sqrt{\tau}(y_{ij} - \mu_{ij}(\boldsymbol{\beta})) = \frac{(y_{ij} - \mu_{ij}(\boldsymbol{\beta}))}{\sqrt{\sigma^2_{ij}(\boldsymbol{\beta})}} \in \mathbb{R} \tag{3.7}$$

Let $\nabla \mathbf{r}_i(\boldsymbol{\beta}) \in \mathbb{R}^{d_i \times p}$ denote the matrix of differentials of all n_i observations for the i^{th} individuals standardized residual vector $\mathbf{r}_i(\boldsymbol{\beta})$. This quantity is important for our score and hessian computation, which really helps the optimization algorithm to speed up convergence.

$$\nabla \mathbf{r}_i(\boldsymbol{\beta})^t \quad = \begin{pmatrix} \nabla r_{i1}(\boldsymbol{\beta}) & \nabla r_{i2}(\boldsymbol{\beta}) & \ldots & \nabla r_{id_i}(\boldsymbol{\beta}) \end{pmatrix}$$

For each of the $j \in [1, n_i]$ observations for the i^{th} individual, $\nabla \mathbf{r}_i(\boldsymbol{\beta})^t$ can be formed column by column, where $\nabla r_{ij}(\boldsymbol{\beta})$ denotes the j^{th} column. $\nabla \mu_{ij}(\boldsymbol{\beta})$ and $\nabla \sigma^2_{ij}(\boldsymbol{\beta})$ respectively reflect the derivative of the mean and variance of the hypothesized density, with respect to $\boldsymbol{\beta}$.

$$\nabla r_{ij}(\boldsymbol{\beta}) \quad = \quad -\frac{1}{\sigma_{ij}(\boldsymbol{\beta})} \nabla \mu_{ij}(\boldsymbol{\beta}) - \frac{1}{2} \frac{y_{ij} - \mu_{ij}(\boldsymbol{\beta})}{\sigma^3_{ij}(\boldsymbol{\beta})} \nabla \sigma^2_{ij}(\boldsymbol{\beta}) \in \mathbb{R}^p \tag{3.8}$$

$$\nabla\mu_{ij}(\boldsymbol{\beta}) = \frac{\partial\mu_{ij}(\boldsymbol{\beta})}{\partial\eta_{ij}(\boldsymbol{\beta})} * \frac{\partial\eta_{ij}(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}} = \begin{pmatrix} \frac{\partial\mu_{ij}(\boldsymbol{\beta})}{\partial\eta_{ij}(\boldsymbol{\beta})} * \frac{\partial\mathbf{x_{ij}}\boldsymbol{\beta}}{\partial\beta_1} \\ \vdots \\ \frac{\partial\mu_{ij}(\boldsymbol{\beta})}{\partial\eta_{ij}(\boldsymbol{\beta})} * \frac{\partial\mathbf{x_{ij}}\boldsymbol{\beta}}{\partial\beta_p} \end{pmatrix} = \frac{\partial\mu_{ij}(\boldsymbol{\beta})}{\partial\eta_{ij}(\boldsymbol{\beta})} * \begin{pmatrix} x_{ij_1} \\ x_{ij_2} \\ \vdots \\ x_{ij_p} \end{pmatrix} = \frac{\partial\mu_{ij}(\boldsymbol{\beta})}{\partial\eta_{ij}(\boldsymbol{\beta})} * \mathbf{x_{ij}} \in \mathbb{R}^p$$

$$\nabla\sigma^2_{ij}(\boldsymbol{\beta}) = \frac{\partial\sigma^2_{ij}(\boldsymbol{\beta})}{\partial\mu_{ij}(\boldsymbol{\beta})}\frac{\partial\mu_{ij}(\boldsymbol{\beta})}{\partial\eta_{ij}(\boldsymbol{\beta})}\frac{\partial\eta_{ij}(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}} = \frac{\partial\sigma^2_{ij}(\boldsymbol{\beta})}{\partial\mu_{ij}(\boldsymbol{\beta})}\frac{\partial\mu_{ij}(\boldsymbol{\beta})}{\partial\eta_{ij}(\boldsymbol{\beta})} * \mathbf{x_{ij}} \in \mathbb{R}^p$$

For the Gaussian base model, since the identity function is the appropriate canonical link, we have that $\mu_{ij}(\boldsymbol{\beta}) = \eta_{ij}(\boldsymbol{\beta}) = \mathbf{x_{ij}}\boldsymbol{\beta} = x_{ij_1} * \beta_1 + ... + x_{ij_p} * \beta_p$ where $\mathbf{x_{ij}}$ denotes the vector of p covariate values for j^{th} measurement of the i^{th} person.

$$\nabla\mu_{ij}(\boldsymbol{\beta}) = \frac{\partial\eta_{ij}(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}} = \begin{pmatrix} \frac{\partial\mathbf{x_{ij}}\boldsymbol{\beta}}{\partial\beta_1} \\ \vdots \\ \frac{\partial\mathbf{x_{ij}}\boldsymbol{\beta}}{\partial\beta_p} \end{pmatrix} = \begin{pmatrix} X_{ij_1} \\ X_{ij_2} \\ \vdots \\ X_{ij_p} \end{pmatrix} = \mathbf{x_{ij}} \in \mathbb{R}^p$$

$$\nabla\sigma^2_{ij}(\boldsymbol{\beta}) = \frac{\partial\sigma^2_{ij}(\boldsymbol{\beta})}{\partial\mu_{ij}(\boldsymbol{\beta})}\frac{\partial\mu_{ij}(\boldsymbol{\beta})}{\partial\eta_{ij}(\boldsymbol{\beta})}\frac{\partial\eta_{ij}(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}} = 0 * 1 * \mathbf{x_{ij}} = \mathbf{0} \in \mathbb{R}^p$$

In the table below, we derive the same quantities for the Normal, Poisson, Bernoulli and negative binomial distributions, under the appropriate canonical link function. The details of the derivation for the above table is below.

Distribution	$g(\mu_{ij}(\boldsymbol{\beta})) = \eta_{ij}(\boldsymbol{\beta})$	$\mu_{ij}(\boldsymbol{\beta}) \in \mathbb{R}$	$\sigma^2_{ij}(\boldsymbol{\beta}) \in \mathbb{R}$	$\nabla\mu_{ij}(\boldsymbol{\beta}) \in \mathbb{R}^p$	$\nabla\sigma^2_{ij}(\boldsymbol{\beta}) \in \mathbb{R}^p$
Normal	Identity Link	$\eta_{ij}(\boldsymbol{\beta})$	σ^2_{ij}	$\mathbf{x_i}$	$\mathbf{0}$
Poisson	Log Link	$e^{\eta_{ij}(\boldsymbol{\beta})}$	$\mu_{ij}(\boldsymbol{\beta})$	$e^{\eta_{ij}(\boldsymbol{\beta})} * \mathbf{x_i}$	$e^{\eta_{ij}(\boldsymbol{\beta})} * \mathbf{x_{ij}}$
Bernoulli	Logit Link	$\frac{e^{\eta_{ij}(\boldsymbol{\beta})}}{1+e^{\eta_{ij}(\boldsymbol{\beta})}}$	$\frac{e^{\eta_{ij}(\boldsymbol{\beta})}}{(1+e^{\eta_{ij}(\boldsymbol{\beta})})^2}$	$\frac{e^{\eta_{ij}(\boldsymbol{\beta})})}{(1+e^{\eta_{ij}(\boldsymbol{\beta})})^2} * \mathbf{x_{ij}}$	$\frac{e^{\eta_{ij}(\boldsymbol{\beta})}(1-e^{\eta_{ij}(\boldsymbol{\beta})})^2}{(1+e^{\eta_{ij}(\boldsymbol{\beta})})^2} * \mathbf{x_i}$
Negative Binomial	Log Link	$e^{\eta_{ij}(\boldsymbol{\beta})}$	$e^{\eta_{ij}(\boldsymbol{\beta})} * (1 + \frac{e^{\eta_{ij}(\boldsymbol{\beta})}}{r})$	$e^{\eta_{ij}(\boldsymbol{\beta})} * \mathbf{x_i}$	$(\frac{e^{\eta_{ij}(\boldsymbol{\beta})}}{r} + (1 + \frac{e^{\eta_{ij}(\boldsymbol{\beta})}}{r})) * e^{\eta_{ij}(\boldsymbol{\beta})} * \mathbf{x_i}$

3.10.3.2 MM-Algorithm to Update Variance Components

To update the variance components $\boldsymbol{\theta} = \{\theta_k, k \in [1,m]\}$, the relevant part of the loglikelihood is

$$f(\boldsymbol{\theta}) = \sum_{i=1}^{n} \ln(1+\boldsymbol{\theta}^t \mathbf{b}_i) - \sum_{i=1}^{n} \ln(1+\boldsymbol{\theta}^t \mathbf{c}_i) \tag{3.9}$$

by defining the vectors $\mathbf{b}_i$ and $\mathbf{c}_i$ with nonnegative components

$$\begin{aligned} \mathbf{b}_{ik} &= \frac{1}{2}\mathbf{r}_i(\boldsymbol{\beta})^t \Omega_{ik} \mathbf{r}_i(\boldsymbol{\beta}) \\ \mathbf{c}_{ik} &= \frac{1}{2}\operatorname{tr}(\Omega_{ik}). \end{aligned}$$

If f is a convex function of a vector $\mathbf{x}$, $f(\mathbf{x}) \geq f(\mathbf{x_0}) + (\mathbf{x}-\mathbf{x_0})\nabla f(\mathbf{x_0})$

$$\begin{aligned} \sum_{i=1}^{n} -ln(1+\boldsymbol{\theta}_r^t \mathbf{c}_i) &\geq -\sum_{i=1}^{n} ln(1+\boldsymbol{\theta}_r^t \mathbf{c}_i) + (1+\boldsymbol{\theta}^t \mathbf{c}_i - (1+\boldsymbol{\theta}_r^t \mathbf{c}_i) * \frac{-1}{1+\boldsymbol{\theta}_r^t \mathbf{c}_i} \\ &= \mathbf{c_1}^{(t)} - \sum_{i=1}^{n} \frac{1}{1+\boldsymbol{\theta}_r^t \mathbf{c}_i}(\boldsymbol{\theta}^t \mathbf{c}_i - \boldsymbol{\theta}_r^t \mathbf{c}_i) \\ &= -\sum_{i=1}^{n} \frac{1}{1+\boldsymbol{\theta}_r^t \mathbf{c}_i}(\boldsymbol{\theta}^t \mathbf{c}_i) + \mathbf{c}^{(t)} \end{aligned}$$

The first term is minorized by the Jensen's function

$$\sum_{i=1}^{n} \left[\frac{1}{1+\boldsymbol{\theta}_r^t \mathbf{b}_i} \ln\left(\frac{1+\boldsymbol{\theta}_r^t \mathbf{b}_i}{1}\right) + \sum_{k=1}^{m} \frac{\theta_{rk} b_{ik}}{1+\boldsymbol{\theta}_r^t \mathbf{b}_i} \ln\left(\frac{1+\boldsymbol{\theta}_r^t \mathbf{b}}{\theta_{rk} b_{ik}} \theta_{r+1,k} b_{ik}\right) \right].$$

Proof: $f(\mathbf{a})$ is a concave vector function if for any vectors $\mathbf{a_1}, \mathbf{a_2}, \lambda \in [0,1]$

$$\begin{aligned} & f(\lambda \mathbf{a_1} + (1-\lambda)\mathbf{a_2}) \\ = \; & f\Big(\frac{1}{1+\theta_r^t \mathbf{b}_i}(1+\theta_r^t \mathbf{b}_i) + \frac{\theta_r^t \mathbf{b}_i}{1+\theta_r^t \mathbf{b}_i}\frac{1+\theta_r^t \mathbf{b}_i}{\theta_r^t \mathbf{b}_i}\theta^t \mathbf{b}_i\Big) \\ = \; & f\Big(1+\theta^t \mathbf{b}_i\Big) \\ \geq \; & \lambda f(\mathbf{a_1}) + (1-\lambda) f(\mathbf{a_2}) \\ = \; & \frac{1}{1+\theta_r^t \mathbf{b}_i} ln(1+\theta_r^t \mathbf{b}_i) + \frac{\theta_r^t \mathbf{b}_i}{1+\theta_r^t \mathbf{b}_i} ln(\frac{1+\theta_r^t \mathbf{b}_i}{\theta_r^t \mathbf{b}_i}\theta^t \mathbf{b}_i) \end{aligned}$$

$$\sum_{i=1}^{n} ln(1+\theta^t \mathbf{b}_i) \geq \sum_{i=1}^{n} \left[\frac{1}{1+\theta_r^t \mathbf{b}_i} \ln\left(\frac{1+\theta_r^t \mathbf{b}_i}{1}\right) + \sum_{k=1}^{m} \frac{\theta_{rk} b_{ik}}{1+\theta_r^t \mathbf{b}_i} \ln\left(\frac{1+\theta_r^t \mathbf{b}}{\theta_{rk} b_{ik}} \theta_{r+1,k} b_{ik}\right) \right].$$

The sum of these two minorizations constitutes the surrogate $h(\mathbf{a} \mid \mathbf{a}_r)$.

$$h(\theta \mid \theta_r) = -\sum_{i=1}^{n} \frac{1}{1+\theta_r^t \mathbf{c}_i}(\theta^t \mathbf{c}_i) + \left[\frac{1}{1+\theta_r^t \mathbf{b}_i} \ln\left(\frac{1+\theta_r^t \mathbf{b}_i}{1}\right) + \sum_{k=1}^{m} \frac{theta_{rk} b_{ik}}{1+\theta_r^t \mathbf{b}_i} \ln\left(\frac{1+\theta_r^t \mathbf{b}}{\theta_{rk} b_{ik}} \theta_{r+1,k} b_{ik}\right) \right].$$

We can maximize the surrogate function by taking a derivative with respect to $a_k, k \in [1,m]$. The stationarity condition $\nabla h(\mathbf{a} \mid \mathbf{a}_r) = \mathbf{0}$ has components

$$\frac{\partial}{\partial \theta_k} h(\theta \mid \theta_r) = \sum_{i=1}^{n} \frac{-c_{ik}}{1+\mathbf{a}_r^t \mathbf{b}_i} + \frac{\theta_{rk} b_{ik}}{1+\theta_r^t \mathbf{b}_i} \frac{1}{\theta_{r+1,k}} := 0$$

with solution

$$\theta_{r+1,k} \;\; = \;\; \theta_{rk} \left(\frac{\sum_{i=1}^{n} \frac{b_{ik}}{1+\theta_r^t \mathbf{b}_i}}{\sum_{i=1}^{n} \frac{c_{ik}}{1+\theta_r^t \mathbf{c}_i}} \right).$$

3.10.4 Quasi-Newton Algorithm

Alternatively, we can estimate the mean and variance parameters joinly using the Quasi-Newton algorithm.

3.10.4.1 Score and Hessian

For the AR(1) model, $\boldsymbol{\theta} = \{\sigma^2, \rho\}$, the score (gradient of loglikelihood function) is

$$\begin{aligned}
\nabla_{\sigma^2}\mathcal{L} &= -\sum_{i=1}^{n} \frac{\frac{d_i}{2}}{1+\frac{d_i\sigma^2}{2}} + \sum_{i=1}^{n} \frac{\frac{1}{2}\mathbf{r}_i(\boldsymbol{\beta})^t\mathbf{V}_i(\rho)\mathbf{r}_i(\boldsymbol{\beta})}{1+\frac{\sigma^2}{2}\mathbf{r}_i(\boldsymbol{\beta})^t\mathbf{V}_i(\rho)\mathbf{r}_i(\boldsymbol{\beta})} \\
\nabla_{\rho}\mathcal{L} &= \sum_{i=1}^{n} \frac{1}{1+\frac{\sigma^2}{2}\mathbf{r}_i(\boldsymbol{\beta})^t\mathbf{V}_i(\rho)\mathbf{r}_i(\boldsymbol{\beta})} * \frac{\sigma^2}{2}\mathbf{r}_i(\boldsymbol{\beta})^t\nabla\mathbf{V}_i(\rho)\mathbf{r}_i(\boldsymbol{\beta}).
\end{aligned}$$

The approximate Hessian is

$$\begin{aligned}
d^2_{\sigma^2}\mathcal{L} &= \sum_{i=1}^{n} \frac{(\frac{d_i}{2})^2}{(1+\frac{d_i}{2}\sigma^2)^2} - \sum_{i=1}^{n} \frac{(\frac{1}{2}\mathbf{r}_i(\boldsymbol{\beta})^t\mathbf{V}_i(\rho)\mathbf{r}_i(\boldsymbol{\beta}))^2}{(1+\frac{\sigma^2}{2}\mathbf{r}_i(\boldsymbol{\beta})^t\mathbf{V}_i(\rho)\mathbf{r}_i(\boldsymbol{\beta}))^2} \\
d^2_{\rho}\mathcal{L} &= \sum_{i=1}^{n} \frac{1}{1+\frac{\sigma^2}{2}\mathbf{r}_i(\boldsymbol{\beta})^t\mathbf{V}_i(\rho)\mathbf{r}_i(\boldsymbol{\beta})} * \frac{\sigma^2}{2}\mathbf{r}_i(\boldsymbol{\beta})^t\nabla^2\mathbf{V}_i(\rho)\mathbf{r}_i(\boldsymbol{\beta}) \\
&- \sum_{i=1}^{n} \frac{1}{(1+\frac{\sigma^2}{2}\mathbf{r}_i(\boldsymbol{\beta})^t\mathbf{V}_i(\rho)\mathbf{r}_i(\boldsymbol{\beta}))^2} * \Big(\frac{\sigma^2}{2}\mathbf{r}_i(\boldsymbol{\beta})^t\nabla\mathbf{V}_i(\rho)\mathbf{r}_i(\boldsymbol{\beta})\Big)^2,
\end{aligned}$$

For the CS model, $\boldsymbol{\theta} = (\sigma^2, \rho)$, and the gradient is

$$\begin{aligned}
\nabla_{\sigma^2}\mathcal{L} &= -\sum_{i=1}^{n} \frac{\frac{d_i}{2}}{1+\frac{d_i\sigma^2}{2}} + \sum_{i=1}^{n} \frac{\frac{1}{2}\mathbf{r}_i(\boldsymbol{\beta})^t\mathbf{V}_i(\rho)\mathbf{r}_i(\boldsymbol{\beta})}{1+\frac{\sigma^2}{2}\mathbf{r}_i(\boldsymbol{\beta})^t\mathbf{V}_i(\rho)\mathbf{r}_i(\boldsymbol{\beta})} \\
\nabla_{\rho}\mathcal{L} &= \sum_{i=1}^{n} \frac{1}{1+\frac{\sigma^2}{2}\mathbf{r}_i(\boldsymbol{\beta})^t\mathbf{V}_i(\rho)\mathbf{r}_i(\boldsymbol{\beta})} * \frac{\sigma^2}{2}\mathbf{r}_i(\boldsymbol{\beta})^t\nabla\mathbf{V}_i(\rho)\mathbf{r}_i(\boldsymbol{\beta}).
\end{aligned}$$

The approximate Hessian is

$$\begin{aligned}
d^2_{\sigma^2}\mathcal{L} &= \sum_{i=1}^{n} \frac{(\frac{d_i}{2})^2}{(1+\frac{d_i}{2}\sigma^2)^2} - \sum_{i=1}^{n} \frac{(\frac{1}{2}\mathbf{r}_i(\boldsymbol{\beta})^t\mathbf{V}_i(\rho)\mathbf{r}_i(\boldsymbol{\beta}))^2}{(1+\frac{\sigma^2}{2}\mathbf{r}_i(\boldsymbol{\beta})^t\mathbf{V}_i(\rho)\mathbf{r}_i(\boldsymbol{\beta}))^2} \\
d^2_{\rho}\mathcal{L} &= -\sum_{i=1}^{n} \frac{1}{(1+\frac{\sigma^2}{2}\mathbf{r}_i(\boldsymbol{\beta})^t\mathbf{V}_i(\rho)\mathbf{r}_i(\boldsymbol{\beta}))^2} * \Big(\frac{\sigma^2}{2}\mathbf{r}_i(\boldsymbol{\beta})^t\nabla\mathbf{V}_i(\rho)\mathbf{r}_i(\boldsymbol{\beta})\Big)^2,
\end{aligned}$$

where $\nabla\mathbf{V}_i(\rho)$ and $\nabla^2\mathbf{V}_i(\rho)$ are, respectively, the element-wise first and second derivatives of the matrix $\mathbf{V}_i(\rho)$ with respect to ρ.

For the VM model the gradient is

$$\nabla_\theta f(\theta) = \sum_{i=1}^{n} \frac{1}{(1+\theta^t \mathbf{b}_i)} * \mathbf{b}_i - \sum_{i=1}^{n} \frac{1}{1+\theta^t \mathbf{c}_i} * \mathbf{c}_i.$$

The approximate Hessian is

$$\nabla^2_{\theta,\theta} f(\theta) = -\sum_{i=1}^{n} \frac{1}{(1+\theta^t \mathbf{b}_i)} * \mathbf{b}_i \mathbf{b}_i^t + \sum_{i=1}^{n} \frac{1}{1+\theta^t \mathbf{c}_i} * \mathbf{c}_i \mathbf{c}_i^t.$$

3.10.5 Negative Binomial

3.10.5.1 Estimating Nuisance Parameter

To estimate the nuisance parameter r in a Negative Binomial model, we use maximum likelihood. Because we are dealing with 1 parameter optimization, Newton's method is a good candidate due to its quadratic rate of convergence. The full loglikelihood is

$$-\sum_{i=1}^{n} \ln\left(1+\frac{1}{2}\mathrm{tr}(\Gamma_i)\right) + \sum_{i=1}^{n}\sum_{i=j}^{d_i} \ln f_{ij}(y_{ij} \mid \beta) + \sum_{i=1}^{n} \ln\left(1+\frac{1}{2}\mathbf{r}_i(\beta)^t \Gamma_i \mathbf{r}_i(\beta)\right)$$

where only the 2nd and 3rd term depends on r. First consider the 2nd term. Because $\mu_{ij} = \frac{r(1-p_{ij})}{p_{ij}}$, $p_{ij} = \frac{r}{r+\mu_{ij}}$, the 2nd term of the loglikelihood is

$$\begin{aligned}
&\sum_{i=1}^{n}\sum_{j=1}^{d_i} \ln\left[\binom{y_{ij}+r-1}{y_{ij}} p_{ij}^r (1-p_{ij})^{y_{ij}}\right] \\
=&\sum_{i=1}^{n}\sum_{j=1}^{d_i} \ln\binom{y_{ij}+r-1}{y_{ij}} + r\ln\left(\frac{r}{\mu_{ij}+r}\right) + y_{ij}\ln\left(\frac{\mu_{ij}}{\mu_{ij}+r}\right) \\
=&\sum_{i=1}^{n}\sum_{j=1}^{d_i} \ln((y_{ij}+r-1)!) - \ln(y_{ij}!) - \ln((r-1)!) + r\ln(r) - (r+y_{ij})\ln(\mu_{ij}+r) + y_{ij}\ln(\mu_{ij})
\end{aligned}$$

Let $\Psi^{(0)}$ be the digamma function and $\Psi^{(1)}$ the trigamma function, then the first and second derivative is

$$\sum_{i=1}^{n}\sum_{j=1}^{d_i}\Psi^{(0)}(y_{ij}+r)-\Psi^{(0)}(r)+1+\ln(r)-\frac{r+y_{ij}}{\mu_{ij}+r}-\ln(\mu_{ij}+r),$$
$$\sum_{i=1}^{n}\sum_{j=1}^{d_i}\Psi^{(1)}(y_{ij}+r)-\Psi^{(0)}(r)+\frac{1}{r}-\frac{2}{\mu_{ij}+r}+\frac{r+y_{ij}}{(\mu_{ij}+r)^2}.$$

Now consider the 3rd term of the full loglikelihood. First recall

$$\mathbf{D}_i = diagonal(\sqrt{var(\mathbf{y}_i)})$$
$$var(y_{ij}) = \frac{r(1-p_{ij})}{p_{ij}^2} = \frac{e^{\eta_{ij}}(e^{\eta_{ij}}+r)}{r}.$$

Using multiple chain rules,

$$\frac{d}{dr}\ln\left(1+\frac{1}{2}\mathbf{r}_i(\boldsymbol{\beta})^t\boldsymbol{\Gamma}_i\mathbf{r}_i(\boldsymbol{\beta})\right)=\sum_{i=1}^{n}\frac{\mathbf{r}_i(\boldsymbol{\beta})^t\boldsymbol{\Gamma}_i d\mathbf{r}_i}{1+\frac{1}{2}\mathbf{r}_i(\boldsymbol{\beta})^t\boldsymbol{\Gamma}_i\mathbf{r}_i(\boldsymbol{\beta})}$$
$$\frac{d^2}{dr^2}\ln\left(1+\frac{1}{2}\mathbf{r}_i(\boldsymbol{\beta})^t\boldsymbol{\Gamma}_i\mathbf{r}_i(\boldsymbol{\beta})\right)=\sum_{i=1}^{n}\frac{-[\mathbf{r}_i(\boldsymbol{\beta})^t\boldsymbol{\Gamma}_i d\mathbf{r}_i]^2}{[1+\frac{1}{2}\mathbf{r}_i(\boldsymbol{\beta})^t\boldsymbol{\Gamma}_i\mathbf{r}_i(\boldsymbol{\beta})]^2}+\frac{d\mathbf{r}(\boldsymbol{\beta})^t\boldsymbol{\Gamma}_i d\mathbf{r}_i(\boldsymbol{\beta})+\mathbf{r}(\boldsymbol{\beta})^t\boldsymbol{\Gamma}_i d\mathbf{r}_i^2(\boldsymbol{\beta})}{1+\frac{1}{2}\mathbf{r}_i(\boldsymbol{\beta})^t\boldsymbol{\Gamma}_i\mathbf{r}_i(\boldsymbol{\beta})}$$

where

$$\mathbf{r}_i(\boldsymbol{\beta}) = \mathbf{D}_i^{-1}(\mathbf{y}_i-\boldsymbol{\mu}_i)$$
$$d\mathbf{r}_i(\boldsymbol{\beta}) = -\mathbf{D}_i^{-1}d\mathbf{D}_i\mathbf{D}_i^{-1}(\mathbf{y}_i-\boldsymbol{\mu}_i)$$
$$d\mathbf{r}_i^2(\boldsymbol{\beta}) = [2\mathbf{D}_i^{-1}d\mathbf{D}_i\mathbf{D}_i^{-1}d\mathbf{D}_i\mathbf{D}_i^{-1}-\mathbf{D}_i^{-1}d^2\mathbf{D}_i\mathbf{D}_i^{-1}](\mathbf{y}_i-\boldsymbol{\mu}_i)$$
$$d\mathbf{D}_i = diagonal\left(\frac{d}{dr}\sqrt{\frac{e^{\eta_{ij}}(e^{\eta_{ij}}+r)}{r}}\right)=diagonal\left(\frac{-e^{2\eta_{ij}}}{2r^{1.5}\sqrt{e^{\eta_{ij}}(e^{\eta_{ij}}+r)}}\right)$$
$$d^2\mathbf{D}_i = diagonal\left(\frac{e^{3\eta}}{4r^{1.5}(e^{\eta}(e^{\eta}+r))^{1.5}}+\frac{3e^{2\eta}}{4r^{2.5}(e^{\eta}(e^{\eta}+r))^{0.5}}\right).$$

Note we used the identity $df(\mathbf{X})^{-1} = -f(\mathbf{X})^{-1}df(\mathbf{X})f(\mathbf{X})^{-1}$ for obtaining $d\mathbf{r}_i(\boldsymbol{\beta})$ and for obtaining $d\mathbf{r}_i^2(\boldsymbol{\beta})$, chain rule implies

$$\begin{aligned}
& d(f(\mathbf{X})^{-1}df(\mathbf{X})f(\mathbf{X})^{-1}) \\
=& [-f(\mathbf{X})^{-1}df(\mathbf{X})f(\mathbf{X})^{-1}]df(\mathbf{X})f(\mathbf{X})^{-1} + f(\mathbf{X})^{-1}d(df(\mathbf{X})f(\mathbf{X})^{-1}) \\
=& -f(\mathbf{X})^{-1}df(\mathbf{X})f(\mathbf{X})^{-1}df(\mathbf{X})f(\mathbf{X})^{-1} + \\
& f(\mathbf{X})^{-1}\left[df(\mathbf{X})(-f(\mathbf{X})^{-1}df(\mathbf{X})f(\mathbf{X})^{-1}) + d^2f(\mathbf{X})f(\mathbf{X})^{-1}\right] \\
=& -2f(\mathbf{X})^{-1}df(\mathbf{X})f(\mathbf{X})^{-1}df(\mathbf{X})f(\mathbf{X})^{-1} + f(\mathbf{X})^{-1}d^2f(\mathbf{X})f(\mathbf{X})^{-1}
\end{aligned}$$

In summary, we update the nuisance parameter r using Newton's update

$$r_{n+1} = r_n - \frac{\frac{d}{dr}L(r \mid \boldsymbol{\mu}, \boldsymbol{\Gamma}, \mathbf{y})}{\frac{d^2}{dr^2}L(r \mid \boldsymbol{\mu}, \boldsymbol{\Gamma}, \mathbf{y})}$$

where

$$\begin{aligned}
\frac{d}{dr}L(r \mid \boldsymbol{\mu}, \boldsymbol{\Gamma}, \mathbf{y}) &= \sum_{i=1}^{n}\sum_{j=1}^{n_i}\Psi^{(0)}(y_{ij}+r) - \Psi^{(0)}(r) + 1 + \ln(r) - \frac{r+y_{ij}}{\mu_{ij}+r} - \ln(\mu_{ij}+r) \\
&\quad + \sum_{i=1}^{n}\frac{\mathbf{r}_i(\boldsymbol{\beta})^t\boldsymbol{\Gamma}_i d\mathbf{r}_i}{1+\frac{1}{2}\mathbf{r}_i(\boldsymbol{\beta})^t\boldsymbol{\Gamma}_i\mathbf{r}_i(\boldsymbol{\beta})} \\
\frac{d^2}{dr^2}L(r \mid \boldsymbol{\mu}, \boldsymbol{\Gamma}, \mathbf{y}) &= \sum_{i=1}^{n}\sum_{j=1}^{n_i}\Psi^{(1)}(y_{ij}+r) - \Psi^{(0)}(r) + \frac{1}{r} - \frac{2}{\mu_{ij}+r} + \frac{r+y_{ij}}{(\mu_{ij}+r)^2} \\
&\quad - \sum_{i=1}^{n}\frac{[\mathbf{r}_i(\boldsymbol{\beta})^t\boldsymbol{\Gamma}_i d\mathbf{r}_i]^2}{[1+\frac{1}{2}\mathbf{r}_i(\boldsymbol{\beta})^t\boldsymbol{\Gamma}_i\mathbf{r}_i(\boldsymbol{\beta})]^2} + \frac{d\mathbf{r}_i(\boldsymbol{\beta})^t\boldsymbol{\Gamma}_i d\mathbf{r}_i(\boldsymbol{\beta}) + \mathbf{r}_i(\boldsymbol{\beta})^t\boldsymbol{\Gamma}_i d\mathbf{r}_i^2(\boldsymbol{\beta})}{1+\frac{1}{2}\mathbf{r}_i(\boldsymbol{\beta})^t\boldsymbol{\Gamma}_i\mathbf{r}_i(\boldsymbol{\beta})}
\end{aligned}$$

For stability, we need to (1) perform line-search and (2) set the second derivative equal to 1 if it is negative. By default, we allow for a maximum of 10 block iterations; In each block iteration, we allow for a maximum of 15 iterations for the quasi-newton update of $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$, and a maximum of 10 newton iterations for the update of r.

3.10.6 Compound Symmetric Covariance

Under the Compound Symmetric (CS) parameterization of $\mathbf{\Gamma}_i$,

$$\begin{aligned}\mathbf{\Gamma}_i &= \sigma^2 \times \left[\rho \mathbf{1}_{d_i}\mathbf{1}_{d_i}^t + (1-\rho)I_{d_i}\right] \\ &= \sigma^2 \times \mathbf{V}_i(\rho)\end{aligned}$$

3.10.6.1 Bounding Correlation Parameter

To ensure that the covariance matrix $\mathbf{\Gamma}_i$ is positive semi-definite, we will focus on $\mathbf{V}_i(\rho)$ and use an eigenvalue argument to bound $\rho \in (-\frac{1}{d_i-1}, 1)$. Let $\mathbf{v}$ be a vector of dimension d_i such that $< \mathbf{v}, \mathbf{v} >= 1$. We will find the conditions on ρ such that $\mathbf{v}^t\mathbf{V}_i(\rho)\mathbf{v} \geq 0$.

$$\begin{aligned}\mathbf{v}^t\mathbf{V}_i(\rho)\mathbf{v} &= \mathbf{v}^t\left[\rho \mathbf{1}_{d_i}\mathbf{1}_{d_i}^t + (1-\rho)I_{d_i}\right]\mathbf{v} \\ &= \rho\mathbf{v}^t\mathbf{1}_{d_i}\mathbf{1}_{d_i}^t\mathbf{v} + (1-\rho)\mathbf{v}^t\mathbf{v} \\ &= \rho(\mathbf{1}_{d_i}^t\mathbf{v})^2 + 1 - \rho \\ &= \rho\left((\mathbf{1}_{d_i}^t\mathbf{v})^2 - 1\right) + 1 \\ &\geq 0\end{aligned}$$

Now solving for ρ and using the Cauchy-Schwartz Inequality, we get

$$\begin{aligned}\rho &\geq \frac{-1}{\left((\mathbf{1}_{d_i}^t\mathbf{v})^2 - 1\right)} \\ &\geq \frac{-1}{(\mathbf{1}_{d_i}^t\mathbf{1}_{d_i}) * (\mathbf{v}^t\mathbf{v}) - 1} \\ &= \frac{-1}{d_i - 1}\end{aligned}$$

3.10.7 Gaussian Base

This section considers the special case of Gaussian base in the quasi-copula framework, and presents detailed derivations of data generation and estimation methods. The joint density of $\mathbf{y} \in \mathbb{R}^d$ is

$$\left(c+\frac{1}{2}\text{tr}\Gamma\right)^{-1}\left(\frac{1}{\sqrt{2\pi}\sigma_0}\right)^d e^{-\frac{\|\mathbf{y}-\mu\|_2^2}{2\sigma_0^2}}\left[c+\frac{1}{2\sigma_0^2}(\mathbf{y}-\mu)^T\Gamma(\mathbf{y}-\mu)\right]. \tag{3.10}$$

The parameter $c \geq 0$ tips the balance between the independent and dependent components.

3.10.7.1 Moments

In the Gaussian case, we have

$$\begin{aligned}
\mathbb{E}(y_i) &= \mu_i \\
\mathbf{Var}(y_i) &= \sigma_0^2\left(1+\frac{\gamma_{ii}}{c+\frac{1}{2}\text{tr}(\Gamma)}\right) \\
\mathbf{Cov}(y_i,y_j) &= \sigma_0^2\frac{\gamma_{ij}}{c+\frac{1}{2}\text{tr}(\Gamma)}, \\
\mathbf{Cor}(y_i,y_j) &= \frac{\gamma_{ij}}{\sqrt{(c+\frac{1}{2}\text{tr}(\Gamma)+\gamma_{ii})(c+\frac{1}{2}\text{tr}(\Gamma)+\gamma_{jj})}}.
\end{aligned}$$

In summary,

$$\mathbf{Cov}(\mathbf{y}) = \sigma_0^2\left[\mathbf{I}+\left(\frac{1}{c+\frac{1}{2}\text{tr}\Gamma}\right)\Gamma\right].$$

In the special case of $\Gamma = \sigma_1^2\mathbf{I}$, we have

$$\begin{aligned}
\mathbf{Var}(y_i) &= \sigma_0^2\left(1+\frac{\sigma_1^2}{c+\frac{n}{2}\sigma_1^2}\right)\mathbf{I} \\
\mathbf{Cov}(y_i,y_j) &= 0, \quad i \neq j.
\end{aligned}$$

In the regression model, we would keep the variance σ_0^2 parameter for more flexibility in modeling the variance.

3.10.7.2 Random number generation

If we are able to generate a residual vector $\mathbf{R}$ from the (standardized) Gaussian copula model

$$\left[1+\frac{1}{2}\text{tr}(\Gamma)\right]^{-1}\left(\frac{1}{\sqrt{2\pi}}\right)^{d} e^{-\frac{\|\mathbf{r}\|_2^2}{2}}\left(1+\frac{1}{2}\mathbf{r}^T\Gamma\mathbf{r}\right),$$

then $\mathbf{Y}=\sigma_0\mathbf{R}+\mu$ is a desired sample from density (3.10).

To generate a sample from the standardized Gaussian copula model, we first sample R_1 from its marginal distribution and then generate remaining components sequentially from the conditional distributions $R_k \mid R_1,\ldots,R_{k-1}$ for $k=2,\ldots,d$.

- To generate R_1 from its marginal density

$$\left[1+\frac{1}{2}\text{tr}(\Gamma)\right]^{-1}\frac{1}{\sqrt{2\pi}}e^{-\frac{r_1^2}{2}}\left[1+\frac{\gamma_{11}}{2}r_1^2+\frac{1}{2}\sum_{i=2}^{d}\gamma_{ii}\right], \tag{3.11}$$

  we recognize it as a mixture of three distributions Normal$(0,1)$, $\sqrt{\chi_3^2}$ and $-\sqrt{\chi_3^2}$ with mixing probabilities $\frac{1+0.5\sum_{i=2}^{d}\gamma_{ii}}{1+0.5\sum_{i=1}^{d}\gamma_{ii}}$, $\frac{0.25\gamma_{11}}{1+0.5\sum_{i=1}^{d}\gamma_{ii}}$ and $\frac{0.25\gamma_{11}}{1+0.5\sum_{i=1}^{d}\gamma_{ii}}$ respectively.

- Next we consider generating R_2 from the conditional distribution $R_2 \mid R_1$. Dividing the marginal distribution of (R_1,R_2)

$$\left[1+\frac{1}{2}\text{tr}(\Gamma)\right]^{-1}\left(\frac{1}{\sqrt{2\pi}}\right)^{2} e^{-\frac{r_1^2+r_2^2}{2}}\left(1+\frac{\gamma_{22}}{2}r_2^2+\gamma_{12}r_1r_2+\frac{\gamma_{11}}{2}r_1^2+\frac{1}{2}\sum_{i=3}^{d}\gamma_{ii}\right)$$

  by the marginal distribution of R_1 (3.11) yields the conditional density

$$\frac{\frac{1}{\sqrt{2\pi}}e^{-\frac{r_2^2}{2}}\left(1+\frac{\gamma_{11}}{2}r_1^2+\frac{1}{2}\sum_{i=3}^{d}\gamma_{ii}+\gamma_{12}r_1r_2+\frac{\gamma_{22}}{2}r_2^2\right)}{1+\frac{\gamma_{11}}{2}r_1^2+\frac{1}{2}\sum_{i=2}^{d}\gamma_{ii}},$$

  which unfortunately is not a mixture of standard distributions. However we can evaluate its

cumulative distribution function (CDF)

$$F(x) = \frac{\left(1 + \frac{\gamma_{11}}{2} r_1^2 + \frac{1}{2}\sum_{i=3}^{d} \gamma_{ii}\right) \Phi(x) - \gamma_{12} r_1 \phi(x) + \frac{\gamma_{22}}{2}\left[\frac{1}{2} + \frac{\mathrm{sgn}(x)}{2} F_{\chi_3^2}(x^2)\right]}{1 + \frac{\gamma_{11}}{2} r_1^2 + \frac{1}{2}\sum_{i=2}^{d} \gamma_{ii}}$$

in terms of the density ϕ and CDF Φ of standard normal and the CDF $F_{\chi_3^2}$ of chi-squared distribution with degree of freedom 3. This suggests the inverse CDF approach. To generate one sample from $R_2 \mid R_1$, we draw a uniform variate U and use nonlinear root finding to locate R_2 such that $F(R_2) = U$.

- In general, the conditional distribution $R_k \mid R_1, \ldots, R_{k-1}$ has density

$$\frac{\frac{1}{\sqrt{2\pi}} e^{-\frac{r_k^2}{2}} \left(1 + \frac{1}{2}\mathbf{r}_{[k-1]}^T \Gamma_{[k-1],[k-1]} \mathbf{r}_{[k-1]} + \frac{1}{2}\sum_{i=k+1}^{n} \gamma_{ii} + (\sum_{i=1}^{k-1} r_i \gamma_{ik}) r_k + \frac{\gamma_{kk}}{2} r_k^2\right)}{1 + \frac{1}{2}\mathbf{r}_{[k-1]}^T \Gamma_{[k-1],[k-1]} \mathbf{r}_{[k-1]} + \frac{1}{2}\sum_{i=k}^{d} \gamma_{ii}}$$

and CDF

$$\frac{\left(1 + \frac{1}{2}\mathbf{r}_{[k-1]}^T \Gamma_{[k-1],[k-1]} \mathbf{r}_{[k-1]} + \frac{1}{2}\sum_{i=k+1}^{d} \gamma_{ii}\right) \Phi(x) - (\sum_{i=1}^{k-1} r_i \gamma_{ik}) \phi(x) + \frac{\gamma_{kk}}{2}\left[\frac{1}{2} + \frac{\mathrm{sgn}(x)}{2} F_{\chi_3^2}(x^2)\right]}{1 + \frac{1}{2}\mathbf{r}_{[k-1]}^T \Gamma_{[k-1],[k-1]} \mathbf{r}_{[k-1]} + \frac{1}{2}\sum_{i=k}^{d} \gamma_{ii}}.$$

We apply the inverse CDF approach to sample R_k given $R_1, \ldots, R_{k-1}$.

For a general GLM model, we need to sample from conditional densities of form $cf(y)(a_0 + a_1 y + a_2 y^2)$ where a_i, $i = 1,2,3$, are constants and c is a normalizing constant. For most continuous distributions, e.g., exponential, gamma, beta, chi-squared, and beta, the CDF can be expressed conveniently using special functions.

3.10.7.3 Parameter Estimation

Suppose we have n independent realizations $\mathbf{y}_i$ from the quasi-copula density. Each of these may be of different dimensions, d_i. Assuming the component distribution $\mathbf{y}_i \sim \mathrm{Normal}(\mathbf{X}_i\beta, \sigma_0^2 \mathbf{I}_{d_i})$, the

component densities take form

$$\ln f_i(\mathbf{y}_i \mid \beta, \sigma_0^2) = -\frac{d_i}{2}\ln 2\pi - \frac{d_i}{2}\ln\sigma_0^2 - \frac{1}{2}\frac{\|\mathbf{y}_i - \mathbf{X}_i\beta\|_2^2}{\sigma_0^2}$$

and the joint loglikelihood of the sample is

$$\begin{aligned}
& -\sum_i \ln\left(c + \frac{1}{2}\mathrm{tr}(\Gamma_i)\right) - \frac{\sum_i d_i}{2}\ln 2\pi - \frac{\sum_i d_i}{2}\ln\sigma_0^2 - \frac{1}{2}\frac{\sum_i \|\mathbf{y}_i - \mathbf{X}_i\beta\|_2^2}{\sigma_0^2} \\
& + \sum_i \ln\left[c + \frac{1}{2\sigma_0^2}(\mathbf{y}_i - \mathbf{X}_i\beta)^T\Gamma_i(\mathbf{y}_i - \mathbf{X}_i\beta)\right] \\
= & -\sum_i \ln\left(c + \frac{1}{2}\mathrm{tr}(\Gamma_i)\right) - \frac{\sum_i d_i}{2}\ln 2\pi + \frac{\sum_i d_i}{2}\ln\tau - \frac{\tau}{2}\sum_i \|\mathbf{y}_i - \mathbf{X}_i\beta\|_2^2 \\
& + \sum_i \ln\left[c + \frac{\tau}{2}(\mathbf{y}_i - \mathbf{X}_i\beta)^T\Gamma_i(\mathbf{y}_i - \mathbf{X}_i\beta)\right]
\end{aligned}$$

where $\Gamma_i = \sum_{k=1}^m \theta_k \mathbf{V}_{ik}$ are parameterized via variance components $\theta = (\theta_1, \ldots, \theta_m)$. We work with the parameterization $\tau = \sigma_0^{-2}$ because the loglikelihood is concave in τ.

3.10.7.4 Score and Hessian

The score (gradient of loglikelihood function) is

$$\begin{aligned}
\nabla_\beta &= \sigma_0^{-2}\sum_i \mathbf{X}_i^T(\mathbf{y}_i - \mathbf{X}_i\beta) - \sum_i \frac{\mathbf{X}_i^T\Gamma_i(\mathbf{y}_i - \mathbf{X}_i\beta)}{c\sigma_0^2 + \frac{1}{2}(\mathbf{y}_i - \mathbf{X}_i\beta)^T\Gamma_i(\mathbf{y}_i - \mathbf{X}_i\beta)} \\
&= \tau\sum_i \mathbf{X}_i^T(\mathbf{y}_i - \mathbf{X}_i\beta) - \tau\sum_i \frac{\mathbf{X}_i^T\Gamma_i(\mathbf{y}_i - \mathbf{X}_i\beta)}{c + \frac{\tau}{2}(\mathbf{y}_i - \mathbf{X}_i\beta)^T\Gamma_i(\mathbf{y}_i - \mathbf{X}_i\beta)} \\
\nabla_\tau &= \frac{\sum_i d_i}{2\tau} - \frac{1}{2}\sum_i \|\mathbf{y}_i - \mathbf{X}_i\beta\|_2^2 + \sum_i \frac{\frac{1}{2}(\mathbf{y}_i - \mathbf{X}_i\beta)^T\Gamma_i(\mathbf{y}_i - \mathbf{X}_i\beta)}{c + \frac{\tau}{2}(\mathbf{y}_i - \mathbf{X}_i\beta)^T\Gamma_i(\mathbf{y}_i - \mathbf{X}_i\beta)} \\
\nabla_c &= \sum_i \frac{1}{c + \frac{\tau}{2}(\mathbf{y}_i - \mathbf{X}_i\beta)^T\Gamma_i(\mathbf{y}_i - \mathbf{X}_i\beta)} \\
\nabla_\theta &= -\sum_i \left(c + \sum_k \theta_k t_{ik}\right)^{-1}\mathbf{t}_i + \tau\sum_i \left(c + \sum_k \theta_k q_{ik}\right)^{-1}\mathbf{q}_i
\end{aligned}$$

where

$$
\begin{aligned}
t_{ik} &= \frac{1}{2}\mathrm{tr}(\mathbf{V}_{ik}), \quad \mathbf{t}_i = (t_{i1}, \ldots, t_{im})^T \\
q_{ik} &= \frac{1}{2}(\mathbf{y}_i - \mathbf{X}_i\beta)^T \mathbf{V}_{ik}(\mathbf{y}_i - \mathbf{X}_i\beta), \quad \mathbf{q}_i = (q_{i1}, \ldots, q_{im})^T.
\end{aligned}
$$

The Hessian is

$$
\begin{aligned}
\nabla^2_{\beta,\beta} &= -\tau \sum_i \mathbf{X}_i^T \mathbf{X}_i + \sum_i \frac{\tau \mathbf{X}_i^T \Gamma_i \mathbf{X}_i}{c + \frac{\tau}{2}(\mathbf{y}_i - \mathbf{X}_i\beta)^T \Gamma_i (\mathbf{y}_i - \mathbf{X}_i\beta)} \\
&\quad - \sum_i \frac{\tau^2 [\mathbf{X}_i^T \Gamma_i (\mathbf{y}_i - \mathbf{X}_i\beta)][\mathbf{X}_i^T \Gamma_i (\mathbf{y}_i - \mathbf{X}_i\beta)]^T}{\left[c + \frac{\tau}{2}(\mathbf{y}_i - \mathbf{X}_i\beta)^T \Gamma_i (\mathbf{y}_i - \mathbf{X}_i\beta)\right]^2} \\
&\approx -\tau \sum_i \mathbf{X}_i^T \mathbf{X}_i - \tau \sum_i \frac{\tau [\mathbf{X}_i^T \Gamma_i (\mathbf{y}_i - \mathbf{X}_i\beta)][\mathbf{X}_i^T \Gamma_i (\mathbf{y}_i - \mathbf{X}_i\beta)]^T}{\left[c + \frac{\tau}{2}(\mathbf{y}_i - \mathbf{X}_i\beta)^T \Gamma_i (\mathbf{y}_i - \mathbf{X}_i\beta)\right]^2} \\
\nabla^2_{\beta,\tau} &= \sum_i \mathbf{X}_i^T (\mathbf{y}_i - \mathbf{X}_i\beta) - \sum_i \frac{\mathbf{X}_i^T \Gamma_i (\mathbf{y}_i - \mathbf{X}_i\beta)}{\left[c + \frac{\tau}{2}(\mathbf{y}_i - \mathbf{X}_i\beta)^T \Gamma_i (\mathbf{y}_i - \mathbf{X}_i\beta)\right]^2} \\
\nabla^2_{\beta,\theta} &= \sum_i \frac{\mathbf{X}_i^T \Gamma_i (\mathbf{y}_i - \mathbf{X}_i\beta)\mathbf{q}_i^T}{\left[c + \frac{\tau}{2}(\mathbf{y}_i - \mathbf{X}_i\beta)^T \Gamma_i (\mathbf{y}_i - \mathbf{X}_i\beta)\right]^2} \\
\nabla^2_{\tau,\tau} &= -\frac{\sum_i d_i}{2\tau^2} - \sum_i \left[\frac{\frac{1}{2}(\mathbf{y}_i - \mathbf{X}_i\beta)^T \Gamma_i (\mathbf{y}_i - \mathbf{X}_i\beta)}{c + \frac{\tau}{2}(\mathbf{y}_i - \mathbf{X}_i\beta)^T \Gamma_i (\mathbf{y}_i - \mathbf{X}_i\beta)}\right]^2 \\
\nabla^2_{\tau,\theta} &= -\sum_i \frac{\mathbf{X}_i^T \Gamma_i (\mathbf{y}_i - \mathbf{X}_i\beta)}{\left[c + \frac{\tau}{2}(\mathbf{y}_i - \mathbf{X}_i\beta)^T \Gamma_i (\mathbf{y}_i - \mathbf{X}_i\beta)\right]^2} \mathbf{q}_i^T \\
\nabla^2_{\theta,\theta} &= \sum_i \left(c + \sum_k \theta_k t_{ik}\right)^{-2} \mathbf{t}_i \mathbf{t}_i^T - \tau \sum_i \left(c + \sum_k \theta_k q_{ik}\right)^{-2} \mathbf{q}_i \mathbf{q}_i^T.
\end{aligned}
$$

Note $\mathbb{E}[\nabla^2_{\beta,\tau}]$, $\mathbb{E}[\nabla^2_{\beta,\theta}]$, and $\mathbb{E}[\nabla^2_{\tau,\theta}]$ are approximately zero.

3.10.7.5 MM algorithm

Because the MM update of θ and τ is cheap, we maximize the profiled likelihood. That is, after each Newton update of β, we update (τ, θ) conditional on current β using the MM algorithm and evaluate the gradient and (approximate) Hessian using the newest (τ, θ). To update τ and θ given

β, the relevant objective function is

$$-\sum_i \ln\left(c+\sum_k \theta_k t_{ik}\right)+\frac{\sum_i d_i}{2}\ln\tau-\frac{\sum_i r_i^2}{2}\tau+\sum_i \ln\left(c+\tau\sum_k \theta_k q_{ik}\right),$$

which is minorized by

$$\begin{aligned}
&- \sum_i\sum_k \frac{t_{ik}}{c^{(t)}+\sum_k \theta_k^{(t)} t_{ik}}\theta_k-\sum_i \frac{1}{c^{(t)}+\sum_k \theta_k^{(t)} t_{ik}}c\\
&+ \frac{\sum_i d_i}{2}\ln\tau-\frac{\sum_i r_i^2}{2}\tau\\
&+ \sum_i\sum_k \frac{\tau^{(t)}\theta_k^{(t)} q_{ik}}{c^{(t)}+\tau^{(t)}\sum_k \theta_k^{(t)} q_{ik}}(\ln\tau+\ln\theta_k)\\
&+ \sum_i \frac{c^{(t)}}{c^{(t)}+\tau^{(t)}\sum_k \theta_k^{(t)} q_{ik}}\ln c\\
&+ \text{const.}
\end{aligned}$$

The resultant updates are

$$\begin{aligned}
\tau^{(t+1)} &= \frac{\sum_i d_i+2\sum_i \frac{\tau^{(t)} q_i^{(t)}}{c^{(t)}+\tau^{(t)} q_i^{(t)}}}{\sum_i r_i^2}\\
c^{(t+1)} &= c^{(t)}\frac{\sum_i \frac{1}{c^{(t)}+\tau^{(t)} q_i^{(t)}}}{\sum_i \frac{1}{c^{(t)}+t^{(t)}}} \qquad (3.12)\\
\theta_k^{(t+1)} &= \theta_k^{(t)}\frac{\sum_i \frac{\tau^{(t)} q_{ik}}{c^{(t)}+\tau^{(t)} q_i^{(t)}}}{\sum_i \frac{t_{ik}}{c^{(t)}+t_i^{(t)}}}, \quad k=1,\ldots,m, \qquad (3.13)
\end{aligned}$$

where $q_i^{(t)}=\sum_k \theta_k^{(t)} q_{ik}$ and $t_i^{(t)}=\sum_k \theta_k^{(t)} t_{ik}$.

If we opt to use the optimal quadratic minorization

$$\ln(1+x)\geq \ln(1+x^{(t)})+(x-x^{(t)})-\frac{x^2-x^{2(t)}}{2(1+x^{(t)})},$$

the minorization function becomes

$$-\sum_i \sum_k \frac{t_{ik}}{1+\sum_k \theta_k^{(t)} t_{ik}} \theta_k + \frac{\sum_i d_i}{2} \ln \tau + \left(\sum_i \sum_k \theta_k q_{ik} - \frac{\sum_i r_i^2}{2} \right) \tau - \frac{\tau^2}{2} \sum_i \frac{(\sum_k \theta_k q_{ik})^2}{1+\tau^{(t)} \sum_k \theta_k^{(t)} q_{ik}} + c^{(t)}.$$

To update τ given θ_k, let

$$\begin{aligned} a^{(t)} &= \sum_i \frac{(\sum_k \theta_k^{(t)} q_{ik})^2}{1+\tau^{(t)} \sum_k \theta_k^{(t)} q_{ik}} \\ b^{(t)} &= \sum_i \sum_k \theta_k^{(t)} q_{ik} - \frac{\sum_i r_i^2}{2} \\ c^{(t)} &= \frac{\sum_i d_i}{2} \end{aligned}$$

then

$$\tau^{(t+1)} = \frac{b^{(t)} + \sqrt{b^{2(t)} + 4a^{(t)} c^{(t)}}}{2a^{(t)}}.$$

To update θ_k given τ, we minimize quadratic function

$$\frac{1}{2} \sigma^{2T} \mathbf{Q}^T \mathbf{W}^{(\mathbf{t})} \mathbf{Q} \sigma^2 - \mathbf{c}^{(t)T} \sigma^2$$

subject to nonnegativity constraint $\theta_k \geq 0$, where $\mathbf{W}^{(t)} = \text{diag}(w_1^{(t)}, \ldots, w_n^{(t)})$ with

$$w_i^{(t)} = \frac{\tau^{2(t)}}{1+\tau^{(t)} \sum_k \theta_k^{(t)} q_{ik}}$$

and $\mathbf{c}^{(t)}$ has entries

$$c_k^{(t)} = \tau^{(t)} \sum_i q_{ik} - \sum_i \frac{t_{ik}}{1+\sum_k \theta_k^{(t)} t_{ik}}.$$

It turns out this update based on quadratic minorization converges slower than the update (3.13) based on Jensen's inequality.

3.10.8 Additional Simulation Study Results

In each simulation scenario, the non-intercept entries of the predictor matrix $\mathbf{X}_i$ are independent standard normal deviates. When simulating under our model for the CS and AR(1) covariance structures, the true regression coefficients $\boldsymbol{\beta}_{\text{true}} \sim \text{Uniform}(-2,2)$. When comparing estimates with `MixedModels.jl` under the random intercept model for the Poisson, Bernoulli and negative binomial base, smaller regression coefficients $\boldsymbol{\beta}_{\text{true}} \sim \text{Uniform}(-0.2,0.2)$ hold. For Gaussian base, all precisions $\tau_{\text{true}} = 100$. For the negative binomial base, all dispersion parameters are $\mathbf{r}_{\text{true}} = 10$. Under both CS and AR(1) parameterizations of $\boldsymbol{\Gamma}_i$, $\sigma^2_{\text{true}} = 0.5$ and $\rho_{\text{true}} = 0.5$. Each simulation scenario was run on 100 replicates for each sample size $n \in \{100, 1000, 10000\}$ and number of observations $d_i \in \{2,5,10,15,20,25\}$ per independent sampling unit. By default, convergence tolerances are set to 10^{-6}.

Under the VC parameterization of $\boldsymbol{\Gamma}_i$, the choice $\boldsymbol{\Gamma}_{i,\text{true}} = \theta_{\text{true}} \times \mathbf{1}_{d_i}\mathbf{1}_{d_i}^t$ allows us to compare to the random intercept GLMM fit using `MixedModels.jl`. When the random effect term is a scalar, `MixedModels.jl` uses Gaussian quadrature for parameter estimation. We compare estimates and run-times to the random intercept GLMM fit of `MixedModels.jl` with 25 Gaussian quadrature points. We conduct simulation studies under two scenarios (simulation I and II). In simulation I, it is assumed that the data are generated by the quasi-copula model with $\theta_{\text{true}} = 0.1$, and in simulation II, it is assumed that the true distribution is the random intercept GLMM with $\theta_{\text{true}} = 0.01, 0.05$.

Figures 3.8 - 3.11 summarize the performance of the MLEs using mean squared errors (MSE) under the AR(1) parameterization of $\boldsymbol{\Gamma}_i$. Figures 3.12 - 3.15 summarize the same under the CS parameterization of $\boldsymbol{\Gamma}_i$. Figures 3.16 - 3.17 help us assess estimation accuracy and how well the GLMM density approximates the quasi-copula density under simulation I for the Bernoulli and Gaussian base. Under simulation II, Figures 3.18 - 3.21 shed light on how well the quasi-copula density approximates the GLMM density under different magnitudes of variance components. Figure 3.18 shows that for the Bernoulli base distribution with $\theta_{\text{true}} = 0.05$, the quasi-copula estimates of the variance component has average MSE of about 10^{-3}. Figure 3.19 shows that for the Bernoulli base

distribution with $\theta_{\text{true}} = 0.01$, the quasi-copula estimates of the variance component improves to an average MSE of about 10^{-4}. Figure 3.20 shows for the Gaussian base distribution with $\theta_{\text{true}} = 0.05$, the quasi-copula estimates of the variance component has an average MSE around 10^{-4}, and the quasi-copula estimates of the precision has an average MSE around 10^{-2}. Figure 3.21 shows that for the Gaussian base distribution with $\theta_{\text{true}} = 0.01$, the quasi-copula estimates of the variance component improves to an average MSE of about 10^{-6}, and the quasi-copula estimates of the precision improves to an average MSE of about 10^{-4}. The QC model accurately estimates the mean components even with large cluster sizes ($d_i = 25$) and small sample sizes ($n = 100$), even when the true density is that of the GLMM and LMM.

3.10.8.1 AR(1) Covariance

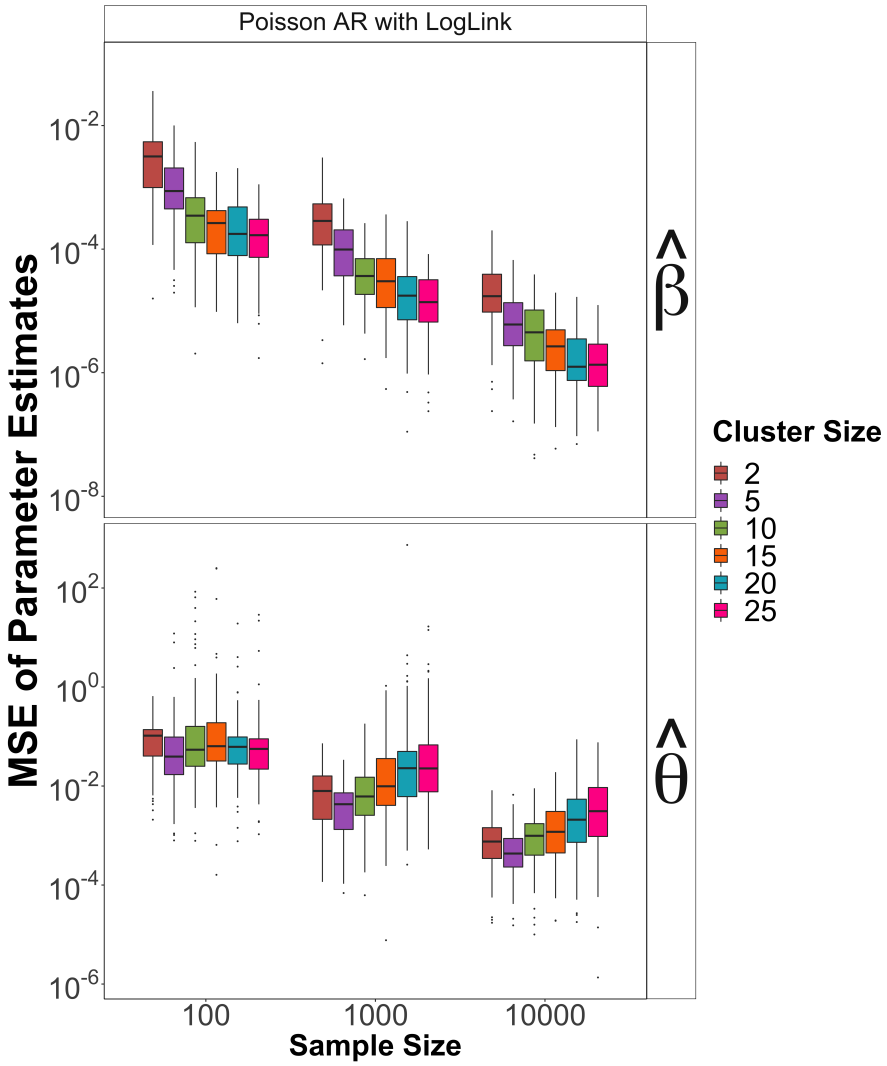

Figure 3.8: Mean squared errors (MSE) of parameter estimates $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ under the AR(1) covariance for the Poisson base distribution with log link function. Each scenario reports involves 100 replicates.

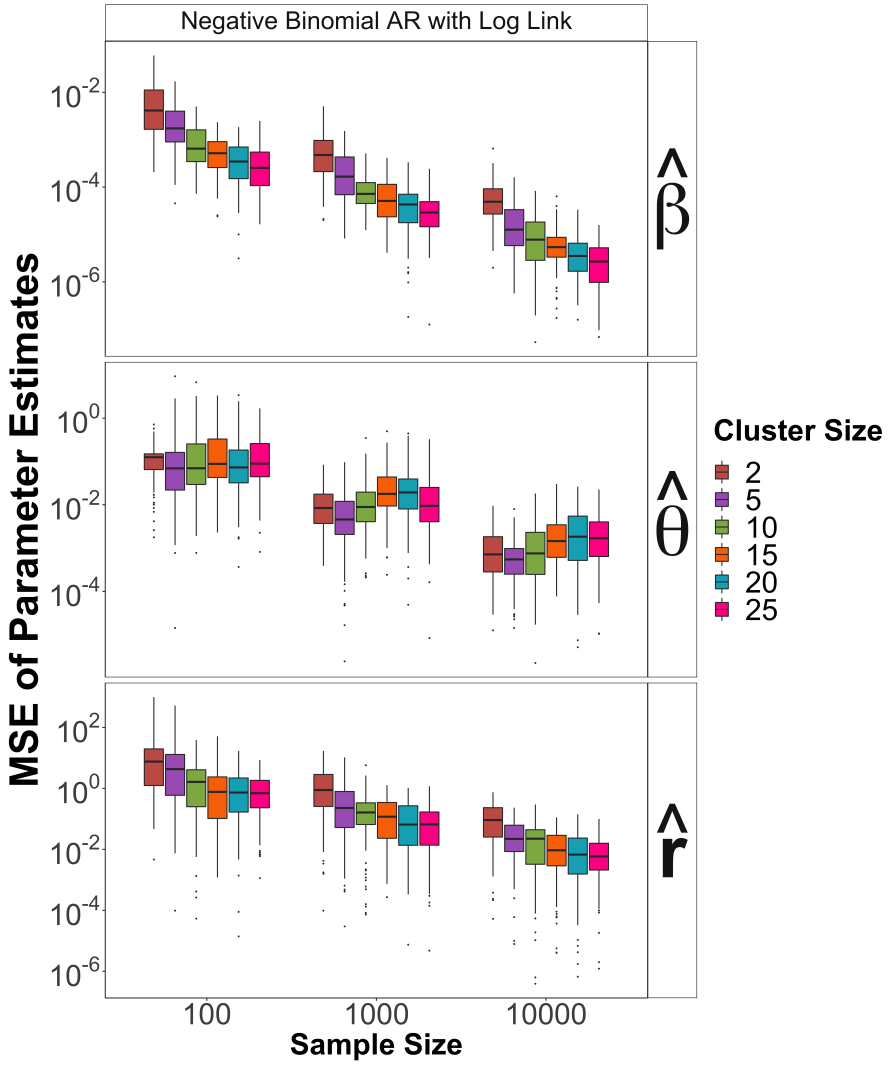

Figure 3.9: Mean squared errors (MSE) of parameter estimates β and θ under the AR(1) covariance for the negative binomial base distribution with log link function. Each scenario reports involves 100 replicates.

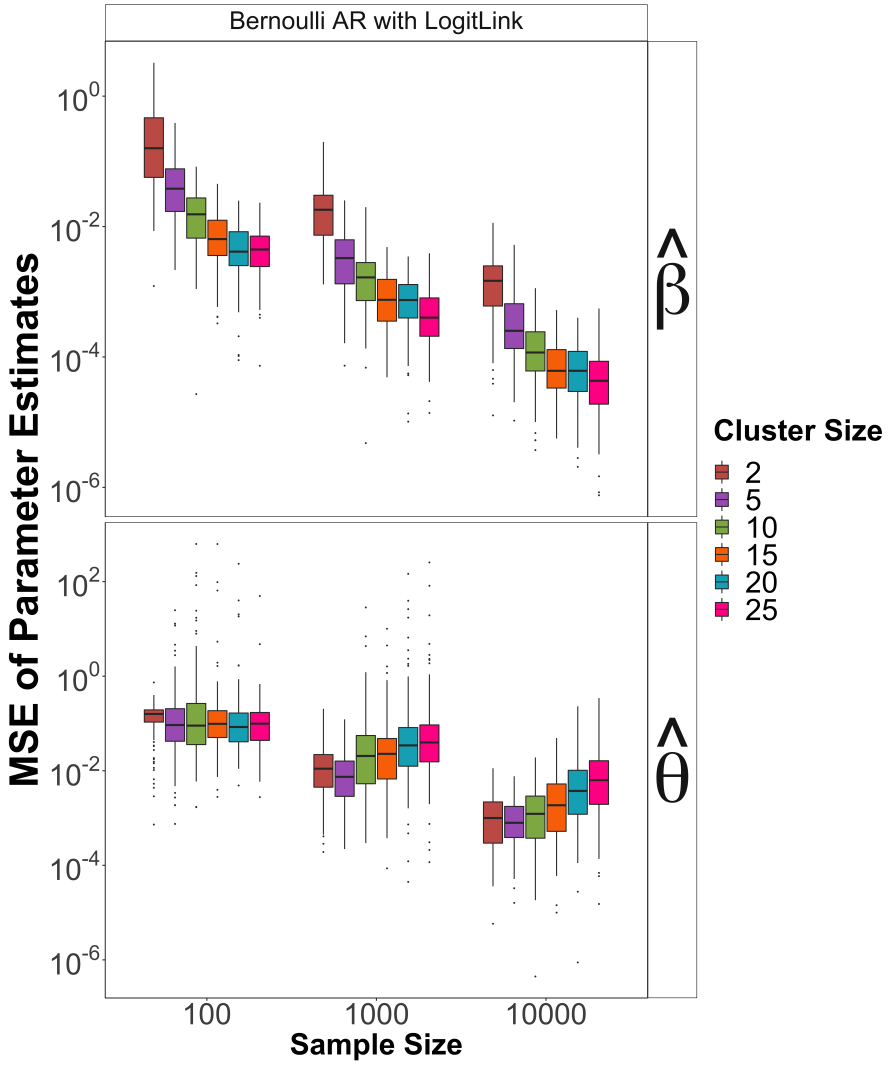

Figure 3.10: Mean squared errors (MSE) of parameter estimates $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ under the AR(1) covariance for the Bernoulli base distribution with logit link function. Each scenario reports involves 100 replicates.

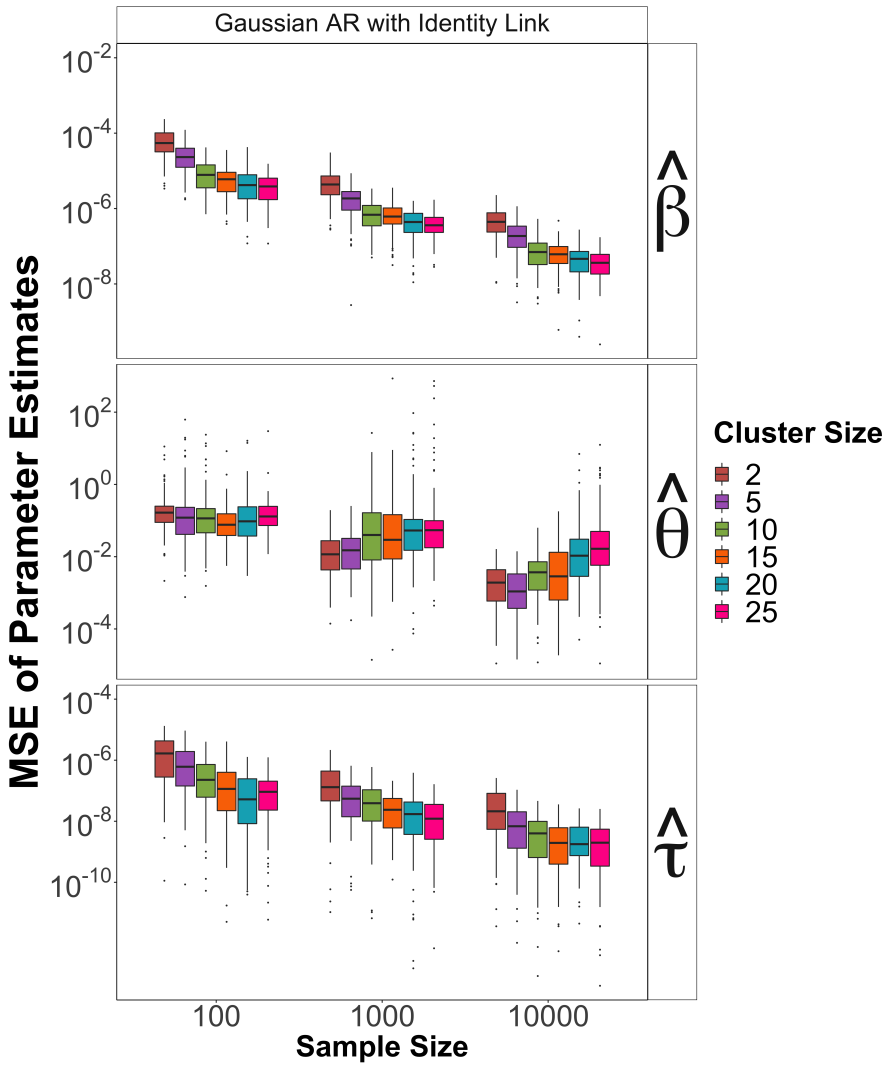

Figure 3.11: Mean squared errors (MSE) of parameter estimates β and θ under the AR(1) covariance for the Normal base distribution with Identity link function. Each scenario reports involves 100 replicates.

3.10.8.2 CS Covariance

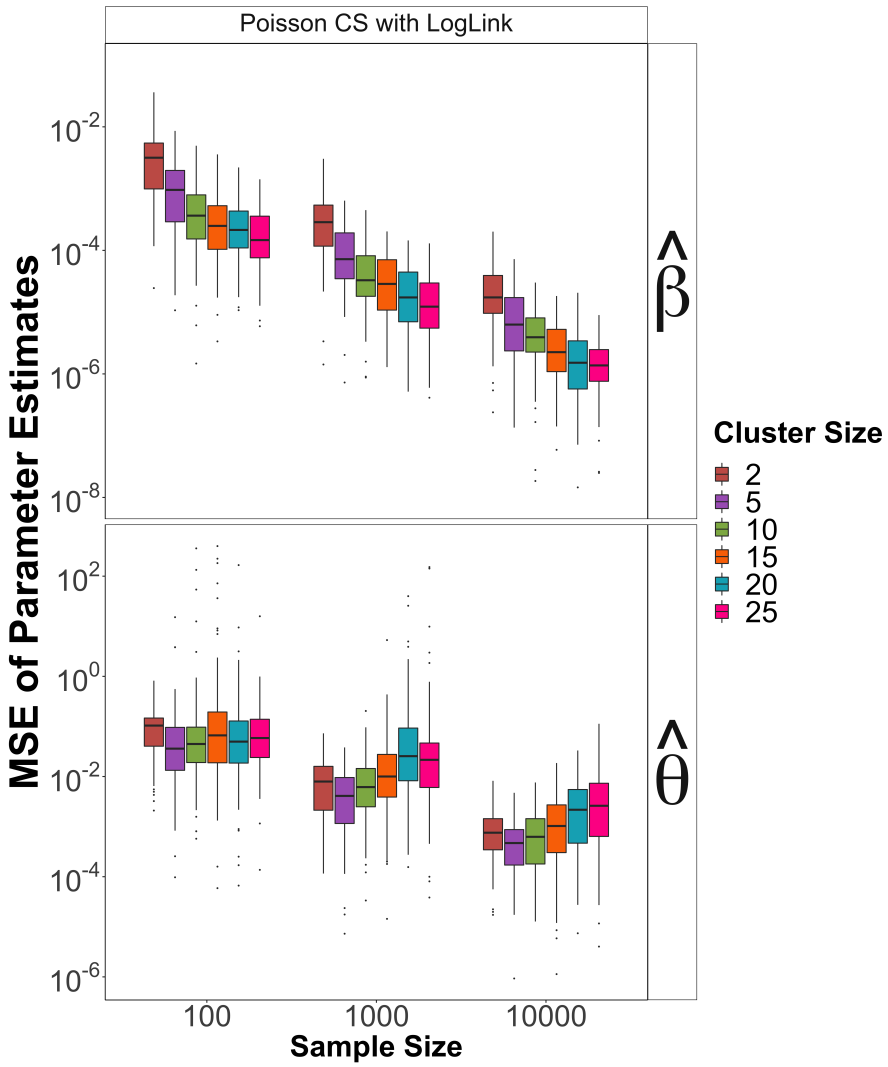

Figure 3.12: Mean squared errors (MSE) of parameter estimates $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ under the CS covariance for the Poisson base distribution with log link function. Each scenario reports involves 100 replicates.

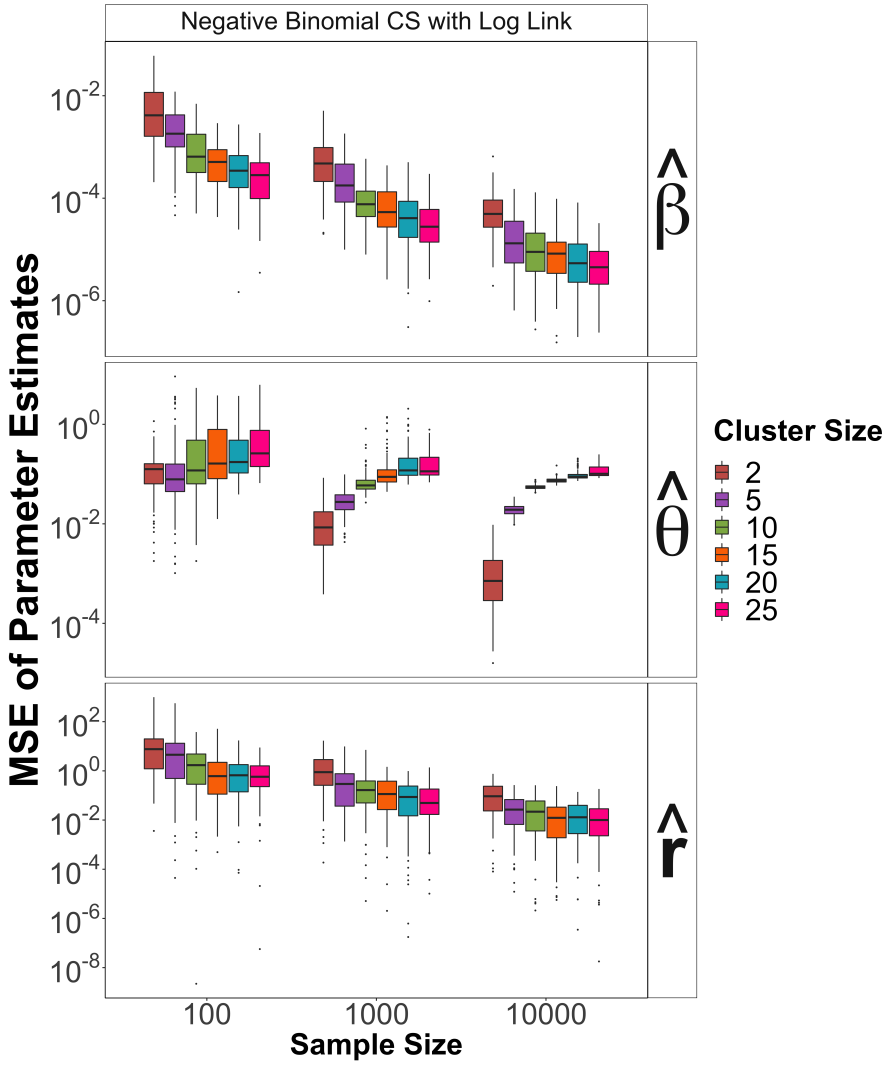

Figure 3.13: Mean squared errors (MSE) of parameter estimates $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ under the CS covariance for the negative binomial base distribution with log link function. Each scenario reports involves 100 replicates.

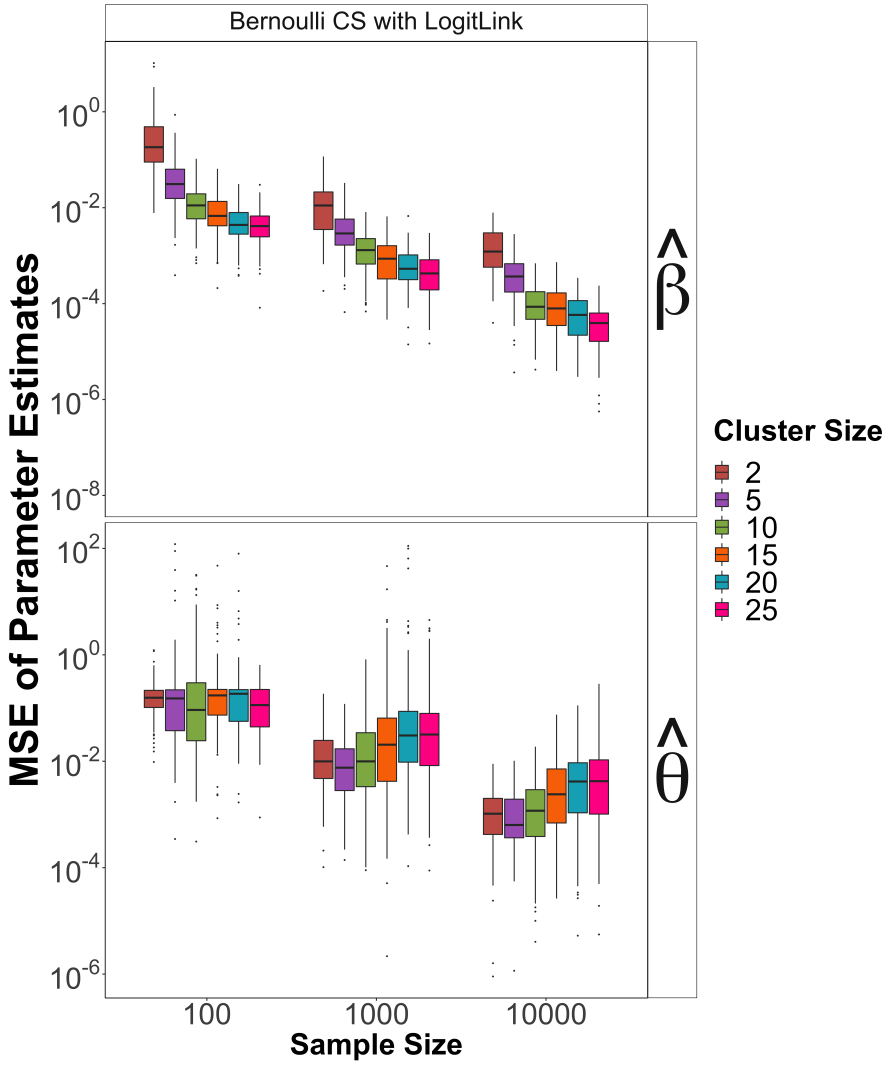

Figure 3.14: Mean squared errors (MSE) of parameter estimates $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ under the CS covariance for the Bernoulli base distribution with logit link function. Each scenario reports involves 100 replicates.

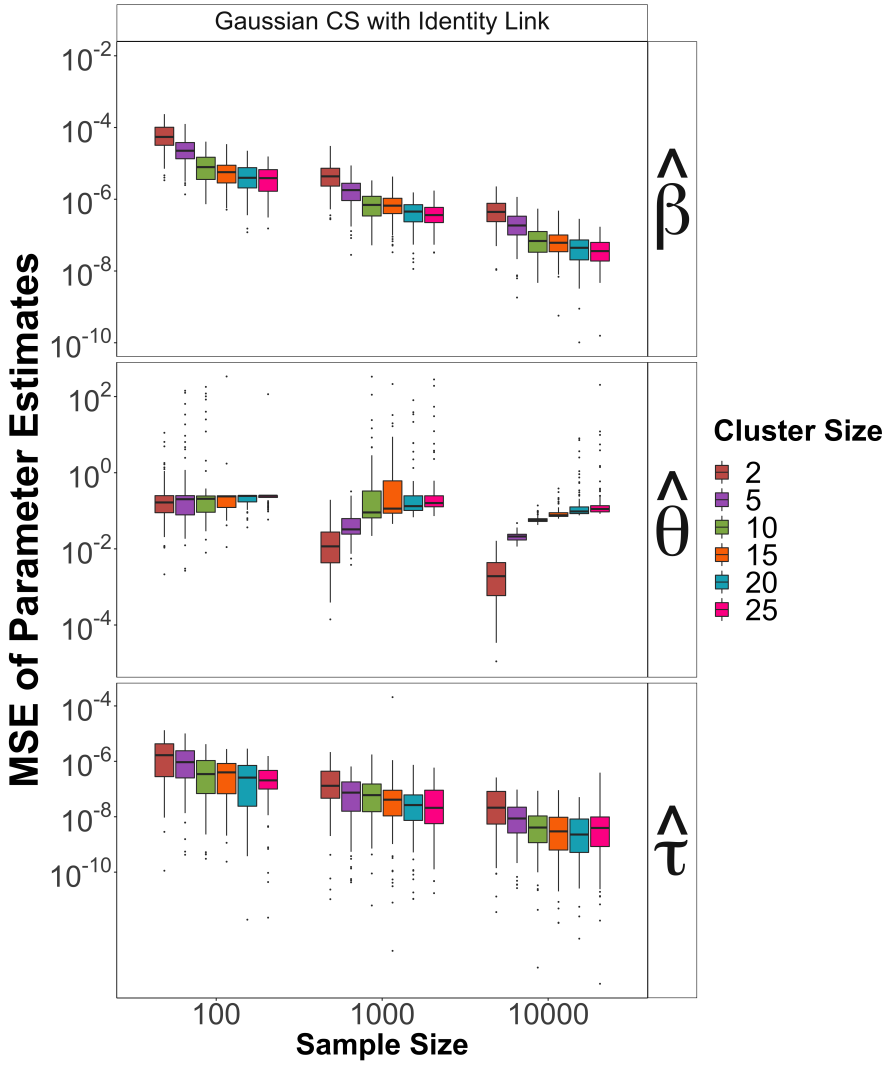

Figure 3.15: Mean squared errors (MSE) of parameter estimates $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ under the CS covariance for the Normal base distribution with Identity link function. Each scenario reports involves 100 replicates.

3.10.8.3 VC Covariance

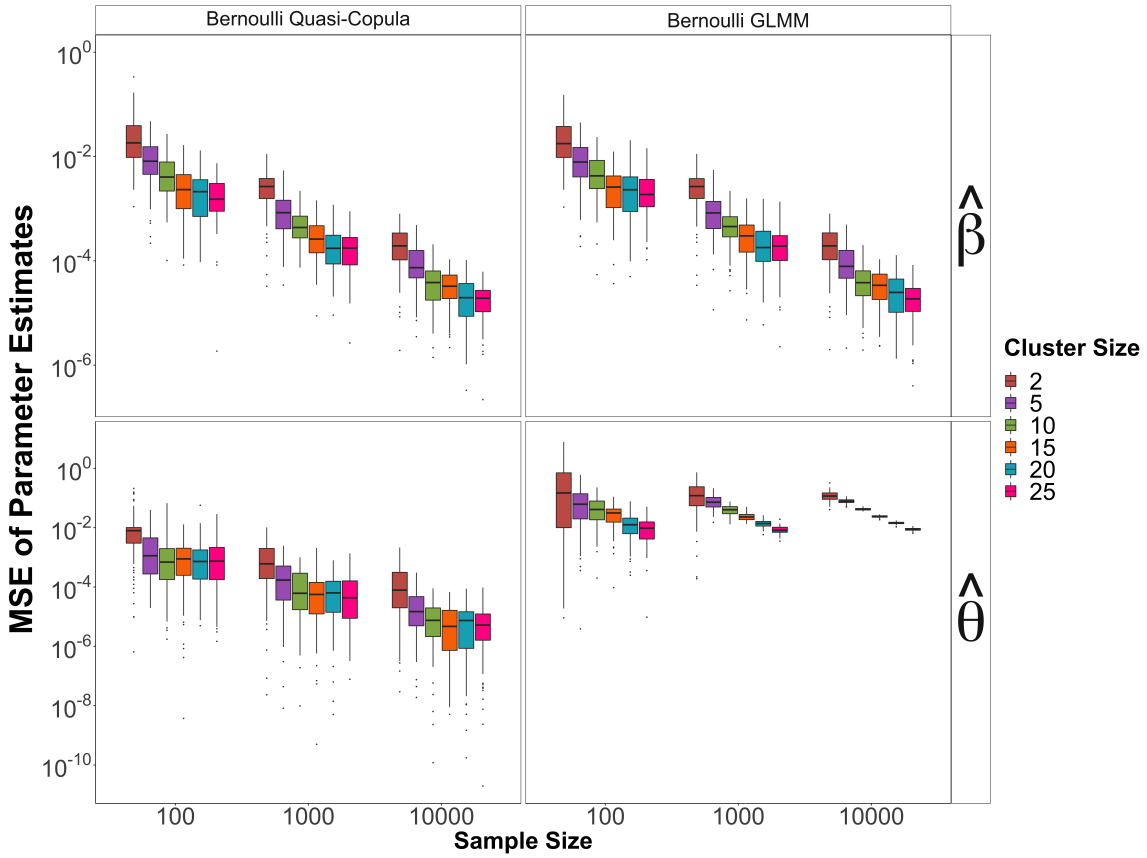

Figure 3.16: Simulation I: Mean squared errors (MSE) of parameter estimates $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ under the Bernoulli base distribution with logit link function and a single VC versus a random intercept GLMM fit via `MixedModels.jl`. Each scenario reports involves 100 replicates.

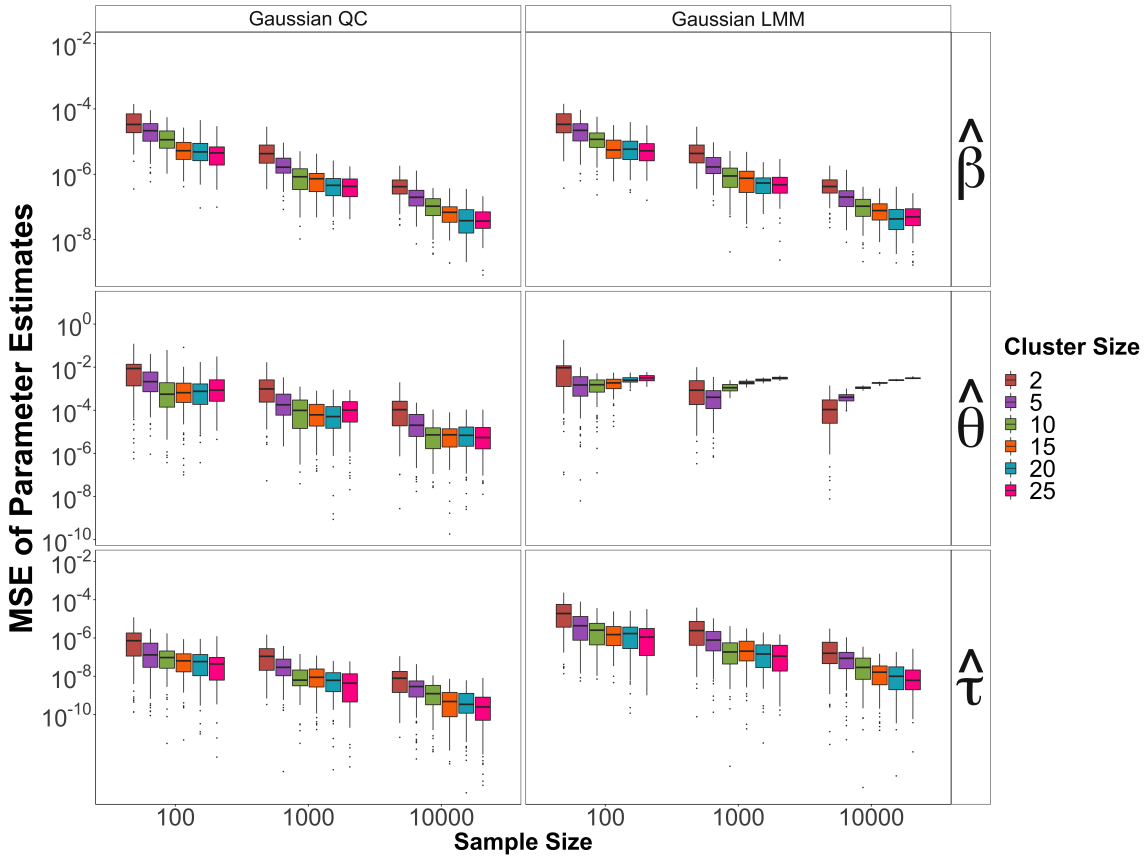

Figure 3.17: Simulation I: Mean squared errors (MSE) of parameter estimates $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ under the Normal base distribution with Identity link function and a single VC versus a random intercept LMM fit via `MixedModels.jl`. Each scenario reports involves 100 replicates.

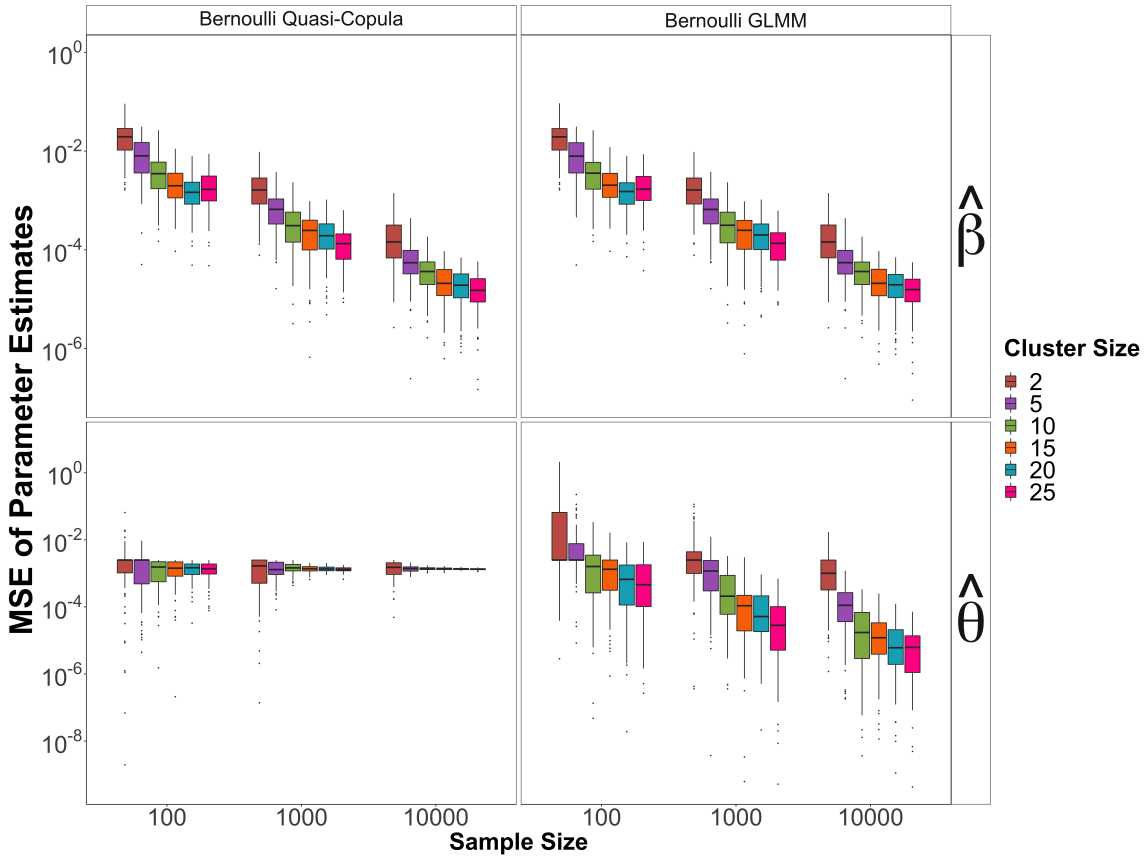

Figure 3.18: Simulation II: Mean squared errors (MSE) of parameter estimates $\boldsymbol{\beta}$ and $\boldsymbol{\theta} = 0.05$ under the Bernoulli base distribution with logit link function and a single VC versus a random intercept GLMM fit via `MixedModels.jl`. Each scenario reports involves 100 replicates.

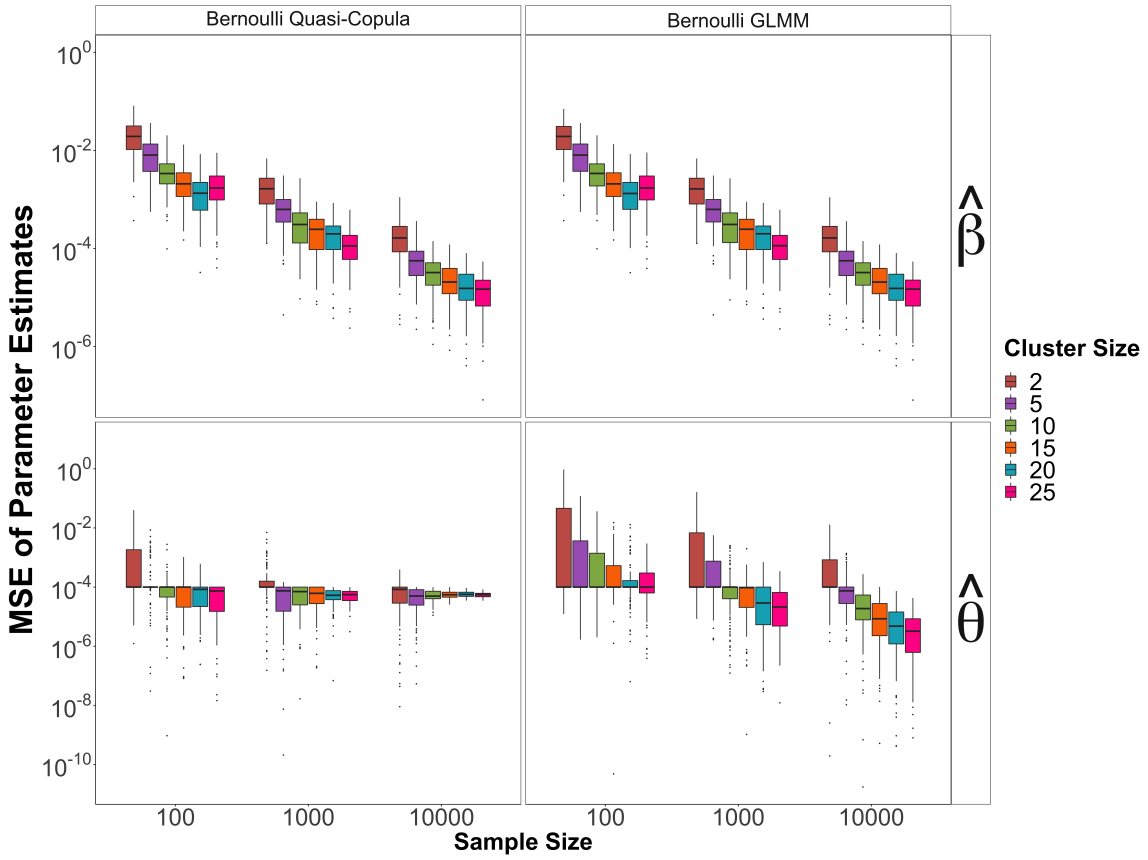

Figure 3.19: Simulation II: Mean squared errors (MSE) of parameter estimates $\boldsymbol{\beta}$ and $\boldsymbol{\theta} = 0.01$ under the Bernoulli base distribution with logit link function and a single VC versus a random intercept GLMM fit via `MixedModels.jl`. Each scenario reports involves 100 replicates.

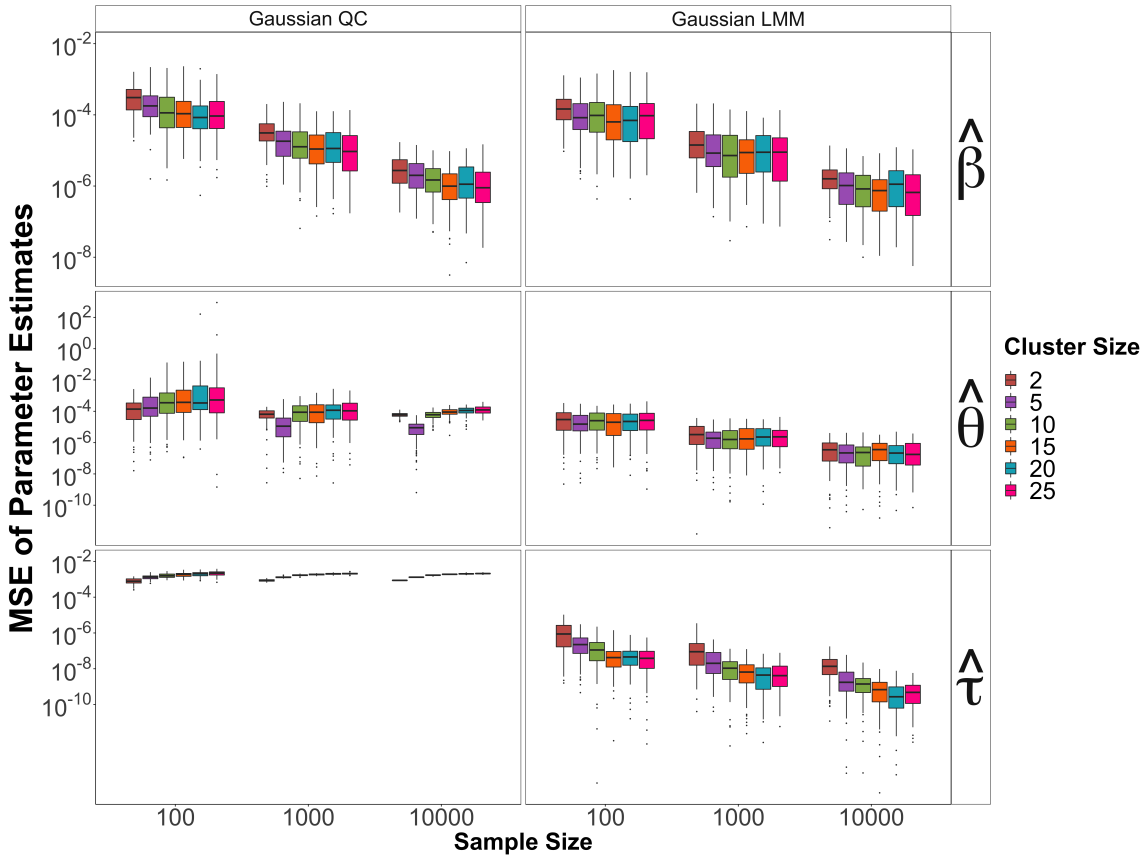

Figure 3.20: Simulation II: Mean squared errors (MSE) of parameter estimates $\boldsymbol{\beta}$ and $\boldsymbol{\theta} = 0.05$ under the Normal base distribution with Identity link function and a single VC versus a random intercept LMM fit via `MixedModels.jl`. Each scenario reports involves 100 replicates.

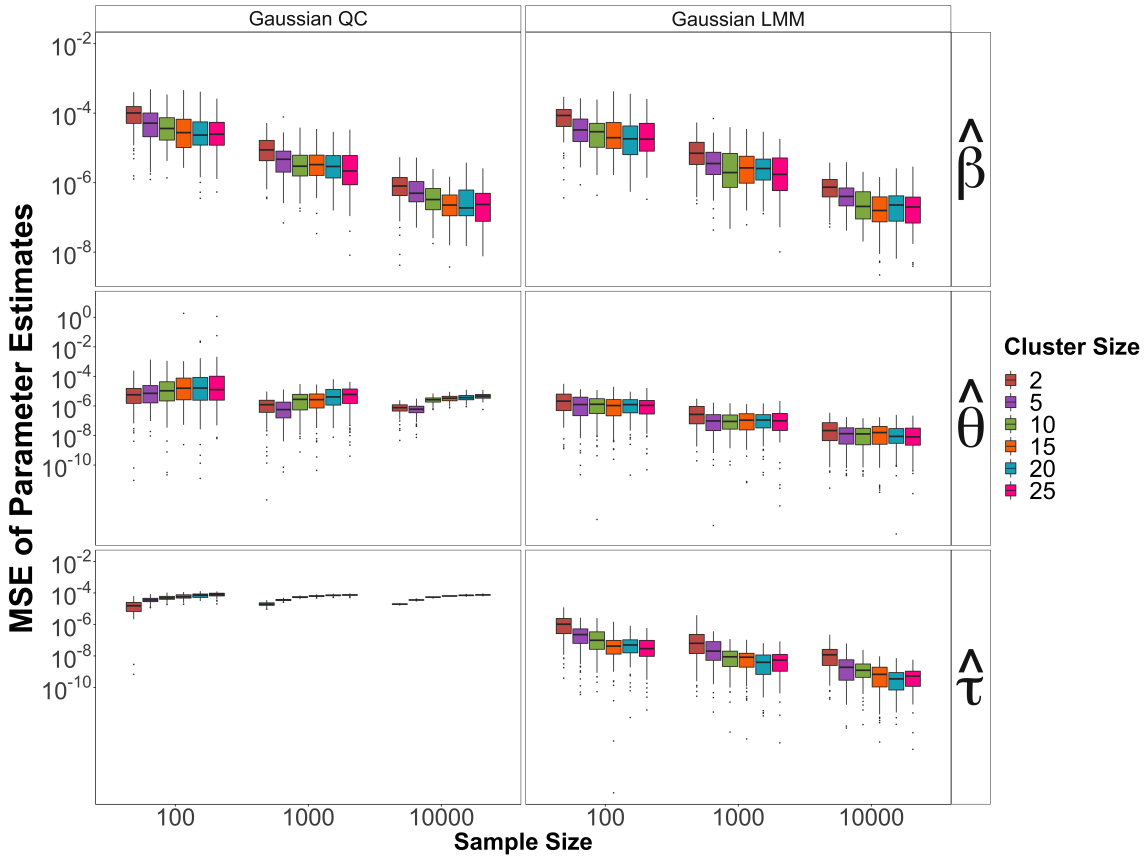

Figure 3.21: Simulation II: Mean squared errors (MSE) of parameter estimates $\boldsymbol{\beta}$ and $\boldsymbol{\theta} = 0.01$ under the Normal base distribution with Identity link function and a single VC versus a random intercept LMM fit via `MixedModels.jl`. Each scenario reports involves 100 replicates.

3.10.8.4 Run Times

Run times under simulation I and II are comparable. Tables 3.5 - 3.8 presents average run times and their standard errors in seconds, for 100 replicates under the AR(1) and CS covariance structures. Tables 3.9 - 3.10 present average run times and their standard errors in seconds, for 100 replicates under simulation II with $\theta_{\text{true}} = 0.01$. All computer runs were performed on a standard 2.3 GHz Intel i9 CPU with 8 cores. Runtimes for the quasi-copula model are presented using multi-threading across 8 cores.

n	d_i	Poisson AR(1) time	Poisson CS time
100	2	0.057 (0.001)	0.065 (0.002)
100	5	0.069 (0.002)	0.079 (0.002)
100	10	0.112 (0.008)	0.133 (0.011)
100	15	0.214 (0.021)	0.212 (0.019)
100	20	0.238 (0.023)	0.234 (0.020)
100	25	0.307 (0.025)	0.289 (0.023)
1000	2	0.060 (0.001)	0.066 (0.001)
1000	5	0.074 (0.001)	0.081 (0.001)
1000	10	0.096 (0.001)	0.108 (0.002)
1000	15	0.112 (0.002)	0.125 (0.002)
1000	20	0.153 (0.012)	0.158 (0.008)
1000	25	0.153 (0.003)	0.180 (0.011)
10000	2	0.201 (0.002)	0.199 (0.002)
10000	5	0.271 (0.002)	0.302 (0.003)
10000	10	0.358 (0.002)	0.446 (0.004)
10000	15	0.447 (0.004)	0.564 (0.006)
10000	20	0.543 (0.005)	0.651 (0.006)
10000	25	0.703 (0.008)	0.757 (0.007)

Table 3.5: Run times and (standard error of run times) in seconds based on 100 replicates for Poisson Base under AR(1) and CS covariance structure with sampling unit size d_i and sample size n.

n	d_i	NB AR(1) time	NB CS time
100	2	0.323 (0.009)	0.300 (0.009)
100	5	0.339 (0.007)	0.311 (0.008)
100	10	0.320 (0.008)	0.337 (0.012)
100	15	0.334 (0.011)	0.391 (0.016)
100	20	0.364 (0.013)	0.372 (0.015)
100	25	0.376 (0.016)	0.362 (0.016)
1000	2	0.445 (0.004)	0.381 (0.004)
1000	5	0.499 (0.003)	0.429 (0.004)
1000	10	0.564 (0.004)	0.520 (0.009)
1000	15	0.654 (0.010)	0.700 (0.021)
1000	20	0.798 (0.019)	0.864 (0.030)
1000	25	0.938 (0.022)	0.864 (0.030)
10000	2	2.656 (0.012)	2.297 (0.017)
10000	5	3.161 (0.013)	2.706 (0.012)
10000	10	3.875 (0.015)	4.001 (0.059)
10000	15	4.924 (0.016)	5.302 (0.140)
10000	20	6.353 (0.028)	6.073 (0.142)
10000	25	7.449 (0.109)	6.987 (0.144)

n	d_i	Bernoulli AR(1) time	Bernoulli CS time
100	2	0.052 (0.002)	0.051 (0.002)
100	5	0.062 (0.002)	0.069 (0.003)
100	10	0.176 (0.019)	0.123 (0.012)
100	15	0.218 (0.021)	0.213 (0.017)
100	20	0.253 (0.022)	0.310 (0.021)
100	25	0.299 (0.024)	0.339 (0.021)
1000	2	0.080 (0.002)	0.056 (0.002)
1000	5	0.081 (0.001)	0.069 (0.002)
1000	10	0.096 (0.006)	0.088 (0.001)
1000	15	0.121 (0.006)	0.119 (0.007)
1000	20	0.179 (0.016)	0.179 (0.015)
1000	25	0.226 (0.020)	0.232 (0.020)
10000	2	0.183 (0.002)	0.171 (0.003)
10000	5	0.256 (0.002)	0.264 (0.003)
10000	10	0.304 (0.002)	0.356 (0.003)
10000	15	0.432 (0.004)	0.450 (0.004)
10000	20	0.507 (0.005)	0.535 (0.007)
10000	25	0.614 (0.005)	0.673 (0.007)

Table 3.7: Run times and (standard error of run times) in seconds based on 100 replicates for Bernoulli Base under AR(1) and CS covariance structure with sampling unit size d_i and sample size n.

n	d_i	Gaussian AR(1) time	Gaussian CS time
100	2	0.213 (0.008)	0.214 (0.008)
100	5	0.305 (0.021)	0.338 (0.022)
100	10	0.392 (0.025)	0.432 (0.027)
100	15	0.507 (0.028)	0.441 (0.029)
100	20	0.533 (0.027)	0.448 (0.031)
100	25	0.590 (0.027)	0.429 (0.030)
1000	2	0.236 (0.006)	0.236 (0.006)
1000	5	0.272 (0.005)	0.309 (0.006)
1000	10	0.365 (0.011)	0.415 (0.010)
1000	15	0.461 (0.024)	0.547 (0.021)
1000	20	0.548 (0.028)	0.628 (0.026)
1000	25	0.561 (0.030)	0.669 (0.026)
10000	2	0.604 (0.013)	0.582 (0.011)
10000	5	0.753 (0.016)	0.793 (0.017)
10000	10	0.871 (0.015)	1.053 (0.015)
10000	15	1.032 (0.018)	1.300 (0.022)
10000	20	1.233 (0.030)	1.718 (0.025)
10000	25	1.437 (0.033)	2.191 (0.042)

Table 3.8: Run times and (standard error of run times) in seconds based on 100 replicates for Gaussian Base under AR(1) and CS covariance structure with sampling unit size d_i and sample size n.

n	d_i	Bernoulli QC time	Bernoulli GLMM time
100	2	0.048 (<0.001)	0.022 (0.002)
100	5	0.049 (0.001)	0.041 (0.001)
100	10	0.050 (0.001)	0.086 (0.004)
100	15	0.049 (0.001)	0.125 (0.005)
100	20	0.047 (0.001)	0.167 (0.005)
100	25	0.047 (0.001)	0.203 (0.008)
1000	2	0.045 (0.001)	0.166 (0.003)
1000	5	0.045 (0.001)	0.446 (0.013)
1000	10	0.043 (0.001)	0.899 (0.022)
1000	15	0.044 (0.001)	1.435 (0.038)
1000	20	0.054 (0.002)	1.888 (0.041)
1000	25	0.077 (0.002)	2.461 (0.057)
10000	2	0.138 (0.003)	1.726 (0.034)
10000	5	0.160 (0.003)	4.711 (0.099)
10000	10	0.189 (0.003)	10.389 (0.221)
10000	15	0.232 (0.003)	15.958 (0.327)
10000	20	0.276 (0.003)	21.609 (0.313)
10000	25	0.349 (0.003)	28.723 (0.494)

Table 3.9: Run times and (standard error of run times) in seconds based on 100 replicates under simulation II with Bernoulli Base, $\theta_{\text{true}} = 0.01$, sampling unit size d_i and sample size n.

n	d_i	Gaussian QC time	LMM time
100	2	0.112 (0.002)	0.003 (0.003)
100	5	0.106 (0.003)	0.001 (<0.001)
100	10	0.097 (0.002)	0.001 (<0.001)
100	15	0.099 (0.004)	0.001 (<0.001)
100	20	0.105 (0.008)	0.001 (<0.001)
100	25	0.109 (0.008)	0.003 (0.002)
1000	2	0.110 (0.002)	0.004 (0.002)
1000	5	0.103 (0.002)	0.002 (<0.001)
1000	10	0.100 (0.002)	0.006 (0.002)
1000	15	0.095 (0.001)	0.006 (0.001)
1000	20	0.094 (0.002)	0.008 (0.001)
1000	25	0.099 (0.002)	0.011 (0.002)
10000	2	0.200 (0.005)	0.018 (0.003)
10000	5	0.192 (0.004)	0.029 (0.003)
10000	10	0.216 (0.004)	0.050 (0.004)
10000	15	0.219 (0.003)	0.067 (0.003)
10000	20	0.239 (0.003)	0.091 (0.003)
10000	25	0.258 (0.002)	0.099 (0.003)

Table 3.10: Run times and (standard error of run times) in seconds based on 100 replicates under simulation II with Gaussian Base, $\theta_{\text{true}} = 0.01$, sampling unit size d_i and sample size n.

3.11 Availability of source code

Project name: QuasiCopula.jl

Project home page: https://github.com/OpenMendel/QuasiCopula.jl

Supported operating systems: Mac OS, Linux, Windows

Programming language: Julia 1.6

License: MIT

All commands needed to reproduce the following results are available at the `QuasiCopula` site in the `manuscript` sub-folder. The NHANES data is available in the 'causaldata' R package at https://cran.r-project.org/web/packages/causaldata/causaldata.pdf.

CHAPTER 4

Genome-Wide Association Analysis with Quasi-Copula

4.1 Motivation

Genetic epidemiological studies typically collect data on multiple correlated traits. However, most analysis software for genome-wide association studies (GWAS) currently restricts researchers to single-trait analysis as the likelihood function for correlated phenotypes can be difficult to specify, particularly for mixed outcomes. When appropriate, there are considerable advantages for analyzing multiple correlated phenotypes jointly rather than separately [10, 3, 25]. The advantages of performing a true joint analysis of correlated traits include the ability to (a) better detect pleiotropy (one locus influencing multiple correlated traits), (b) incorporate extra information on cross-trait covariances, (c) reduce the burden of multiple testing, and (d) ultimately increase statistical power. These advantages have been shown to hold even if the correlation among traits is weak [10].

Motivated by the previous chapter, we are interested in applying the quasi-copula model to genome-wide association studies (GWAS) of correlated phenotypes. First, we present the quasi-copula model in a genetic context. Second, we derive an efficient testing strategy that makes the quasi-copula model computationally feasible for biobank scale GWAS.

4.2 Genetic Model

Consider n independent realizations $\mathbf{y}_i$ from the quasi-copula density. Following the notation from the previous chapter, let $\mathbf{X}_i$ denote the design matrix and $\boldsymbol{\mu}_i$ denote the mean vector for the i^{th}

sampling unit. Under the alternative model, the mean components $\boldsymbol{\mu}_i$ are a function of both genetic and non-genetic covariates. The design matrix can be written as $\mathbf{X}_i = (\mathbf{N}_i, \mathbf{G}_i)$, where $\mathbf{N}_i$ is the design matrix under the null model that contains p non-genetic covariates (i.e. age, sex) and $\mathbf{G}_i$ contains the set of q genotypes to be tested. Using the inverse link function $g(\cdot)$, the mean vector for the i^{th} observation takes the form $\boldsymbol{\mu}_i(\boldsymbol{\beta}, \boldsymbol{\gamma}) = g(\mathbf{N}_i\boldsymbol{\beta})$ under the null model, and $\boldsymbol{\mu}_i(\boldsymbol{\beta}, \boldsymbol{\gamma}) = g(\mathbf{N}_i\boldsymbol{\beta} + \mathbf{G}_i\boldsymbol{\gamma})$ under the full model.

For a single SNP, $\mathbf{G}_i$ is the scalar genotype value and for a SNP-set, $\mathbf{G}_i$ is the genotype vector of q SNPs. We will test $\mathbf{H_0}: \boldsymbol{\gamma} = \mathbf{0}$ to assess the significance of the SNP or SNP-set. In either case, practical association testing strategies scalable to biobank data must be considered. Using the likelihood ratio test (LRT) becomes computationally challenging in biobank scale data, as the full model needs to be re-fitted at every single SNP or SNP-set. In contrast, the score test only requires fitting the null model once and updating a score test statistic. A sensible strategy when running a GWAS on biobank scale data is to first use the score test to screen all the SNPs, and then re-analyze only the most significant SNPs using the more powerful, but slower, LRT.

4.3 Score Test for Individual SNPs or SNP-sets

4.3.1 Score and Approximate Observed Information

In implementing the score test, we assume that the observed information matrix is approximately block diagonal in the mean and variance parameters. We further approximate the observed information matrix by the sum of cross products of the scores from the individual sampling units. The

pertinent quantities are then

$$\begin{aligned}
\nabla\mathcal{L}(\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\theta}) &= \sum_{i=1}^{n}\nabla\mathcal{L}_i(\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\theta}) \\
&= \sum_{i=1}^{n}\mathbf{X_i}^t\mathbf{W_i}(\boldsymbol{\beta},\boldsymbol{\gamma})[\mathbf{Y_i}-\mu_i(\boldsymbol{\beta},\boldsymbol{\gamma})]+\sum_{i=1}^{n}\frac{d\mathbf{r}_i(\boldsymbol{\beta},\boldsymbol{\gamma})^t\boldsymbol{\Gamma}_i(\boldsymbol{\theta})\mathbf{r}_i(\boldsymbol{\beta},\boldsymbol{\gamma})}{1+\frac{1}{2}\mathbf{r}_i(\boldsymbol{\beta},\boldsymbol{\gamma})^t\boldsymbol{\Gamma}_i(\boldsymbol{\theta})\mathbf{r}_i(\boldsymbol{\beta},\boldsymbol{\gamma})} \\
&= \sum_{i=1}^{n}\begin{bmatrix}\mathbf{N}_i^t\\ \mathbf{G}_i^t\end{bmatrix}\mathbf{W_i}(\boldsymbol{\beta},\boldsymbol{\gamma})[\mathbf{Y_i}-\mu_i(\boldsymbol{\beta},\boldsymbol{\gamma})]+\sum_{i=1}^{n}\frac{d\mathbf{r}_i(\boldsymbol{\beta},\boldsymbol{\gamma})^t\boldsymbol{\Gamma}_i(\boldsymbol{\theta})\mathbf{r}_i(\boldsymbol{\beta},\boldsymbol{\gamma})}{1+\frac{1}{2}\mathbf{r}_i(\boldsymbol{\beta},\boldsymbol{\gamma})^t\boldsymbol{\Gamma}_i(\boldsymbol{\theta})\mathbf{r}_i(\boldsymbol{\beta},\boldsymbol{\gamma})} \\
&= \begin{bmatrix}\sum_{i=1}^{n}\mathbf{N}_i^t\mathbf{W_i}(\boldsymbol{\beta},\boldsymbol{\gamma})[\mathbf{Y_i}-\mu_i(\boldsymbol{\beta},\boldsymbol{\gamma})]\\ \\ \sum_{i=1}^{n}\mathbf{G}_i^t\mathbf{W_i}(\boldsymbol{\beta},\boldsymbol{\gamma})[\mathbf{Y_i}-\mu_i(\boldsymbol{\beta},\boldsymbol{\gamma})]\end{bmatrix}+\begin{bmatrix}\sum_{i=1}^{n}\frac{d_{\boldsymbol{\beta}}\mathbf{r}_i(\boldsymbol{\beta},\boldsymbol{\gamma})^t\boldsymbol{\Gamma}_i(\boldsymbol{\theta})\mathbf{r}_i(\boldsymbol{\beta},\boldsymbol{\gamma})}{1+\frac{1}{2}\mathbf{r}_i(\boldsymbol{\beta},\boldsymbol{\gamma})^t\boldsymbol{\Gamma}_i(\boldsymbol{\theta})\mathbf{r}_i(\boldsymbol{\beta},\boldsymbol{\gamma})}\\ \\ \sum_{i=1}^{n}\frac{d_{\boldsymbol{\gamma}}\mathbf{r}_i(\boldsymbol{\beta},\boldsymbol{\gamma})^t\boldsymbol{\Gamma}_i(\boldsymbol{\theta})\mathbf{r}_i(\boldsymbol{\beta},\boldsymbol{\gamma})}{1+\frac{1}{2}\mathbf{r}_i(\boldsymbol{\beta},\boldsymbol{\gamma})^t\boldsymbol{\Gamma}_i(\boldsymbol{\theta})\mathbf{r}_i(\boldsymbol{\beta},\boldsymbol{\gamma})}\end{bmatrix}
\end{aligned}$$

and

$$-d^2\mathcal{L}(\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\theta}) \approx \sum_{i=1}^{n}\nabla\mathcal{L}_i(\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\theta})d\mathcal{L}_i(\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\theta}),$$

where ∇ and d produce a gradient and differential, respectively, with respect to the mean parameters $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$, $\mathbf{W_i}(\boldsymbol{\beta},\boldsymbol{\gamma})$ is a diagonal matrix of "working weights"

$$\mathbf{W_i}(\boldsymbol{\beta},\boldsymbol{\gamma}) = \begin{bmatrix}\frac{g'(\eta_{i1})}{\sigma_{i1}^2} & 0 & \cdots & 0\\ 0 & \frac{g'(\eta_{i2})}{\sigma_{i2}^2} & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & \frac{g'(\eta_{id_i})}{\sigma_{id_i}^2}\end{bmatrix},$$

and η_{ij} is the jth entry of $\mathbf{N}_i\boldsymbol{\beta}+\mathbf{G}_i\boldsymbol{\gamma}$.

4.3.2 Score Test Statistic

To test the SNP-set hypothesis

$$\mathbf{H_0}: \gamma_1 = \cdots = \gamma_q = 0,$$

we partition the approximate observed information under the null model into the block matrix

$$\begin{array}{c} \\ \beta_1 \\ \vdots \\ \beta_p \\ \\ \gamma_1 \\ \vdots \\ \gamma_q \end{array} \begin{array}{c} \begin{array}{cccc} \beta_1 & \cdots & \beta_p & \end{array} \\ \left(\begin{array}{c|c} \multicolumn{2}{c}{\mathbf{P}} & \multicolumn{2}{c}{\mathbf{R}} \\ & & & \\ \hline & & & \\ \multicolumn{2}{c}{\mathbf{R^T}} & \multicolumn{2}{c}{\mathbf{Q}} \\ & & & \end{array}\right) \end{array} \begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \end{array} \begin{array}{cccc} \gamma_1 & \cdots & \gamma_q & \\ \\ \\ \\ \\ \\ \\ \end{array}$$

and denote the score vector (gradient) with respect to γ at the null by $\nabla_\gamma \mathcal{L}$. The score test statistic

$$\mathbf{S} = \nabla_\gamma \mathcal{L}^t (\mathbf{Q} - \mathbf{R^T P^{-1} R})^{-1} \nabla_\gamma \mathcal{L} \sim \chi^2_q$$

is calculated from estimated parameters under the null model (keeping $\gamma = 0$). The various quantities defining the mean vector, including residuals, standardized residual vectors, and working weight matrices $\mathbf{W_i}(\beta, \gamma)$, do not change when we expand the null model to the alternative model. As indicated, the test statistic $\mathbf{S}$ follows a χ^2_q distribution asymptotically.

CHAPTER 5

Conclusions and Future Research

This dissertation presented three distinct projects with different applications in human genetics. Each of these projects aims to develop methods and tools that scale to large sample sizes. The first project provides the genetic community with flexible simulation tools that easily simulate phenotypic traits conditional on PLINK formatted genotype data. We illustrate through two case studies, how `TraitSimulation.jl` cooperates with specific genetic analysis options in `OpenMendel` for efficient statistical power analyses. The second project presents a new class of multivariate models, the quasi-copula model, in which the loglikelihoods contain no integrals, determinants, or matrix inverses. In addition to relaxing independence assumptions, the quasi-copula model offers another avenue for analysis of correlated data and a simple way to capture over-dispersion. In our opinion, its speed and versatility make up for its defects. The final project presents the quasi-copula model in a genetic context, and outlines an efficient testing procedure that can scale to biobank size data.

In the remaining chapter, we discuss potential extensions of the quasi-copula model. Specifically, we will we present two different applications of the quasi-copula model for GWAS under unstructured and structured covariance matrices. For testing individual SNP's or SNP-sets, we can use the efficient testing procedure from the previous chapter regardless of the covariance structure. We first present the application of the quasi-copula model to random sample population data and derive the steepest ascent algorithm for parameter estimation under an unstructured covariance matrix. We then discuss a potential application of the quasi-copula model for pedigree-based GWAS, accounting for kinships. For the pedigree GWAS application of the model, we also present the bivariate trait model and derive another MM-Algorithm that hinges on convexity arguments.

5.1 Quasi-Copula model for Random Sample GWAS

For the application of the quasi-copula model on random sample data, consider the case where each independent realization from the model $\mathbf{y}_i$ represents the d_i phenotypes from the ith subject in our sample of n total independent subjects. The unstructured covariance matrix gives considerable flexibility in capturing dependencies between phenotypes measured from the same subject. In the following section, we derive a steepest ascent algorithm for parameter estimation under an unstructured covariance matrix.

5.1.1 Steepest Ascent Estimation of an Unstructured Covariance

When $\mathbf{\Gamma}_i = \mathbf{\Gamma}$ is unstructured for every sampling unit i, one can also implement steepest ascent. We can parameterize each $d_i \times d_i$ positive semi-definite covariance matrix $\mathbf{\Gamma}$ by its Cholesky decomposition $\mathbf{C}$. The directional derivative of the loglikelihood of the data along the direction specified by the lower triangular matrix $\mathbf{V}$ is

$$d_{\mathbf{V}}\mathcal{L}(\mathbf{C}) = -\frac{n\,\mathrm{tr}(\mathbf{C}\mathbf{V}^t)}{1+\frac{1}{2}\mathrm{tr}(\mathbf{C}\mathbf{C}^t)} + \sum_{i=1}^{n} \frac{\mathrm{tr}[\mathbf{r}_i(\boldsymbol{\beta})\mathbf{r}_i(\boldsymbol{\beta})^t\mathbf{C}\mathbf{V}^t]}{1+\frac{1}{2}\mathbf{r}_i(\boldsymbol{\beta})^t\mathbf{\Gamma}\mathbf{r}_i(\boldsymbol{\beta})}.$$

Hence, the corresponding gradient $\mathbf{G} = \nabla\mathcal{L}(\mathbf{C})$ is the lower triangle of the matrix

$$\mathbf{G} = -\frac{n\mathbf{C}}{1+\frac{1}{2}\mathrm{tr}(\mathbf{C}\mathbf{C}^t)} + \sum_{i=1}^{n} \frac{\mathbf{r}_i(\boldsymbol{\beta})\mathbf{r}_i(\boldsymbol{\beta})^t\mathbf{C}}{1+\frac{1}{2}\mathbf{r}_i(\boldsymbol{\beta})^t\mathbf{C}\mathbf{C}^t\mathbf{r}_i(\boldsymbol{\beta})}.$$

The quadratic form generated by the second directional derivative is

$$\begin{aligned} d^2_{\mathbf{V}}\mathcal{L}(\mathbf{C}) &= \frac{-n\,\mathrm{tr}(\mathbf{V}\mathbf{V}^t)}{1+\frac{1}{2}\mathrm{tr}(\mathbf{C}\mathbf{C}^t)} + \frac{n\,\mathrm{tr}(\mathbf{C}\mathbf{V}^t)^2}{[1+\frac{1}{2}\mathrm{tr}(\mathbf{C}\mathbf{C}^t)]^2} \\ &+ \sum_{i=1}^{n} \frac{\mathbf{r}_i(\boldsymbol{\beta})^t\mathbf{V}\mathbf{V}^t\mathbf{r}_i(\boldsymbol{\beta})}{1+\frac{1}{2}\mathbf{r}_i(\boldsymbol{\beta})^t\mathbf{C}\mathbf{C}^t\mathbf{r}_i(\boldsymbol{\beta})} - \sum_{i=1}^{n} \frac{[\mathbf{r}_i(\boldsymbol{\beta})^t\mathbf{C}\mathbf{V}^t\mathbf{r}_i(\boldsymbol{\beta})]^2}{[1+\frac{1}{2}\mathbf{r}_i(\boldsymbol{\beta})^t\mathbf{C}\mathbf{C}^t\mathbf{r}_i(\boldsymbol{\beta})]^2} \end{aligned}$$

At each iteration, r, the gradient direction $\mathbf{V}_r = \mathbf{G}_r$ maximizes $\mathrm{tr}(\mathbf{V}_r\mathbf{G}_r)$ among all matrices with

the same Frobenius norm. The direction of steepest ascent consequently updates $\mathbf{C}_r$ at iteration r by $\mathbf{C}_{r+1} = \mathbf{C}_r + s_r \mathbf{G}_r$ for some well chosen scalar $s_r > 0$. The choice of s_r is dictated by the competing desires of improving the loglikelihood and keeping the update positive semidefinite. To reduce the error at each iteration, the optimal step size s_r can be selected by minimizing the second-order Taylor expansion

$$\mathcal{L}(\mathbf{C}_{r+1}) \approx \mathcal{L}(\mathbf{C}_r) + s_r \operatorname{tr}(\mathbf{G}_r^t \mathbf{G}_r) + \frac{s_r^2}{2} \operatorname{tr}(\mathbf{G}_r^t d_{\mathbf{V}}^2 \mathcal{L}(\mathbf{C}) \mathbf{G}_r)$$

with respect to s_r. The maximum occurs at the step size

$$s_r = -\frac{\operatorname{tr}(\mathbf{G}_r^\top \mathbf{G}_r)}{\operatorname{tr}[\mathbf{G}_r^\top d_{\mathbf{V}}^2 \mathcal{L}(\mathbf{C}) \mathbf{G}_r]}.$$

5.2 GWAS on Pedigree and Structured Population Data

Family designs are better equipped to detect rare variants and control for population stratification. Even in population-based association studies, not taking into account estimated identity-by-descent (IBD) information can lead to false positives and decreases in statistical power [31]. Unfortunately, pedigree likelihoods are notoriously hard to compute.

Existing methods for GWAS on pedigree or structured populations build on the linear mixed model framework and are often limited to continuous phenotypes. Once one ventures beyond the confines of multivariate Gaussian distributions, analysis choices are limited. To our knowledge, there are no scalable likelihood-based methods for multivariate-trait GWAS when the phenotype is discrete or of mixed types. In contrast, the quasi-copula model is relatively easy to fit and friendly to likelihood ratio hypothesis testing. Additionally, it easily extends to accommodate mixtures of different base distributions and a variety of covariance structures.

5.2.1 Quasi-Copula model for Pedigree GWAS

Following the notation from the previous chapter, for each independent sampling unit i, we can use the quasi-copula density to jointly model a trait vector $\mathbf{y}_i = \begin{bmatrix} y_{i,1} \\ \dots \\ y_{i,d_i} \end{bmatrix}$ with covariance $\mathbf{\Gamma}_i$. Under the blanket assumptions of additivity and independence, let observations $\mathbf{y}_i$ have covariance matrix structured under the variance component model (VCM) framework

$$\mathbf{\Gamma}_i \;\; = \;\; 2\sigma_g^2 \Phi_i + \sigma_e^2 \mathbf{I}_{d_i}.$$

Here σ_g^2 is the polygenic additive variance, σ_e^2 is the random environmental variance, and Φ_i is an estimate of the kinship coefficients in the ith pedigree. When family structures are known, these kinship coefficients are easily calculated [18]. When pedigrees are unknown or suspect, kinship coefficients can be estimated empirically from dense markers. One popular estimate is the genetic relationship matrix (GRM). Within the `OpenMendel` platform, `MendelKinship.jl` [35] provides a number of options to estimate the kinship matrix Φ_i.

5.2.2 Pedigree GWAS Example: Bivariate Trait

Suppose we want to consider matrix outcomes realized from the quasi-copula model. For instance, a bivariate trait from n pedigrees under the quasi-copula model gives rises to the matrix response $\mathbf{Y}_i = \begin{bmatrix} \mathbf{y}_{1i} & \mathbf{y}_{2i} \end{bmatrix}$, where $\mathbf{y}_{1i} = \begin{bmatrix} y_{1,1} \\ \dots \\ y_{1,d_i} \end{bmatrix}$ and $\mathbf{y}_{2i} = \begin{bmatrix} y_{2,1} \\ \dots \\ y_{2,d_i} \end{bmatrix}$ are the corresponding trait vectors for the d_i members of pedigree i. It is convenient stack the columns of the matrix outcome $\mathbf{Y}_i$, to form the response vector $\mathbf{y}_i = \begin{bmatrix} \mathbf{y}_{1i} \\ \mathbf{y}_{2i} \end{bmatrix}$ for pedigree i. Under the assumptions of additivity and independence,

$\mathbf{y}_i$ has the structured covariance matrix

$$\mathbf{\Gamma}_i \quad = \quad \mathbf{\Sigma}_g \otimes 2\Phi_i + \mathbf{\Sigma}_e \otimes \mathbf{I}_{d_i}$$

in the VCM framework. Here $\mathbf{\Sigma}_g$ is the additive genetic covariance matrix, $\mathbf{\Sigma}_e$ is the environmental covariance matrix, Φ_i is the estimate of the kinship matrix, and $\mathbf{I_{d_i}}$ is the $d_i \times d_i$ identity matrix. Kronecker products $\otimes$ are required as explained in [17]. The estimation procedure for mean effects $\boldsymbol{\beta}$ stays relatively unchanged. However, to update variance component matrices $\mathbf{\Sigma}_g$ and $\mathbf{\Sigma}_e$, we exploit another MM algorithm.

5.2.3 An MM-Algorithm for Variance Component Matrices

For m sources of variation, we can decompose the variance of $\mathbf{y}_i$ as

$$\mathbf{\Gamma}_i = \sum_{k=1}^{m} \mathbf{\Sigma}_k \otimes \mathbf{V}_{ik},$$

where $\mathbf{V}_{ik}$ represents a known covariance matrix such as Φ_i and $\mathbf{I}_{d_i}$, and $\mathbf{\Sigma}_k$ represents an unknown covariance matrix to be estimated. In the bivariate example presented above, each $\mathbf{\Sigma}_k$ is of dimension 2×2. The following derivation holds only for a single independent realization from the quasi-copula density.

To derive the MM-update, we will use two identities involving Kronecker products. The first is $\text{tr}(\mathbf{A} \otimes \mathbf{B}) = \text{tr}(\mathbf{A})\,\text{tr}(\mathbf{B})$ [1]. The second, Roth's Kronecker Lemma, is $\mathbf{r}^t(\mathbf{A} \otimes \mathbf{B})\mathbf{r} = \text{tr}(\mathbf{A}\mathbf{R}^t\mathbf{B}\mathbf{R})$, where $\mathbf{r} = \text{vec}(\mathbf{R})$. In our case $\mathbf{r}$ is the stacked vector of standardized residuals.

For a single observation, the relevant parts of the loglikelihood are

$$\mathcal{L}(\mathbf{\Sigma}_k \mid \boldsymbol{\beta}) = -\ln\left[1 + \frac{1}{2}\text{tr}\left(\sum_{k=1}^{m} \mathbf{\Sigma}_k \otimes \mathbf{V}_k\right)\right] + \ln\left[1 + \frac{1}{2}\mathbf{r}(\boldsymbol{\beta})^t \sum_{k=1}^{m} \mathbf{\Sigma}_k \otimes \mathbf{V}_k\, \mathbf{r}(\boldsymbol{\beta})\right].$$

Since $-\ln x$ is convex, the supporting hyperplane inequality applied to the first term of the log-

likelihood gives the minorization

$$-\ln\Big[1+\frac{1}{2}\operatorname{tr}\Big(\sum_{k=1}^{m}\Sigma_k\otimes\mathbf{V}_k\Big)\Big] \;\geq\; -\sum_{k=1}^{m}z_{rk}\operatorname{tr}(\Sigma_k)+c_r$$

at iteration r, where c_r is an irrelevant constant and $z_{rk}=\frac{\frac{1}{2}\operatorname{tr}(\mathbf{V}_k)}{1+\frac{1}{2}\sum_{k=1}^{m}\operatorname{tr}(\mathbf{V}_k)\operatorname{tr}(\Sigma_{rk})}$. Since $\ln x$ is concave, Jensen's Inequality yields the minorization

$$\begin{aligned}
&\ln\Big[1+\frac{1}{2}\mathbf{r}(\boldsymbol{\beta})^t\sum_{k=1}^{m}(\Sigma_k\otimes V_k)\mathbf{r}(\boldsymbol{\beta})\Big]\\
\geq\;&\sum_{k=1}^{m}w_{rk}\ln\Big[\mathbf{r}(\boldsymbol{\beta})^t(\Sigma_k\otimes V_k)\mathbf{r}(\boldsymbol{\beta})\Big]+b_r\\
=\;&\sum_{k=1}^{m}w_{rk}\ln\operatorname{tr}[\Sigma_k\mathbf{R}(\boldsymbol{\beta})^t\mathbf{V}_k\mathbf{R}(\boldsymbol{\beta})]+b_r,
\end{aligned}$$

where b_r is an irrelevant constant and $w_{rk}=\frac{\frac{1}{2}\operatorname{tr}[\Sigma_{rk}\mathbf{R}(\boldsymbol{\beta})^t\mathbf{V}_k\mathbf{R}(\boldsymbol{\beta})]}{1+\frac{1}{2}\sum_{k=1}^{m}\operatorname{tr}[\Sigma_{rk}\mathbf{R}(\boldsymbol{\beta})^t\mathbf{V}_k\mathbf{R}(\boldsymbol{\beta}))]}$.

5.2.4 Surrogate Function

Thus, to update Σ_k, we maximize

$$l(\Sigma_k|\boldsymbol{\beta}) \;=\; -z_{rk}\operatorname{tr}(\Sigma_k)+w_{rk}\ln[\operatorname{tr}(\Sigma_k\mathbf{R}(\boldsymbol{\beta})^t\mathbf{V}_k\mathbf{R}(\boldsymbol{\beta})].$$

The Fan-Von Neuman inequality [19] states that

$$\operatorname{tr}(\Sigma_k\mathbf{R}(\boldsymbol{\beta})^t\mathbf{V}_k\mathbf{R}(\boldsymbol{\beta})) \;\leq\; \sum_{l}\lambda_{kl}\sigma_{kl}$$

where the λ_{kl} are the ordered eigenvalues of Σ_k and the σ_{kl} are the ordered eigenvalues of the matrix $\mathbf{R}(\boldsymbol{\beta})^t\mathbf{V}_k\mathbf{R}(\boldsymbol{\beta})$. Equality holds when the ordered eigenvectors of Σ_k are the same as those of

$\mathbf{R}(\boldsymbol{\beta})^t \mathbf{V}_k \mathbf{R}(\boldsymbol{\beta})$. Thus, we maximize the surrogate function

$$h(\boldsymbol{\lambda} \mid \Sigma_{rk}) \;=\; -z_{rk} \sum_l \lambda_{kl} + w_{rk} \ln \Big(\sum_l \lambda_{kl} \sigma_{kl} \Big)$$

subject to the constraint $\lambda_{kl} \geq 0$.

To tackle this maximkzation problem we drop the subscripts r and k and write the Karush-Kuhn-Tucker stationary condition for $\lambda_l \geq 0$ as $0 = -z + w \frac{\sigma_l}{\sum_j \lambda_j \sigma_j} + \mu_l$, where $\mu_l \geq 0$ and $\mu_l \lambda_l = 0$. Multiplying the stationary condition by λ_l and summing on l give $0 = -z \sum_l \lambda_l + w$ or $\sum_l \lambda_l = \frac{w}{z}$. The MM update problem reduces to maximizing $-w + w \ln \Big(\sum_l \lambda_l \sigma_l \Big)$ subject to $\sum_l \lambda_l = \frac{w}{z}$. If the σ_l are unique and in decreasing order, then the obvious solution is to take $\lambda_1 = \frac{w}{z}$. This leads to the paradoxical conclusion that Σ_k has rank 1 and suggests that amalgamating all pedigrees into a single pedigree is possibly a bad idea. It might be better to optimize the surrogate by steepest ascent via Cholesky decompositions. When several σ_l are maximal, the solution to the KKT equations is nonunique because one can arbitrarily divide the mass $\frac{w}{z}$ among the top λ_l.

Bibliography

[1] S. Banerjee and A. Roy. *Linear algebra and matrix analysis for statistics*, volume 181. Crc Press Boca Raton, FL, USA:, 2014.

[2] D. Bates, M. Mächler, B. Bolker, and S. Walker. Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, 67(1):1–48, 2015.

[3] L. E. Bauman, L. Almasy, J. Blangero, R. Duggirala, J. S. Sinsheimer, and K. Lange. Fishing for pleiotropic qtls in a polygenic sea. *Annals of human genetics*, 69(5):590–611, 2005.

[4] M. Besançon, D. Anthoff, A. Arslan, S. Byrne, D. Lin, T. Papamarkou, and J. Pearson. Distributions.jl: Definition and modeling of probability distributions in the juliastats ecosystem. *arXiv e-prints*, page arXiv:1907.08611, Jul 2019.

[5] N. E. Breslow and D. G. Clayton. Approximate inference in generalized linear mixed models. *Journal of the American Statistical Association*, 88(421):9–25, 1993.

[6] N. E. Breslow and D. G. Clayton. Approximate inference in generalized linear mixed models. *Journal of the American statistical Association*, 88(421):9–25, 1993.

[7] M. E. Brooks, K. Kristensen, K. J. van Benthem, A. Magnusson, C. W. Berg, A. Nielsen, H. J. Skaug, M. Maechler, and B. M. Bolker. glmmTMB balances speed and flexibility among packages for zero-inflated generalized linear mixed modeling. *The R Journal*, 9(2):378–400, 2017.

[8] H. Chen, C. Wang, M. P. Conomos, A. M. Stilp, Z. Li, T. Sofer, A. A. Szpiro, W. Chen, J. M. Brehm, J. C. Celedon, S. Redline, G. J. Papanicolaou, T. A. Hornton, C. C. Laurie, K. Rice, and X. Lin. Control for population structure and relatedness for binary traits in genetic association studies via logistic mixed models. *American Journal of Human Genetics*, 98:653–666, 2017.

[9] B. B. Cohen, H. E. Barbano, C. S. Cox, J. J. Feldman, F. F. Finucane, J. C. Kleinman, and J. H. Madans. Plan and operation of the NHANES I Epidemiologic Followup Study: 1982-84. *Vital and Health Statistics, Series 1*, 22:1–142, 1987.

[10] T. E. Galesloot, K. Van Steen, L. A. Kiemeney, L. L. Janss, and S. H. Vermeulen. A comparison of multivariate genome-wide association methods. *PloS one*, 9(4):e95923, 2014.

[11] C. A. German, J. S. Sinsheimer, Y. C. Klimentidis, H. Zhou, and J. J. Zhou. Ordered multinomial regression for genetic association analysis of ordinal phenotypes at Biobank scale. *Genetic Epidemiology*, 44:248–260, 2020.

[12] IGSR. International genome sample resource, 2020.

[13] S. S. Ji, C. A. German, K. Lange, J. S. Sinsheimer, H. Zhou, J. Zhou, and E. M. Sobel. Modern simulation utilities for genetic analysis. *BMC bioinformatics*, 22(1):1–13, 2021.

[14] JuliaComputing. Distibuted computing.

[15] JuliaComputing. Multi-threading.

[16] JuliaComputing. Parallel computing.

[17] K. Lange. *Mathematical and Statistical Methods for Genetic Analysis*. Springer, New York, 2nd edition, 2002.

[18] K. Lange. *Mathematical and statistical methods for genetic analysis*, volume 488. Springer, 2002.

[19] K. Lange. *MM Optimization Algorithms*. SIAM, 2016.

[20] K. Lange, D. R. Hunter, and I. Yang. Optimization transfer using surrogate objective functions. *Journal of Computational and Graphical Statistics*, 9(1):1–20, 2000.

[21] K.-Y. Liang and S. L. Zeger. Longitudinal data analysis using generalized linear models. *Biometrika*, 73(1):13–22, 1986.

[22] D. Lin, J. M. White, S. Byrne, D. Bates, A. Noack, J. Pearson, A. Arslan, K. Squire, D. Anthoff, T. Papamarkou, M. Besançon, and et al. JuliaStats/Distributions.jl: a Julia package for probability distributions and associated functions, may 2019.

[23] M. Ohtaki. Globally convergent algorithm without derivatives for maximizing a multivariate function. *Proceedings of Development of Statistical Theories and their Application for Complex Nonlinear Data*, 1999.

[24] J. C. Pinheiro and D. M. Bates. Approximations to the log-likelihood function in the nonlinear mixed-effects model. *Journal of Computational and Graphical Statistics*, 4(1):12–35, 1995.

[25] H. F. Porter and P. F. O'Reilly. Multivariate simulation framework reveals performance of multi-trait gwas methods. *Scientific reports*, 7(1):1–12, 2017.

[26] M. Sklar. Fonctions de repartition an dimensions et leurs marges. *Publ. Inst. Statist. Univ. Paris*, 8:229–231, 1959.

[27] P. X.-K. Song, M. Li, and Y. Yuan. Joint regression analysis of correlated data using gaussian copulas. *Biometrics*, 65(1):60–68, 2009.

[28] T. Tonda. A class of multivariate discrete distributions based on an approximate density in {GLMM}. *Hiroshima Mathematical Journal*, 35(2):327–349, 2005.

[29] S. L. Zeger and M. R. Karim. Generalized linear models with random effects; a gibbs sampling approach. *Journal of the American statistical association*, 86(413):79–86, 1991.

[30] H. Zhou. Snparrays.jl.

[31] H. Zhou, J. Blangero, T. D. Dyer, K.-h. K. Chan, K. Lange, and E. M. Sobel. Fast genome-wide qtl association mapping on pedigree and population data. *Genetic epidemiology*, 41(3):174–186, 2017.

[32] H. Zhou, L. Hu, J. Zhou, and K. Lange. Mm algorithms for variance components models. *Journal of Computational and Graphical Statistics*, 28(2):350–361, 2019.

[33] H. Zhou, J. C. Papp, S. Ko, C. A. German, J. Day, M. A. Suchard, A. Landeros, and A. Noack. SnpArrays.jl: Julia package for compressed storage of SNP data, Feb. 2020.

[34] H. Zhou, J. Sinsheimer, D. Bates, B. Chu, C. German, S. Ji, K. Keys, J. Kim, S. Ko, G. Mosher, J. Papp, E. Sobel, J. Zhai, J. Zhou, and K. Lange. OPENMENDEL: A cooperative programming project for statistical genetics. *Human Genetics*, 139:61–71, 2020.

[35] H. Zhou, J. S. Sinsheimer, D. M. Bates, B. B. Chu, C. A. German, S. S. Ji, K. L. Keys, J. Kim, S. Ko, G. D. Mosher, et al. OpenMendel: a cooperative programming project for statistical genetics. *Human Genetics*, 139(1):61–71, 2020.