Comparison of Generalized Linear Mixed Model Estimation Methods

Jacob Larsen Rich

Department of Mathematical Sciences
Montana State University

May 3, 2018

A writing project submitted in partial fulfillment
of the requirements for the degree

Master of Science in Statistics

APPROVAL

of a writing project submitted by

Jacob Larsen Rich

This writing project has been read by the writing project advisor and has been found to be satisfactory regarding content, English usage, format, citations, bibliographic style, and consistency, and is ready for submission to the Statistics Faculty.

Date	Laura Hildreth Writing Project Advisor
Date	Mark C. Greenwood Writing Projects Coordinator

Contents

Abstract

Generalized linear mixed models (GLMMs) arose from the necessity to analyze non-normal responses that are correlated or clustered in some way, relying on generalized linear and linear mixed model theory. These models have wide ranging applications in ecological and evolutionary biology where observations are often correlated and have non-normal responses. The likelihood functions of these models lack closed form solutions for parameter estimates and as such, a number of likelihood-based approximation methods have been developed for inference in GLMMs along with Bayesian approaches which rely on Markov chain Monte Carlo methods. Previous studies on the performance of these methods have either focused on specific classes of GLMMs, such as Binomial GLMMs, or on highly structured data seen in situations like clinical trials. In ecological and evolutionary biological data, small datasets are commonplace and the properties of these estimation methods in this context are not well explored. Through a simulation study, the accuracy and precision of four methods (Laplace approximation, adaptive Gaussian-Hermite Quadrature, Penalized Quasi-likelihood, and a Bayesian hierarchical model) are assessed on simulated responses generated from a real-world biological dataset at multiple sample sizes. While a large dataset ($n = 599$) shows precise and accurate estimation of parameters, smaller datasets ($n < 250$) show dramatic bias in estimation of variance components for likelihood-based methods. The exhibited bias indicates that caution should be exercised when applying these methods to small datasets and that further work needs to be done to explore the properties of these estimation methods.

1 Introduction

In many ecology and evolutionary biology applications, the response variable is often not continuous or normally distributed and are comprised of discrete counts or presence-absence type data. Typically, if the observations are independent, analyses of these non-normal responses can be carried out with generalized linear models. However, if observations are known to be correlated or clustered, a different class of linear model known as generalized linear mixed models (GLMMs) must be used to incorporate that dependent structure of the data. These GLMMs are relatively new in statistics, being formally introduced in 1989 by McCullagh and Nelder in their book *Generalized Linear Models*. In the years since, many different methods of estimation for these complex models have been developed.

Practitioners have a wide range of options for fitting GLMMs, but not all methods are equal in terms of bias and precision of their estimates. Some methods have known issues in certain situations, but literature on the performance of GLMM estimation methods is either sparse or for specific classes of GLMMs - like binomial GLMMs. As such, this project is intended to explore four of the most common methods used for estimation of GLMMs with a real-world dataset, something not often seen in the study of GLMMs. Through a simulation study, the bias and precision of the four selected methods (Laplace approximation, adaptive Gaussian-Hermite quadrature, penalized quasi-likelihood, and Bayesian hierarchical modeling) will be assessed on the owlet begging dataset from Roulin and Bersier (2007, by way of Zuur et al. 2009) with simulated responses. Starting with a review of linear models, we will briefly look at the background for generalized linear mixed models then explore the practical consequences of the selected estimation methods, finishing with a discussion of the simulation study and some extensions of this project.

1.1 Background

To motivate the variety of estimation methods for GLMMs, we will first look at a very specific case of a GLMM, the general linear model. Suppose we have the following linear model:

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

where $\boldsymbol{y}$ is an $n \times 1$ vector of responses, $\boldsymbol{X}$ is the $n \times p$ design matrix which is assumed to be full column rank and have fixed values, $\boldsymbol{\beta}$ is the $p \times 1$ fixed effects vector and $\boldsymbol{\varepsilon}$ is an $n \times 1$ vector of random errors. In this model, the $\boldsymbol{X}\boldsymbol{\beta}$ portion is the deterministic portion of the model while the $\boldsymbol{\varepsilon}$ portion is the probabilistic part of the model. Further, we assume the following:

1. $\boldsymbol{\varepsilon} \sim N(\boldsymbol{0}, \sigma^2\boldsymbol{I})$.
2. $\boldsymbol{X}\boldsymbol{\beta}$ is linear in the β's.

The parameters in $\boldsymbol{\beta}$ are typically estimated using OLS. Least squares are obtained through minimizing the sum of squared residuals (Faraway 2014). Provided linearity assumptions hold, the OLS estimators will be the Best Linear Unbiased Estimators (BLUE) for the β's (Faraway 2014). The $\boldsymbol{\beta}$ effects are specific to the sample and are not random variables. Importantly, the general least squares estimator, $\hat{\boldsymbol{\beta}}_{OLS} = (\boldsymbol{X}^\intercal\boldsymbol{X})^{-1}\boldsymbol{X}^\intercal\boldsymbol{y}$, is also the maximum likelihood (ML) estimator since it was specified that the errors are assumed to be normally distributed (Faraway 2014).

In many applications, these assumptions are reasonably satisfied and OLS estimation can be used without issue; however, there are also situations where the normality or independence assumptions are violated, which has lead to the development of other models. Currently, the most versatile linear model available is the generalized linear mixed model (GLMM), which uses theory from linear mixed models to allow dependent errors (LMM) and generalized linear models (GLM) to allow for non-normally distributed responses for correlated observations. Building up to GLMMs, we will take a quick look at these two other classes of models and start with linear mixed models.

1.1.1 Mixed Models

The assumption of independence of errors is a strong assumption that is violated in many study designs with repeated measurements, clustered observations, or with spatial or temporal elements. Procedures for mixed effects models were primarily developed by Goldberger, Henderson, and Harville in the 1960's and 1970's, with Henderson's work in animal genetics providing the earliest solutions to mixed effects model estimation

(McLean et al. 1991, Robinson 1991). In the decades since, mixed effects models have become common in usage, ranging from their origin in agriculture to genetics to sales and marketing.

As an example of a linear mixed model, suppose we wish to compare mercury levels in walleye within lakes across northern Wisconsin. Walleye caught from the same lake would be expected to have similar levels of mercury compared to walleye from other lakes. The necessary independence assumption in an general linear model would be violated as the responses (mercury levels in walleye) would be correlated at the lake level. Ignoring this correlation will result in biased - upwardly or downwardly depending on the exact correlation structure - variance estimates of slope coefficients, possibly leading to incorrect inference (Liang and Zeger 1993). Essentially, violating the independence assumption means that general linear models could be used, but the estimators will no longer be BLUE.

Mixed models are an alternative to general linear models as they account for correlation between observations. Having correlated observations leads to two major differences in the model. The first is that we would now have a second set of model components known as random effects. Random effects are unobservable random variables with a distributional assumption, typically $\boldsymbol{b} \sim N(\boldsymbol{0}, G)$, where G is the variance-covariance matrix we want to estimate (Hildreth 2016). The second is that the observations are now allowed to be correlated in some way, whether that is by group membership or spatio-temporal correlation of observations through the aforementioned variance-covariance matrix. Incorporating random effects changes our linear model by adding the random effects terms $\boldsymbol{Zb}$:

$$\boldsymbol{y} = \boldsymbol{X\beta} + \boldsymbol{Zb} + \boldsymbol{\varepsilon}$$

where we have q random effects, $\boldsymbol{Z}$ is the $n \times q$ random effects design matrix, and $\boldsymbol{b}$ is a $q \times 1$ vector of random effects. For the walleye testing example, each lake sampled would be represented by an element in the random effects in the vector $\boldsymbol{b}$.

Random effects differ from fixed effects in that the random effects are assumed to be a random sample of all possible levels and values of the variables with the intent to make inferences about the populations which the variables represent (Hildreth 2016). Random effects are incorporated in a model as either random intercepts or slopes. Random intercepts allow for observations in the same group to be correlated, with different group means varying around the overall mean while random slopes allow for a covariate to have a different slope for each group. For the walleye mercury levels, random effects would allow for each lake to have its own baseline mean by using random intercepts.

When using OLS under the assumption that errors normally distributed, parameter estimators are not only BLUE but also are the maximum likelihood (ML) estimators. With mixed models, we can use ML

estimation for the fixed effects as those will be BLUE as before. Estimation of the variance components with maximum likelihood is biased, so we need to use restricted maximum likelihood (REML) estimation to obtain unbiased estimates. With REML, we are modifying the likelihood such that the residuals ($\boldsymbol{y} - \boldsymbol{X\beta}$) are conditional on the random effects. It should be noted that with mixed models estimated with REML, only models with the same mean structure (fixed effects) can be compared (Gałecki and Burzykowski 2013). This restriction means that in doing model selection, likelihood-based methods like AIC can only be used for models with the same fixed effects structure.

1.1.2 Generalized Linear Models

There are times when we would like to model a process but the responses are known to follow a distribution other than a normal distribution. Examples of such responses could be the number of eggs per nest or the presence or absence of an invasive species. While we might be able, in certain cases, to approximate some of these non-normal responses with a normal distribution, we generally need to modify our linear model framework for other response distributions. In these non-normal cases, we can turn to generalized linear models.

Generalized linear models were first introduced in a seminal work by Nelder and Wedderburn in 1972, and further expanded on by McCullagh and Nelder's book *Generalized Linear Models*. There are three distinct elements to a GLM: a distributional assumption, the linear predictor, and the link function. All exponential family distributions have probability density or mass functions that can be expressed in the form:

$$f(y|\mu,\phi) = exp\Big\{\frac{y - b(\mu)}{a(\phi)} + c(y,\phi)\Big\}$$

where μ is the natural or canonical parameter, ϕ is the dispersion parameter, and $b(\theta)$, $a(\phi)$, and $c(y,\phi)$ are real-valued functions and $b(\mu) \geq 0$ (Agresti 2015, Casella and Berger 2002). Generalized linear models allow for the modeling of responses from any distribution with the above form, with the most common distributions used in GLMs being the Binomial, Poisson, Negative Binomial, and Gamma distributions. Linear predictors in GLMs are the same as in general linear models.

The last piece of a GLM is the link function. A link function is a differentiable, monotonic function that maps between the mean of the data and the linear predictor $\boldsymbol{X\beta}$ and allows us to maintain the linearity assumption (Agresti 2015, SAS 2008). Link functions maps between the mean and the linear predictor such that predicted values of the response are in the correct space; the logit link function maps predictions such that $\hat{p} \in (0,1)$ as we would expect with a binomial distribution. Commonly, the canonical link is used with

most GLMs. The canonical link is the function $g(\cdot)$ such that the natural parameter μ is mapped to the linear predictor $\boldsymbol{X\beta}$, and the exact relationship is governed by the assumed distribution of the response (Agresti 2015). With the link function, our GLM can be represented as:

$$g(\boldsymbol{\mu}) = \boldsymbol{X\beta}$$

such that $E(y) = g(\mu) = \boldsymbol{X\beta}$. Without the link function, we would have a non-linear model and would not be able to use the theory from linear models for estimation and inference.

Returning to the walleye, suppose that our walleye from before also were checked for viral hemorrhagic septicemia (VHS), a deadly disease first documented in the Great Lakes region in 2005 which can spread between a number of game fish species (WIDNR 2015). Researchers wish to predict the odds of a fish being susceptible to the virus given a set of covariates. Here, we would be dealing with a non-normal response as the disease will be either present or absent in the fish.

Estimation for GLMs is done with maximum likelihood estimation. With the non-normal response though, we cannot use OLS to estimate the parameters since the likelihood is nonlinear in the parameters. Without a closed form solution, iterative methods for solving the nonlinear likelihood for estimates of $\boldsymbol{\beta}$ like the Newton-Raphson or Fisher Scoring methods are used (Agresti 2015). Also, while OLS cannot be used, it can be shown that from Fisher Scoring an iteratively reweighted least squares method can be derived for maximum likelihood estimation; a good mathematical view of these methods can be found in Agresti's *Foundations of Linear and Generalized Linear Models* (2015).

1.2 Generalized Linear Mixed Models

In situations where responses are both non-normal and correlated, we can use generalized linear mixed models for inference. The first introduction to this class of models was in McCullagh and Nelder's *Generalized Linear Models* text where they analyzed salamander interbreeding with a GLMM. Since their introduction, the study of GLMMs has grown from the introduction of penalized quasi-likelihood for approximate inference to more computationally complex Bayesian methods and non-Bayesian Monte Carlo methods (Breslow and Clayton 1993, Fong et al. 2010, Booth and Hobert 1999).

As the name implies, GLMMs combine elements from both linear mixed models and generalized linear models. These models add a random effects component to the GLM, yielding the following model:

$$g(\boldsymbol{\mu}) = \boldsymbol{X\beta} + \boldsymbol{Zb}$$

where μ is still the mean response through the link function, meaning $E(\boldsymbol{y}|\boldsymbol{b}) = g(\mu) = \boldsymbol{X\beta} + \boldsymbol{Zb}$ (Breslow and Clayton 1993). The random effects, $\boldsymbol{b}$, are still assumed to be unobserved random variables that follow some distribution (typically a normal distribution) with unknown variance components.

For one final example using the walleye, we could suppose that walleye in one lake might be different from walleye in another, so we want to account for that variability when we are estimating the probability that these fish have VHS. Again, we have a non-normal response, but we have added random effects for the clustering of fish within lakes meaning we can use a GLMM for estimation and inference.

Before, when the response was normally distributed, there was no issue with estimation because we could use maximum likelihood to obtain estimates of the fixed effects and use the restricted likelihood for the the variance components. Similar to GLMs, we cannot derive a closed form solutions for any estimates in the GLMM as the likelihood function still is nonlinear in the parameters. The addition of the random effects also complicates matters because with each random effect b, the dimensionality of the integral also increases, adding to the intractability of the likelihood function:

$$L(\beta, D) = \prod_i \int f(y_{ij}|b_i) f(b_i) db_i$$

where $f(Y_{ij}|b_i)$ is the probability distribution of the j^{th} response in the i^{th} random effect and $f(b_i)$ is the distribution of the i^{th} random effect. No closed form solution means that other methods of obtaining the likelihood function must be used, and this is where most of the study in GLMMs has been.

Broadly, inference for GLMMs can be conducted either through approximations of the likelihood or with Bayesian methods. Approximations of the likelihood either focus on approximating the integrand (the function being integrated) like penalized quasi-likelihood or try to approximate the integral itself as in Laplace approximation or Gaussian-Hermite quadrature. Bayesian methods rely on approximating the posterior distribution of the parameters in the GLMM, which is usually accomplished with Markov chain Monte Carlo (MCMC) methods. Section 2 looks at each of the given methods and how they estimate the parameters of GLMMs.

2 Methods of Estimating GLMMs

The theory and implementation of generalized linear mixed models are a fairly new compared to others like ANOVA or general linear regression, with the most development in computational methods for GLMMs occurring in the past twenty-five years. The first wide spread introduction to these models came from Mccullagh and Nelder (1989). Since then, a range of methods from both frequentist and Bayesian perspectives

have been developed for inference on GLMMs.

2.1 Likelihood-based Approaches

Likelihood-based (used interchangeably with frequentist) approaches to GLMMs rely on the likelihood - usually the log-likelihood for simplicity - for use in estimation. As has been pointed out before, the use of the full likelihood function with GLMMs is not possible as there are no closed form solutions. To sidestep this issue, approximations of the likelihood function need to be used. In practice, there are three approximation methods used: Laplace approximation, adaptive Gaussian-Hermite Quadrature, and Penalized Quasi-Likelihood. These three likelihood-based methods are based on of Laplace's method for approximation, but alternatives such as the hierarchical likelihood introduced by Lee and Nelder (1996) have been proposed. The following three methods were chosen as they are the most widely used, being implemented in GLMM estimation in a number of **R** packages as well as in commercial software such as SAS. As the focus of the project is on the performance of the methods and not the derivation of them, please see (Tuerlinckx et al. 2006) and (Breslow and Clayton 1993) for mathematical treatments of the methods.

2.1.1 Laplace Approximation

The Laplace approximation is a quadrature method developed by Laplace and published in 1774 as a method for approximating integrals of the form,

$$\int_a^b f(t)e^{\lambda g(t)}dt$$

where we assume that $g(t)$ is a twice-differentiable function on (a, b) with a maximum in the interval (a, b), both $g(t)$ and $f(t)$ are continuous smooth functions, and $f(t)$ is nonzero at t_0 (Tuerlinckx et al. 2006). The idea behind Laplace's method is that for large λ, the contribution of the integral from near some specified point, t_0, will be the bulk of the integral (Tuerlinckx et al. 2006). Using a second-order Taylor series expansions for $g(t)$ and $f(t)$, the resulting integral can be shown to be the kernel of a normal distribution, which can then be integrated. Looking at the form above we can see that the integrand in the function is similar in nature to the likelihood of a GLMM, which contain exponential functions from the exponential family of probabilty distributions. This process of approximation through expansion forms the mathematical basis for the other two likelihood-based methods as well.

One restriction of Laplace approximation, and Gaussian-Hermite quadrature, is that these approximations are only considered for normally distributed random effects, meaning other approaches such as Bayesian methods or Monte Carlo approximations need to be used for non-normal random effects. A variation on the

Laplace approximation which uses a sixth-order Taylor series expansion has been shown to improve the accuracy of the approximation, being equivalent to adaptive Gaussian-Hermite quadrature and outperforming penalized quasi-likelihood, but there seem to be no software packages that utilize this approach (Raudenbush et al. 2000, Tuerlinckx et al. 2006). Implementation of the Laplace approximation for GLMM estimation in **R** can be carried in a few different packages, namely `lme4`, `glmmML`, and `repeated`, but we will focus on using the `glmer` function in the `lme4` package (Bolker 2013b).

2.1.2 Adaptive Gauss-Hermite Quadrature

Gaussian-Hermite quadrature (GHQ) is, at its heart, a generalization of Laplace approximation. Instead of evaluating the approximation at just t_0, a number of points can be taken to approximate an integral (Tuerlinckx et al. 2006). In general, Gaussian-Hermite quadrature methods can yield near exact approximations if the points where the approximation is evaluated, known as nodes, are optimally spaced. The number of nodes to use for the approximation varies depending on the literature source, but the locations of nodes are determined algorithmically.

Regular GHQ is susceptible to bias if the cluster sizes or variances are large, which is why a variation on GHQ called adaptive GHQ (AGHQ) is often implemented (Tuerlinckx et al. 2006). In adaptive GHQ, the positions of the nodes are chosen conditional on the best estimates of the random effects and the estimates of the fixed effects and variance-covariance matrix (Tuerlinckx et al. 2006). Adaptively picking node positions allows for integrals to be evaluated in the regions that contribute the most to the integral, instead of standard node locations which rely on normal distribution kernels. The result is that less nodes are needed to approximate an integral, leading to increases in accuracy and computational efficiency. Using adaptive GHQ for GLMM estimation in **R** can be done with the same packages that implement Laplace approximation. In `glmer`, this is accomplished by setting the number of quadrature nodes to be greater than one.

It might seem that adaptive GHQ should always be used to obtain a more accurate approximation of the likelihood integral than Laplace approximation, but there are limitations to its usage. Both non-adaptive and adaptive GHQ become computationally intense as the number of random effects grows, which is not an issue with Laplace approximations (SAS 2008). With enough random effects, the likelihood is a high dimensional enough that they cannot be approximated with GHQ or AGHQ. The structure of the random effects is also a limiting factor, with `glmer` only being able to fit a single scalar random effect where the covariance structure is assumed to be $G = \sigma_G^2 \boldsymbol{I}$ (constant variance) with quadrature (Bates et al. 2015). This excludes covariance structures such as crossed random effects, models without subjects, or non-nested subjects (SAS 2008). Laplace approximation and its cousin penalized quasi-likelihood then are more flexible

and slightly more computationally than adaptive GHQ, but for suitablely structured data, adaptive GHQ should yield more accurate estimates.

2.1.3 Penalized Quasi-Likelihood

Penalized quasi-likelihood (PQL) was the first widely used method of estimating parameters for a GLMM, which works through approximating the likelihood integrand. Penalized quasi-likelihood was introduced by Breslow and Clayton (1993) as a modified Laplace approximation of the marginal quasi-likelihood, yielding a normally distributed approximate likelihood. The "penalized" in PQL refers to the random effects being biased towards zero in the approximation. To fit the model from the approximated likelihood, linear mixed models using REML are iteratively fit to the approximate likelihood until convergence is met.

Compared to the previous methods, PQL is the fastest and most flexible approach for estimating GLMMs. The number of random effects and their structure are not as restricted as in AGHQ and should have asymptotic properties as the sample size increase (Breslow and Clayton 1993). Though the fastest, it is the least accurate of the common likelihood-based methods, especially with small datasets (Bolker 2013a). Breslow and Clayton noted that PQL estimation can yield inconsistent estimates of the fixed effects when cluster sizes are small for binomial GLMMs and confirmed the inconsistency in a simulation study (Breslow and Clayton 1993, Tuerlinckx et al. 2006). Since penalized quasi-likelihood is based on the quasi-likelihood, standard likelihood-based tests such as likelihood ratio tests cannot be performed (Tuerlinckx et al. 2006, Zhang et al. 2011).

As a side note, penalized quasi-likelihood and pseudo-likelihood often appear in the literature as interchangeable terms without any delineation between the two with regards to GLMMs. Both methods are different, but they will yield identical parameter estimates as they rely on different expansions with the only difference being an additional constant in pseudo-likelihood (Capanu et al. 2013).

2.2 Bayesian Approach

The Bayesian approach to GLMMs differs from frequentist methods in a few ways and are commonly called hierarchical models. First, Bayesians assume that parameters are random variables and have probability distributions, in contrast to how frequentists treat parameters as fixed constants. In Bayesian methods, inference is conducted through the posterior distribution, which combines information from the probability of the data given the parameters - essentially the likelihood - and the prior distributions of the parameters. Prior distributions incorporate believed information about the parameters and can either be informative, meaning the prior has more weight than the data in the posterior, weakly-informative, where there is some information,

or non-informative. For simple cases where the components of the model are normally distributed or we can use conjugacy of the distributions of the data and the prior, the posterior distribution can be found exactly. In the case of GLMMs though, Bayesian approaches have an issue similar to the likelihood methods in that there is no closed form for the posterior distribution. The Bayesian solution to this problem lies in using Monte Carlo Markov Chain methods to approximate the posterior distribution with the conditional distributions that comprise the posterior.

Markov chain Monte Carlo (MCMC) methods are a large group of stochastic approaches whose goal in Bayesian methods is to numerically integrate the posterior distribution. Markov chains are stochastic processes for which the current state depends solely on the previous state (Hoegh 2017). Monte Carlo methods use random sampling from distributions to produce numerical approximations of mathematical quantities (Hoegh 2017). For each iteration of a MCMC procedure for GLMMs, Monte Carlo samples are drawn from conditional distributions which have parameters dependent upon the previous iterations estimations, making the dependence the Markov chain component.

In building a Bayesian GLMM, we are essentially combining a within-group model (equivalent to the random effects components) and a between-groups model (equivalent to fixed effects). To approximate the posterior distributions, we draw MCMC samples from the full conditionals in the posterior distribution, but we have to employ two separate algorithms to do so: a Gibbs sampler for the within-group model and the Metropolis algorithm for the between-group model. Together, these two algorithms make the Metropolis-Hastings algorithm.

Metropolis-Hastings Algorithm

The Metropolis-Hastings algorithm is a two-part algorithm, combining a Gibbs sampling step for the normally distributed within-group and variance components and a Metropolis step for the between group slope coefficients. Given full conditional distributions, prior distributions, and current values at step s for the GLMM, the Metropolis-Hastings algorithm is as follows from Hoff (2009):

1. Sample the within-group means, $\boldsymbol{\theta}^{(s+1)}$, from its full conditional distribution.
2. Sample the variance-covariance matrix, $\boldsymbol{\Sigma}^{(s+1)}$, from it's full conditional distribution.
3. Sample the between group slope coefficients, β_k^*, conditional on $\boldsymbol{\theta}^{(s+1)}$ and $\boldsymbol{\Sigma}^{(s+1)}$. Then, for each β_k, determine whether to accept the new β_k^* (set $\beta_k^{(s+1)} = \beta_k^*$) or keep $\beta_k^(s)$ depending on the acceptance ratio.

With Bayesian methods, inference about parameters is not limited to point estimates and associated

measures of variability. Credible intervals, the Bayesian analogue to confidence intervals, give the probability a given parameter is within the interval. The posterior predictive distribution gives the predicted distribution of responses conditioned on the already observed data (Hoegh 2017). The posterior predictive distribution can also be extended for checking if the priors or assumed distribution of the data are appropriate for the model. For this project though, we only consider the point estimates for comparison with the frequentist estimation methods.

A major downside to Bayesian methods tend to have a much steeper learning curve than the likelihood-based methods. It generally takes more advanced knowledge to write the MCMC samplers for complex models like a GLMM, so many often opt to use software packages instead. Packages in **R** are available to implement Bayesian GLMMs, with options ranging from focusing on estimation of GLMMs (`MCMCglmm`) to being interfaces between other programs that are used for general Bayesian inference (`rjags, glmmBUGS`) (Bolker 2013b).

2.3 Other Monte Carlo Based Methods

Monte Carlo based methods are a necessity of Bayesian statistics, but have also been incorporated in likelihood-based estimation methods for GLMMs. Most Monte Carlo approaches either use Monte Carlo methods to approximate the full GLMM likelihood function or use Monte Carlo integration coupled with an Expectation-Maximization (EM) algorithm to estimate a model (Booth and Hobert 1999). The motivation behind these methods, introduced in the 1990's, was to avoid the pitfalls of approximation methods such as inconsistent estimates, the ability to model higher order random effects, and to model non-normal random effects.

Packages in **R** are available to implement these methods, with the `glmm` package using Monte Carlo likelihood approximation and `MCEMglmm` using an Monte Carlo EM approach. Initially, one of these two packages were intended to be used in the simulation, but the lack of documentation made implementation difficult and resulted in Monte Carlo estimation methods being excluded.

3 Simulation Study of Performance of Estimation Methods

Assessing the accuracy and precision of the discussed estimation methods was conducted through a simulation study. Data were taken from a study on owlet begging (Roulin and Bersier 2007), by way of the book *Mixed Effects Models and Extensions in Ecology in R* (Zuur et al. 2009). Since model selection was not of interest, the model used in the simulations was also from Zuur et al.'s analysis of the owlet data (Zuur et al. 2009). Empirical sampling distributions of fixed effects estimates and the random effects variance estimates are

used to assess the bias and precision. All **R** code follows in an appendix.

3.1 Data Description

A number of simulation studies exploring the properties of GLMM estimation methods have been performed (Zhang et al. 2011, Nelson and Leroux 2008, Capanu et al. 2013) are good examples), but few use actual data for simulated the data structure. With simulated data, all aspects of the data structure can be controlled and specific characteristics of estimation methods such as downwardly biased variance estimates from PQL on binary data can be explored; however, data from studies often contain data structures like heterogeneous cluster sizes and overdispersion that effect estimation. Studying the performance of estimation methods with messier data structures is necessary as these complex structures are likely to be seen in ecology and evolutionary biology data analyses.

The goal of the original study, *Nestling barn owls beg more intensely in the presence of their mother than in the presence of their father*, was to study exactly what the title says- if barn owl nestlings exhibit different begging behavior depending on which parent feeds the nestlings. For the study, twenty-seven barn owl nests in western Switzerland during the summer of 1997 (Roulin and Bersier 2007). The experimental set up was fairly involved, with the study design described below.

1. Twenty-seven nests were each equipped with audio and video recorders to record parental arrival time and nestling calls.

2. Parents were differentiated with leg bands (females on their left leg, males on the right) for parental recognition and feeding rates.

3. During the morning, around 0900 hours, nestlings were either fed by the experimenters (food satiated) or had food in the nest removed (food-deprived).

4. At night from 2130 hours to 0530 hours, parental feedings were observed with the parental arrival time and duration of stay recorded along with the number of calls from the owlets.

5. In the 30 seconds prior to parent arrival, the number of calls per nestling (called sibling negotiation) was recorded. This is the dependent variable for this analysis.

In total, there are 599 observations in the dataset, with the number of observations ranging from four to fifty-two observations per nest. The number of calls by arrival time and nest are plotted below.

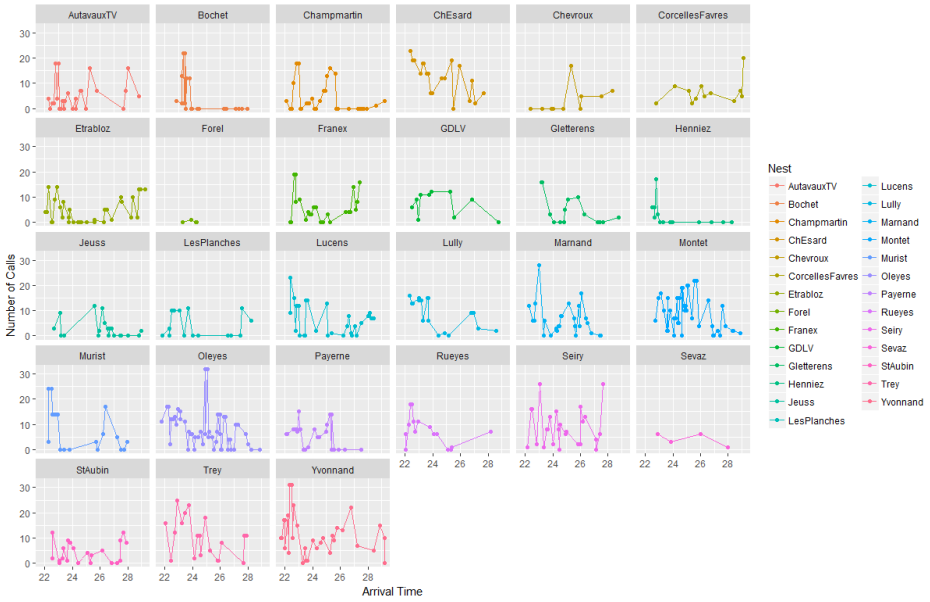

Figure 1. Plot of number of barn owl nestling calls in a 30 second interval before parent arrival by arrival time, grouped by nest.

It is important to note that we are not expecting observations at each nest to be independent. Owlets within a nest could have any number of reasons why they exhibit the same calling patterns, from learned behavior to inherited traits. Because of the dependence within each nest, nests are being treated as random effects using random intercepts to modify the mean structure. Also, we would expect the response, the discrete number of calls within an interval, to follow a Poisson distribution.

3.1.1 Basis Model

The model used in the simulation study comes from an analysis by Zuur et al. (2009) and models the mean number of calls (μ_{ij}) as a function of the parent arrival time, the log of the brood size, whether the owlets were food deprived, and the effect of nest, employing the log link function:

$$\begin{aligned} log(\mu_{ij}) =& \beta_0 + \beta_1 \times \log(\text{BroodSize}_{ij}) + \beta_2 \times \text{FoodTreatment}_{ij} \\ &+ \beta_3 \times \text{ArrivalTime}_{ij} + \text{Nest}_i \end{aligned}$$

where j denotes the j^{th} observation from the i^{th} nest, and $Nest_i \overset{iid}{\sim} N(0, \sigma^2_{nest})$. Section 13.2.2 from Zuur (2009) details the model selection process to get to this model, although the offset term is removed and the log of brood size and an intercept term are fit instead. The figure below shows there are strong signals for the fixed effects and that the assumption of a Poisson distributed response seems reasonable, noting that the plot aggregates across nests.

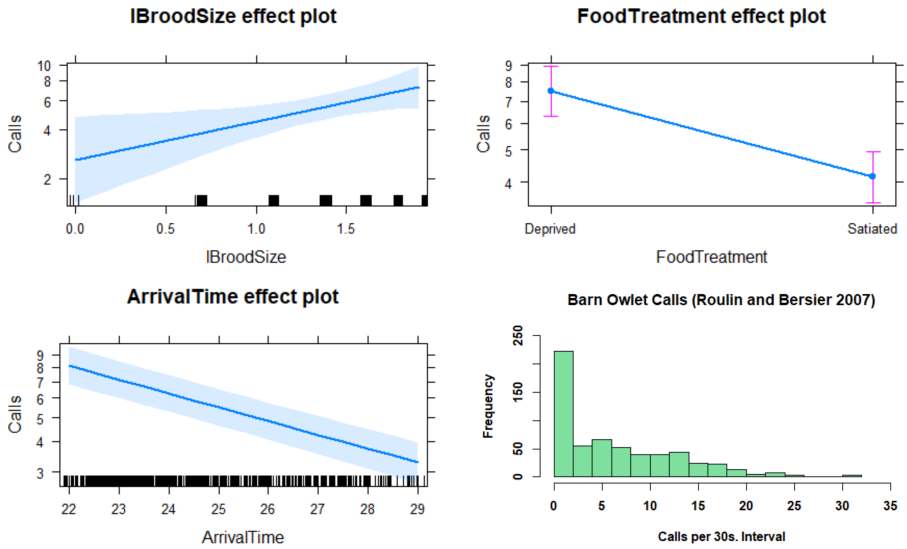

Figure 2. Effects plots of the three fixed effects in the owlet GLMM and the distribution of the response, number of calls.

In the random effects plot below, it seems that including random effects is a good modeling choice as there is a fair amount of variation in the estimates of nest effects. Note the standard errors of the estimates; the nests with the most variability are nests with a relatively small (< 10) number of observations.

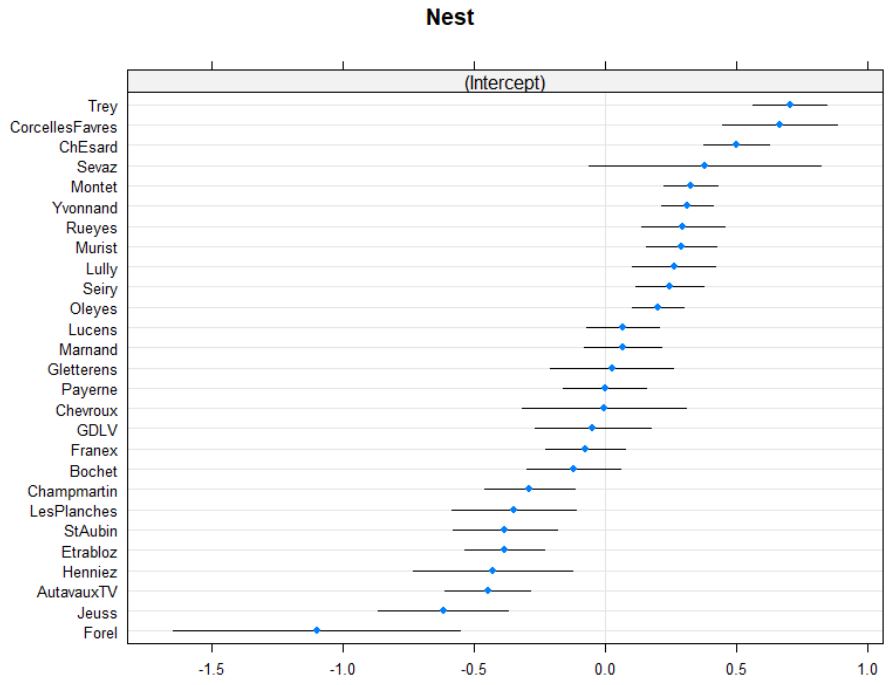

Figure 3. Plot of the random intercept estimates by nest with standard errors.

The estimates from this model (fit using 25 adaptive GHQ nodes with `glmer`) were used in the generation of simulated responses for the first simulation, with a table of the relevant estimates below.

Effect	Estimate	Standard Error
Intercept	4.432	0.384
log(Brood Size)	0.538	0.220
Food Treatment	-0.589	0.036
Arrival Time	-0.129	0.009
$\hat{\sigma}_{Nest}$	0.4206	

3.2 Data Generation

For the simulation study, new responses based off of estimates from a single `glmer` model fit were generated with the `simr` package and then models were fit using the original covariates. Using this package, new responses can be simulated from an `lme4` model object, where the estimates from the model object are treated as the parameters for the simulation distributions. One caveat of this method is that random effects are not simulated using the estimates from the model object, rather they seem to be simulated from a $N(0, \hat{\sigma}_{RE})$ distribution, where $\hat{\sigma}_{RE}$ is the estimated random effect standard deviation from the input model and $\boldsymbol{I}$ is a 27×27 identity matrix. Three different scenarios were simulated for the study, with the intent to explore the effect of sample size:

1. Full owlet dataset, with all 27 nests.
2. Half owlet dataset, with 14 nests randomly selected to make a reduced dataset.
3. Quarter owlet dataset, with 7 nests randomly selected for a reduced dataset.

The datasets remained the same for all three scenarios with the exception of arrival time; with the half and quarter datasets, arrival time was centered in hopes of preventing convergence issues when fitting the model that produced the simulation parameters. For all three scenarios, 250 simulations were created. At each iteration of the simulation, a new response vector was simulated and coupled with the rest of the explanatory variables.

3.3 Model Estimation

With each new set of simulated responses, four identical models were fit, each employing one of the four models discussed in Section 2. For each model fit, estimates of the fixed effects and the random effects variance are stored for later use. Descriptions of how each method was implemented follows:

1. **PQL**: The `glmmPQL` function in the `MASS` package was used for PQL estimation. The default number of iterations (10) was used (Venables and Ripley 2002).

2. **Laplace Approximation**: The `glmer` function in the `lme4` package was used for Laplace approximation estimation. The number of quadrature points was set to 1 (`nAGQ = 1`) to ensure LA was used and the default Nelder-Mead optimizer was used (Bates et al. 2015).

3. **Adaptive Gaussian-Hermite Quadrature**: The `glmer` function in the `lme4` package was used for adaptive GHQ estimation. The number of quadrature points was set to 25 and the default Nelder-Mead optimizer was used (Bates et al. 2015).

4. **Bayesian MCMC**: The `rjags` packages (with JAGS 4.3) was used for Bayesian estimation. JAGS code was adapted from Zuur (2013), with preliminary model fitting showing that with four chains, 130000 MCMC iterations of burn-in was sufficient for adaptation and another 50000 iterations for drawing samples, trimming to 10000 samples overall, resulted in well mixed chains. For the quarter dataset models, 150000 iterations were used for burn-in. For all Bayesian models, non-informative priors were used for $\boldsymbol{\beta}$ and $\boldsymbol{\Sigma}$, with the Half-Cauchy prior distribution of σ_{nest} being a recommended weakly-informative prior distribution for the random effects variance (Gelman et al. 2006). The specific priors are:

   a) $p(\boldsymbol{\beta}) \sim N(\mathbf{0}, 10000\mathbf{I})$

   b) $p(\mathbf{b}) \sim N(\mathbf{0}, \sigma^2_{nest}\mathbf{I})$

   c) $p(\sigma_{nest}) \sim$ Half-Cauchy(2) and one simulation configuration used the prior $p(\sigma_{nest}) \sim \text{HC}(0.5)$

All models were tested on the original datasets prior to the simulations, but due to the variability that can arise in simulating data, warnings such as "insufficient adaptation" for the JAGS models or "failing to reach convergence" with the frequentist models did occur. Future work would include creating an adaptive function for this study to try and avoid these convergence issues.

3.4 Results

As would be expected because of the large sample size, the full dataset had estimates that were the mostly unbiased and precise, as can be seen in Figure 4. The red horizontal lines note the true parameter value used for simulating the responses.

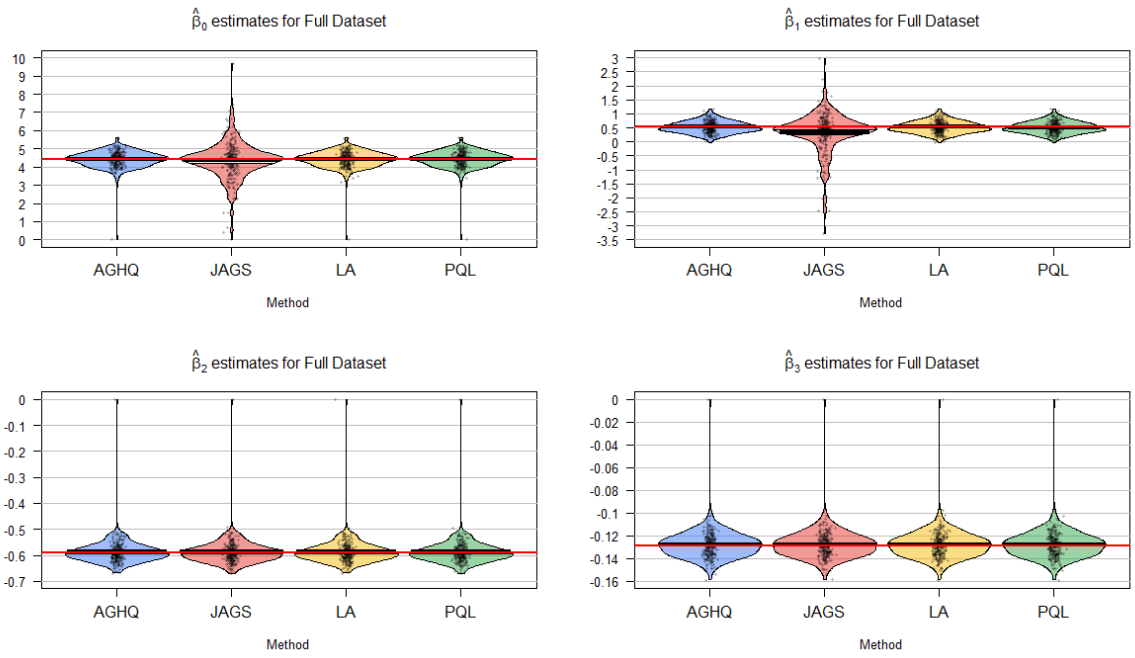

Figure 4. Plot of all fixed effect estimates for full owlet dataset.

The empirical sampling distributions of the fixed effects estimates showed the least amount of bias for all sections of the study, with the Bayesian models exhibiting the most bias and variability of the four as seen in the plot of *log*(Brood Size) estimates in Figure 5 below. Bias is being assessed by comparing the central tendency of the sampling distributions (thick black lines in the centers of the distributions) to the true parameter values.

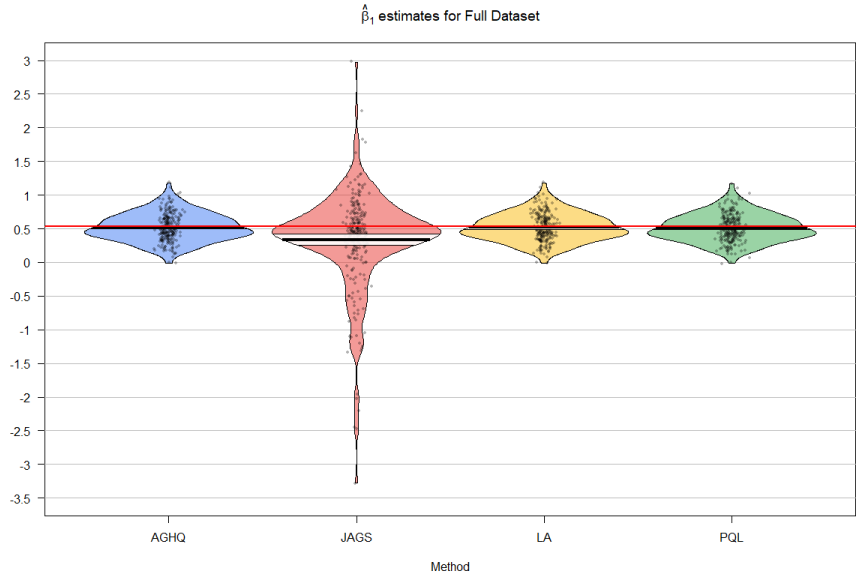

Figure 5. Plot of *log*(Brood Size) effect estimates.

Looking further at the estimates of the random effects variance component in Figure 6, we can see that the Bayesian method is strongly upwardly biased in comparison to the other methods. The distribution of Bayesian $\hat{\sigma}_{re}$ also has a much larger variance than the frequentist methods with the full owlet dataset. There are a few estimates throughout all the simulations that are zeros across methods for $\hat{\sigma}_{re}$ and it's likely that those simulations are simply unusual simulations.

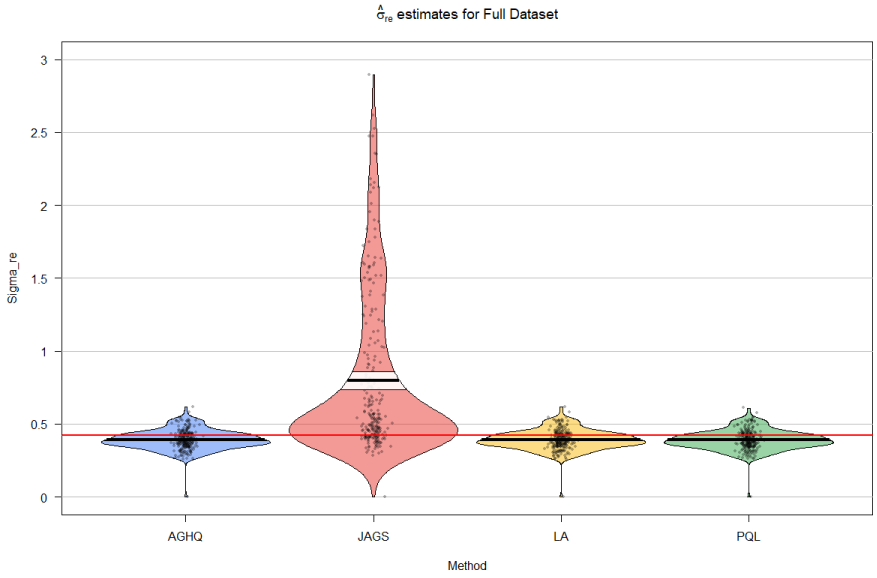

Figure 6. Plot of $\hat{\sigma}_{nest}$ variance estimates.

As the overall sample size decreases, we do see a few changes across methods. In the half dataset, the Bayesian fixed effects estimates are more consistent in variability with the likelihood-based methods. The distributions for the intercept and log(Brood Size) effect estimates are similar to those in the full dataset. The slight bias present in the Bayesian estimates for those two effects are also diminished with the half dataset.

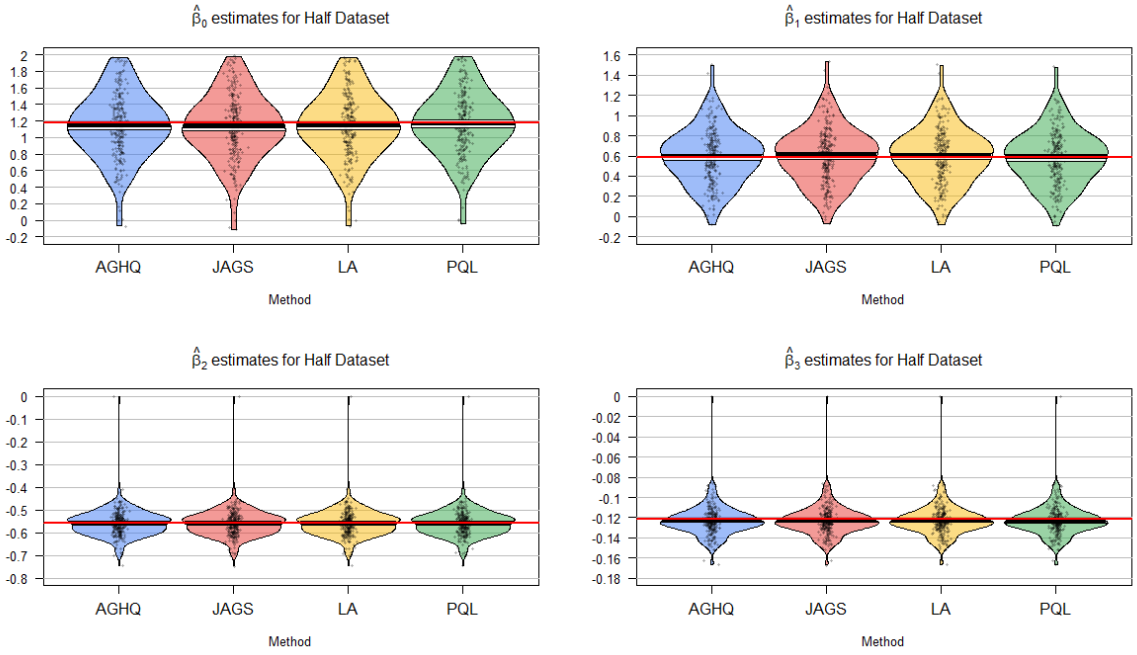

Figure 7. Plot of all fixed effect estimates for a random subset by nest (fourteen nests) of the owlet dataset.

Much like the fixed effects, the estimates of $\hat{\sigma}_{nest}$ are much more consistent with the half dataset, which is interesting as the priors used for the Bayesian models did not change. The large degree of bias seen in the full dataset with the Bayesian estimates is now much smaller; however, there is a systematic bias that is showing with the likelihood-based methods that was not present in the estimates from the full dataset, to the point where the Bayesian method seems to edge out those methods in terms of bias.

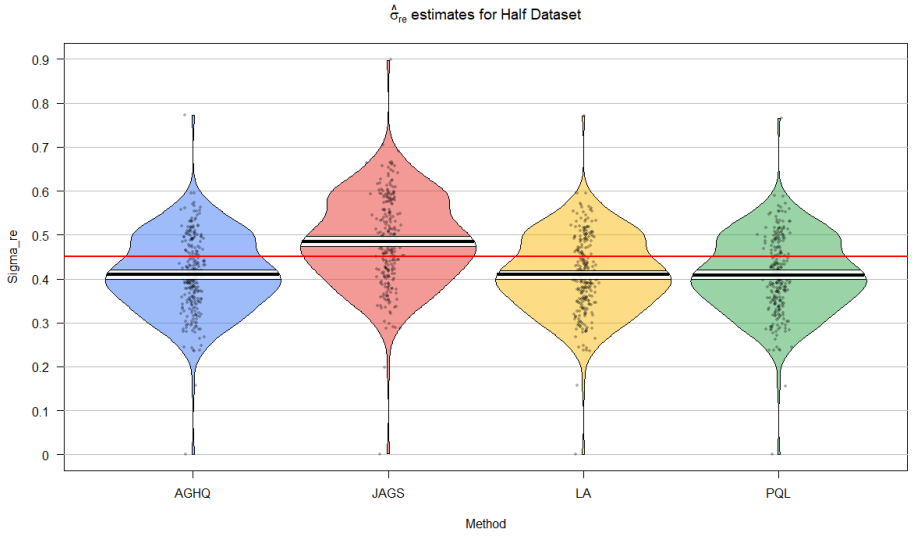

Figure 8. Plot of $\hat{\sigma}_{nest}$ variance estimates for a random subset by nest (fourteen nests) of the owlet dataset.

A similar story is found in the estimates for the smallest subset of the data, the quarter dataset with

just seven nests. Fixed effects estimates are nearly identical in distribution across the four methods, with similar small amounts of upward or downward bias depending on the effect being estimated. While not directly comparable between datasets, the direction of the bias for the quarter and half datasets are similar for the four fixed effects and the random effect variance component.

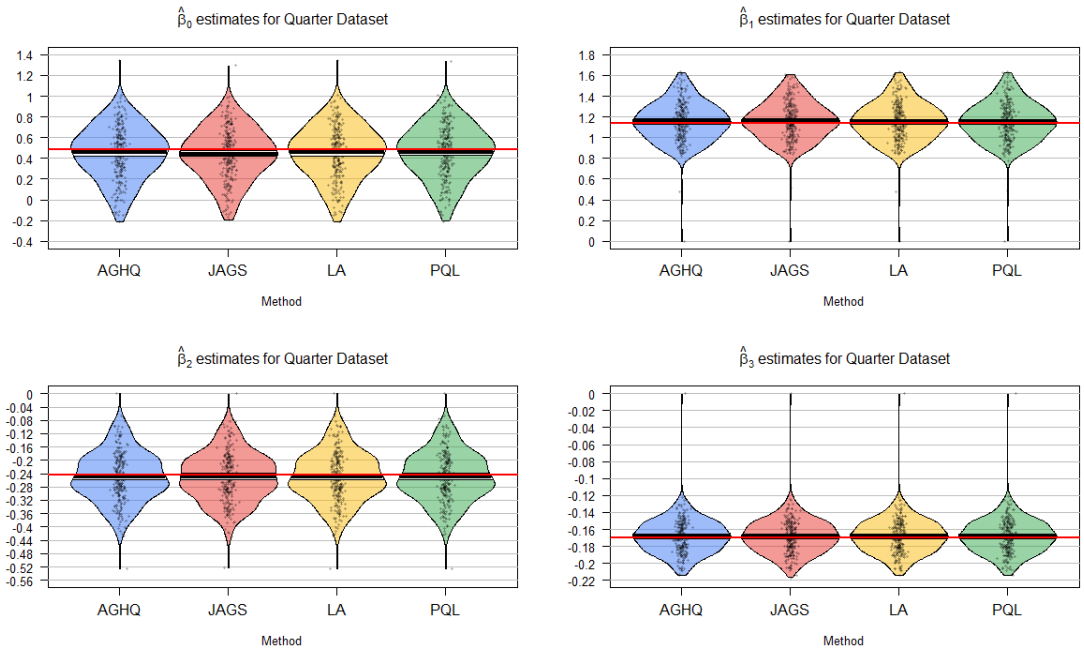

Figure 9. Plot of all fixed effect estimates for a random subset by nest (seven nests) of the owlet dataset.

The sampling distributions of the estimated variance component in the quarter dataset again demonstrate the biases seen in the half dataset, with the likelihood methods downwardly biased about twice as much as the Bayesian method is upwardly biased.

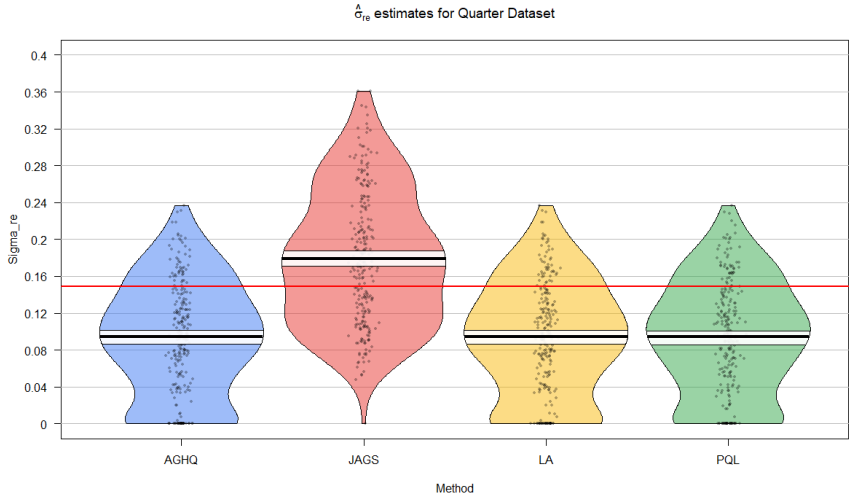

Figure 10. Plot of $\hat{\sigma}_{nest}$ variance estimates for a random subset by nest (seven nests) of the owlet dataset.

To see the effect of a different prior on the Bayesian estimates, the same datasets were again fit with a Half-Cauchy(0.5) prior, and plots of those results are in the appendix in figures 1a and 2a. Overall, there was no noticeable difference in the bias or precision of the Bayesian methods across the estimates other than a slight decrease in bias for the variance component estimates.

4 Discussion

In most of the literature, the properties of estimators for GLMMs have either not been explored or focus on just one type of model, typically binomial GLMMs (Zhang et al. 2011). With this simulation study, there were a few expected results, such as the consistency and precision of estimates for fixed effects. The downward bias seen in the variance components by all the likelihood-based methods was not expected and is one of the main areas of discussion.

Estimates of the fixed effects were not strongly biased or imprecise, with the exception of slight bias in a few of the Bayesian estimates. Even with decreased sample size, the empirical sampling distributions of estimates of the fixed effects showed minimal bias and reasonable precision. It does seem intuitive that as sample size increases, there should be some asymptotic properties for the likelihood-like estimators. There has been work on asymptotic properties of GLMMs (Pinheiro 1994), but most usually confined to subdisciplines (Jiang 2017). Based off of the results of the simulation study, it seems reasonable in this case to assume that inference relying on asymptotics would be sound for the fixed effects.

It is in the distributions of the variance components where the simulation study is most interesting. While it is known that PQL can yield downwardly biased variance estimates for binomial GLMMs, the consistent downward bias for all three likelihood based methods as the sample size decreased is interesting. Most research on GLMM estimation methods and potential bias is related to binomial GLMMs and there is not much in the literature suggesting that bias like this would be present for the other methods with a Poisson GLMM. Since all three of the likelihood-based methods all stem from Laplace approximation, there might be an issue with the compatibility of the data with the estimation methods. While these estimators are precise, a cautious practitioner might search for more information on estimation of variance components before fitting a GLMM using one of the three likelihood-based methods if they have a small sample size.

Related to the likelihood bias, the consistent upward bias in the Bayesian variance estimates is an intriguing result. In the first run of the simulation study, a misunderstanding of the Half-Cauchy distribution resulted in a large bias compared to the likelihood methods. Changing to a more reasonable prior distribution for the variance component improved the bias of the estimates, but also increased the precision of estimates of the fixed effects. With the fixed effects being computed conditional on the draws from the distributions of

the within group random effects, it is likely that the variability due to the misspecified prior distribution had a cascading effect that carried through to the between group, fixed effects. Still, the upward bias exhibited in the estimates could be indicative of the Bayesian estimators incorporating information differently and estimating more variance. Also, this hopefully demonstrated the important of properly researching a suitable prior distribution and that misspecification in one area of a model can impact other parts.

Also interesting with the Bayesian methods is that the bias seen in the variance component is the opposite of that of the likelihood-based methods. Initially, the bias seen could be attributed to the prior distribution, as the median of a Half-Cauchy(2) distribution is 2. In changing the prior distribution to a Half-Cauchy(0.5) and refitting the same models, the apparent bias only decreases by a small amount if at all. It is possible then that the difference is either in how the Bayesian model is incorporating the data into the model or just the moderating effect of the prior. Either way, Bayesian methods for GLMMs then might be seen as an alternative to likelihood-based methods for small sample sizes.

5 Conclusions

Overall, it seems that with a large enough sample size, the common methods for fitting GLMMs are robust enough to yield accurate and relatively precise estimates. Bayesian methods need some additional attention as improper prior specification or inadequate MCMC sampling can drastically impact results, but they also might be better suited to small sample problems. The likelihood-based methods are easier to implement and much quicker, but the apparent bias in estimates of variance components is concerning and worth more investigation.

The practical implications for the project are two-fold. First, with a large enough sample size, there probably are no restrictions on which methods one can use for estimating GLMMs. Even though PQL has been maligned in the past, it appears to be serviceable by these results and in some applications where more complex random effects structures are needed it could be an alternative to a far more complicated Bayesian model. Second, the results show that there might be unstudied issues with the likelihood-based methods that cause bias in estimates. The bias present in estimates of the variance components could impact inference and should be a cause for concern and subject to further study.

5.1 Future Work

Throughout this project, a number of potential improvements or extensions of have arisen. The first of which would be to vary the true parameters used for simulating to test the estimation methods performance in situations such as having a larger variance component or having larger effect sizes. The `simr` package has

a great deal of flexibility in simulating data structured for analysis with GLMMs, specifically in changing characteristics such as random effects variance or slopes of the fixed effects. Doing more simulations with altered parameters would be of interest as this would help investigate issues with approximation based methods and large random effects variances.

Related to changing individual components, incorrect assumptions about the distribution of the response or random effects structure could reveal more about the robustness of the methods. Additionally, the dataset used here had a wide range of observations per cluster, ranging from 4 to 52. Simulating datasets that vary in cluster sizes to different degrees might be useful in diagnosing the bias seen in the frequentist methods variance estimates.

As previously mentioned, rewriting the simulation function to be more flexible to prevent convergence issues and allow the JAGS algorithm to fully adapt the Metropolis-Hastings algorithm will be a future goal. Doing so would help ensure that any effects seen in the simulation results would not be from computational issues.

6 References

Alan Agresti. *Foundations of linear and generalized linear models*. John Wiley & Sons, 2015.

Douglas Bates, Martin Mächler, Ben Bolker, and Steve Walker. Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, 67(1):1–48, 2015. doi: 10.18637/jss.v067.i01.

Benjamin Bolker. Generalized linear mixed models, 2013a. URL `https://ms.mcmaster.ca/~bolker/classes/s4c03/notes`. Stats 4C03/6C03 Course Notes, McMaster University.

Benjamin Bolker. Package comparison, 2013b. URL `glmm.wikidot.com/pkg-comparison`. Mixed Modeling Software Package Comparison.

James G Booth and James P Hobert. Maximizing generalized linear mixed model likelihoods with an automated monte carlo em algorithm. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(1):265–285, 1999.

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

Marinela Capanu, Mithat Gönen, and Colin B Begg. An assessment of estimation methods for generalized linear mixed models with binary outcomes. *Statistics in medicine*, 32(26):4550–4566, 2013.

George Casella and Roger L Berger. *Statistical inference*, volume 2. Duxbury Pacific Grove, CA, 2002.

Julian J Faraway. *Linear models with R*. CRC press, 2014.

Youyi Fong, Håvard Rue, and Jon Wakefield. Bayesian inference for generalized linear mixed models. *Biostatistics*, 11(3):397–412, 2010.

Andrzej Gałecki and Tomasz Burzykowski. *Linear mixed-effects models using R: A step-by-step approach*. Springer Science & Business Media, 2013.

Andrew Gelman et al. Prior distributions for variance parameters in hierarchical models (comment on article by browne and draper). *Bayesian analysis*, 1(3):515–534, 2006.

Peter Green and Catriona J. MacLeod. simr: an r package for power analysis of generalised linear mixed models by simulation. *Methods in Ecology and Evolution*, 7(4):493–498, 2016. doi: 10.1111/2041-210X.12504. URL `https://CRAN.R-project.org/package=simr`.

Laura Hildreth. Mixed effects models, 2016. STAT 506 Course Notes, Montana State University.

Andrew Hoegh. Markov chain monte carlo, 2017. STAT 532 Course Notes, Montana State University.

Peter D Hoff. *A first course in Bayesian statistical methods*. Springer Science & Business Media, 2009.

Jiming Jiang. *Asymptotic analysis of mixed effects models: theory, applications, and open problems*. CRC Press, 2017.

Youngjo Lee and John A Nelder. Hierarchical generalized linear models. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 619–678, 1996.

Kung-Yee Liang and Scott L Zeger. Regression analysis for correlated data. *Annual review of public health*, 14(1):43–68, 1993.

P McCullagh and JA Nelder. *Generalized Linear Models*. Chapman & Hall, 1989.

Robert A McLean, William L Sanders, and Walter W Stroup. A unified approach to mixed linear models. *The American Statistician*, 45(1):54–64, 1991.

Kerrie P Nelson and Brian G Leroux. Properties and comparison of estimation methods in a log-linear generalized linear mixed model. *Journal of Statistical Computation and Simulation*, 78(3):367–384, 2008.

José Carlos Pinheiro. *Topics in mixed effects models*. PhD thesis, University of Wisconsin, Madison, 1994.

Stephen W Raudenbush, Meng-Li Yang, and Matheos Yosef. Maximum likelihood for generalized linear models with nested random effects via high-order, multivariate laplace approximation. *Journal of computational and Graphical Statistics*, 9(1):141–157, 2000.

George K Robinson. That blup is a good thing: the estimation of random effects. *Statistical science*, pages 15–32, 1991.

Alexandre Roulin and Louis-Felix Bersier. Nestling barn owls beg more intensely in the presence of their mother than in the presence of their father. *Animal Behaviour*, 74(4):1099–1106, 2007.

SAS/STAT(R) 9.2 User's Guide. SAS Institute Inc., 2008.

Francis Tuerlinckx, Frank Rijmen, Geert Verbeke, and Paul Boeck. Statistical inference in generalized linear mixed models: A review. *British Journal of Mathematical and Statistical Psychology*, 59(2):225–255, 2006.

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S.* Springer, New York, fourth edition, 2002. URL `http://www.stats.ox.ac.uk/pub/MASS4`. ISBN 0-387-95457-0.

WIDNR. Viral hemorrhagic septicemia fish virus, 2015. URL `https://dnr.wi.gov/topic/Fishing/vhs/index.html`.

Hui Zhang, Naiji Lu, Changyong Feng, Sally W Thurston, Yinglin Xia, Liang Zhu, and Xin M Tu. On fitting generalized linear mixed-effects models for binary responses using different statistical packages. *Statistics in Medicine*, 30(20):2562–2572, 2011.

AF Zuur, EN Ieno, NJ Walker, AA Saveliev, and GM Smith. Mixed effects models and extensions in ecology. *New York, NY: Spring Science and Business Media*, 2009.

Alain F Zuur, Joseph M Hilbe, and Elena N Ieno. *A Beginner's Guide to GLM and GLMM with R: A Frequentist and Bayesian Perspective for Ecologists.* Highland Statistics Limited, 2013.

7 Appendix

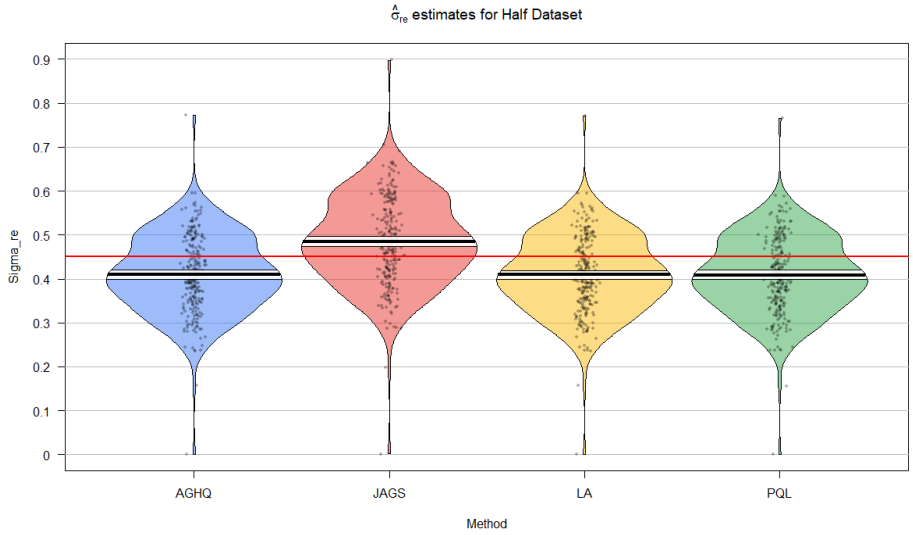

Figure 1A. Plot of $\hat{\beta}$ estimates for a random subset by nest (seven nests) of the owlet dataset with a Half-Cauchy(0.5) prior distribution on $\hat{\sigma}_{nest}$.

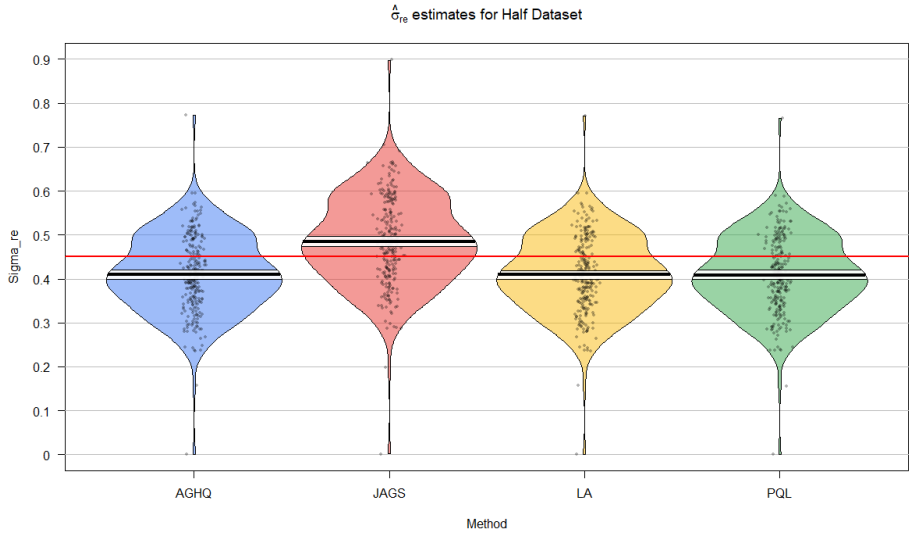

Figure 2A. Plot of $\hat{\sigma}_{nest}$ variance estimates for a random subset by nest (seven nests) of the owlet dataset with a Half-Cauchy(0.5) prior distribution on $\hat{\sigma}_{nest}$.

8 R Code Appendix

An **R** file with all code can be found at the GitHub repository: `https://github.com/jakelrich/MSU_WritingProject`.

```
####################################
#Jake Rich
#Writing Project Code
#Spring 2018
#No guarantees on first compile success.
####################################
###Setting Up R Environment
#Installing necessary packages if not already installed.
######
#NOTE: You must have JAGS installed before using any of the Bayesian analyses
#in this script.
######
list_of_packages <- c("lme4","MCMCglmm","coda","glmm","ggplot2","RCurl", "simr","devtools
    ",
                      "rjags","mcemGLM","MASS","glmmML","GLMMmisc","nlme","yarrr")
new.packages <- list_of_packages[!(list_of_packages %in% installed.packages()[,"Package"
    ])]
#Installing GLMMmisc from Github. GLMMmisc is a package of GLMM diagnostic tools
#and other functions that are useful with GLMMs.
if (!require("GLMMmisc")) devtools::install_github("pcdjohnson/GLMMmisc")
if(length(new.packages)) install.packages(new.packages)
lapply(list_of_packages, library, character.only = TRUE)
##Loading in dataset(s) and cleaning
owl_dat <-read.table(text=getURL("https://raw.githubusercontent.com/jakelrich/MSU_
    WritingProject/master/Owls.txt"), header=T)
#Renaming variable to something shorter
owl_dat$Calls <- owl_dat$SiblingNegotiation
#Saving a copy of the dataset for simulation
sim_dat <- owl_dat
owl_dat$lBroodSize <- log(owl_dat$BroodSize)
###Subsetting Data for multiple simulations
nests <- unique(owl_dat$Nest)
halfnests <- sample(nests, 14)
quarternests <- sample(nests, 7)
owl_dat_14 <- owl_dat[owl_dat$Nest %in% halfnests,]
owl_dat_7 <- owl_dat[owl_dat$Nest %in% quarternests,]
#Releveling to avoid any possible conflicts.
rlev14 <- unique(owl_dat_14$Nest)
rlev7 <- unique(owl_dat_7$Nest)
owl_dat_14$Nest <- factor(owl_dat_14$Nest, levels = rlev14)
owl_dat_7$Nest <- factor(owl_dat_7$Nest, levels = rlev7)
#Centering the reduced datasets to avoid convergence issues.
owl_dat_14$cArrivalTime <- owl_dat_14$ArrivalTime-mean(owl_dat_14$ArrivalTime)
owl_dat_7$cArrivalTime <- owl_dat_7$ArrivalTime-mean(owl_dat_7$ArrivalTime)
####################################
###Exploratory Data Analysis
```

```
qplot(ArrivalTime, Calls, data=owl_dat, facets=~Nest, colour=Nest,
     geom=c("line", "point"),xlab="Arrival Time",ylab="Number of Calls")
###################################
###Writing Simulation Function
##Overview:
#glmm.sim - a wrapper function to simulate data from a given glmm model object
#for a specified number of iterations and return a iter x mod matrix of model
#scores, where iter is the number of iterations and mod is the number of model fitting
#methods used.
##Inputs:
#model - a previously defined glmer object used to simulate data (this will hopefully
#become a user defined function eventually.)
#iter - the number of simulation iterations to run. For each iteration, a new dataset is
    generated.
##Outputs:
#sim_results - a list of estimates for each model.
############
#Notes: simulate.merMod (lme4) will simulate from a glmer object.
############
glmm.sim <- function(model_object, dat, iter, seed, file_name){
  ###Initializing storage matrices
  beta_ests <- matrix(0, iter, 16)
  colnames(beta_ests) <- c("PQL_INT", "PQL_B1", "PQL_B2", "PQL_B3", "LA_INT", "LA_B1", "
      LA_B2", "LA_B3",
                           "AGHQ_INT", "AGHQ_B1", "AGHQ_B2", "AGHQ_B3", "JAGS_INT", "JAGS_B1
                               ", "JAGS_B2", "JAGS_B3")
  sigma_re_ests <- matrix(0, iter, 4)
  colnames(sigma_re_ests) <- c("PQL", "LA", "AGHQ", "JAGS")

  ###Writing JAGS Model Code
  sink("GLMM.txt")
  cat("
      model {
      #1. Diffuse priors for regression parameters.
      beta ~ dmnorm(b0[], B0[,])

      #Diffuse priors for randoms effect hive
      a ~ dmnorm(a0, tau.re*A0[,])
      #Changing the variance in the next line determines
      #the prior for the random effects - currently half-cauchy(2)
      #CHANGE PRIOR BELOW
      num ~ dnorm(0, 0.5)
      denom ~ dnorm(0,1)
      sigma.re <- abs(num/(denom*denom))
      tau.re <- 1 / (sigma.re*sigma.re)

      #2. Likelihood
      for (i in 1:N){
      Y[i] ~ dpois(mu[i])
      log(mu[i]) <- eta[i]
      eta[i] <- inprod(beta[], X[i,]) + a[re[i]]

      #3. Discrepancy Measures
      YNew[i] ~ dpois(mu[i]) #New Data
```

```
    ExpY[i] <- mu[i]
    VarY[i] <- mu[i]
    PRes[i] <- (Y[i] - ExpY[i])/ sqrt(VarY[i])
    PResNew[i] <- (YNew[i] - ExpY[i]) / sqrt(VarY[i])
    D[i] <- pow(PRes[i], 2)
    DNew[i] <- pow(PResNew[i], 2)
    }

    Fit <- sum(D[1:N])
    FitNew <- sum(DNew[1:N])

    }", fill = TRUE)
sink()

inits1 <- function() {
  list(beta = rnorm(K, 0, 1), a = rnorm(Nre, 0, 2), num = runif(1, 0, 25), denom =
      runif(1, 0, 1))
}

params1 <- c('beta', 'a', 'sigma.re', 'Fit', 'FitNew')

###Saving input data as temporary data
tmp_dat <- dat

###Running simulations
for(i in 1:iter){
  timer <- proc.time()
  ###Setting simulation seed
  set.seed(seed+i)

  ###Generating new responses
  #use simr
  tmp_dat$Calls <- doSim(model_object)

  ###Fitting Classical Models
  #Penalized Quasi-likelihood
  owl_glmmPQL <- glmmPQL(Calls ~ lBroodSize + FoodTreatment + ArrivalTime, random = ~
      1|Nest,
                         data = tmp_dat, family = poisson, verbose=FALSE)
  #Laplace Approximation
  owl_glmer_LA <- glmer(Calls ~ lBroodSize + FoodTreatment + ArrivalTime + (1|Nest),
                        data = tmp_dat, family = poisson, nAGQ = 1)
  #Adaptive Gaussian-Hermite Quadrature
  owl_glmer_AGHQ <- glmer(Calls ~ lBroodSize + FoodTreatment + ArrivalTime + (1|Nest),
                          data = tmp_dat, family = poisson, nAGQ = 25)
  #Storing summaries for later use
  PQL <- summary(owl_glmmPQL)
  LA <- summary(owl_glmer_LA)
  AGHQ <- summary(owl_glmer_AGHQ)

  ###Fitting Bayesian Model with JAGS
  X <- model.matrix(model_object)
  K <- ncol(X)
  Nre <- length(unique(tmp_dat$Nest))
```

```
    win.data1 <- list(Y = tmp_dat$Calls, X = X, N = nrow(tmp_dat), re = tmp_dat$Nest,
                      b0 = rep(0, K), B0 = diag(0.0001, K), a0 = rep(0, Nre), A0 = diag(Nre
                          ))
    J0 <- jags.model(file = "GLMM.txt", data = win.data1, inits = inits1,
                     n.chains = 4, n.adapt = 135000)
    mcmc.samples <- coda.samples(J0, params1, n.iter = 50000, thin = 5)
    bayes_stats <- summary(mcmc.samples)$statistics

    ###Storing beta estimates
    beta_ests[i,1:4] <- PQL$coefficients$fixed
    beta_ests[i,5:8] <- LA$coefficients[,1]
    beta_ests[i,9:12] <- AGHQ$coefficients[,1]
    beta_ests[i,13:16] <- bayes_stats[30:33, 1]

    ###Storing random effects variance estimates
    sigma_re_ests[i,1] <- sqrt(c(getVarCov(owl_glmmPQL)))
    sigma_re_ests[i,2] <- sqrt(c(LA$varcor$Nest))
    sigma_re_ests[i,3] <- sqrt(c(AGHQ$varcor$Nest))
    sigma_re_ests[i,4] <- bayes_stats[34,1]

    ###Saving as a list for returning
    sim_results <- list(beta = beta_ests, sigma = sigma_re_ests, model = model_object,
        data = dat, seed = seed)

    ###Saving simulations as RData file every 25 iterations (just in case).
    if(i==1 | i %% 25){
      saveRDS(sim_results, file = file_name)
    }

    ###Print the iteration of simulation, time the iteration took,
    ###and an estimation to ensure seed is changing.
    sim_time <- proc.time() - timer
    diagnos <- list("iter" = i, "time" = sim_time, "sigma_re" = sigma_re_ests[i,1])
    print(diagnos)
  }
  return(sim_results)
}



###Implementing Simulation on Full Data Set
##Getting parameter estimates from the full dataset
full_ests <- glmer(Calls ~ lBroodSize + FoodTreatment + ArrivalTime + (1|Nest),
                   data = owl_dat, family = poisson, nAGQ = 25)
#Passing file name to save simulations as:
file_name <- "D:/Google Drive/GLMMsim_fulldata.RData"
#Remember: glmm.sim <- function(model_object, dat, iter, seed, file_name)
system.time(simulations <- glmm.sim(param_ests, owl_dat, 250, 54177, file_name))
###Saving simulations as RData file (just in case) as an example.
saveRDS(simulations, file = "D:/Google Drive/GLMMsim_fulldata_spare.RData")
```

```
###Implementing Simulation on Half Data Set
##Getting parameter estimates from the half dataset
half_ests <- glmer(Calls ~ lBroodSize + FoodTreatment + ArrivalTime + (1|Nest),
                   data = owl_dat_14, family = poisson, nAGQ = 25)
#Passing file name to save simulations as:
file_name <- "D:/Google Drive/GLMMsim_halfdata.RData"
#Remember: glmm.sim <- function(model_object, dat, iter, seed, file_name)
system.time(simulations <- glmm.sim(half_ests, owl_dat_14, 250, 715, file_name))



###Implementing Simulation on Quarter Data Set
##Getting parameter estimates from the quarter dataset
quart_ests <- glmer(Calls ~ lBroodSize + FoodTreatment + ArrivalTime + (1|Nest),
                   data = owl_dat_7, family = poisson, nAGQ = 25)
#Passing file name to save simulations as:
file_name <- "D:/Google Drive/GLMMsim_quart2data.RData"
#Remember: glmm.sim <- function(model_object, dat, iter, seed, file_name)
system.time(simulations <- glmm.sim(quart_ests, owl_dat_7, 26, 1848, file_name))



###Implementing Simulation on Quarter Data Set with HC(0.5) prior
###NOTE: Change the glmm.txt in glmm.sims to change prior - it's noted.
###not printing the function again to save space
#Passing file name to save simulations as:
file_name <- "D:/Google Drive/GLMMsim_quartdata.RData"
#Remember: glmm.sim <- function(model_object, dat, iter, seed, file_name)
#Using same parameters from quarter dataset
system.time(simulations <- glmm.sim(quart2_ests, owl_dat_7, 250, 1848, file_name))



####################################
###Exploring the Simulation Results
full <- readRDS("D:/Google Drive/GLMMsims_fulldata.RData")
half <- readRDS("D:/Google Drive/GLMMsims_halfdata.RData")
quart <- readRDS("D:/Google Drive/GLMMsims_quartdata.RData")
quart2 <- readRDS("D:/Google Drive/GLMMsims_quart2data.RData")

######Full Data
###Beanplots for Estimated Fixed Effects
fbeta_mat <- full$beta
met <- c(replicate(250, "PQL"), replicate(250, "LA"),replicate(250, "AGHQ"),replicate
    (250, "JAGS"))
fintercept <- data.frame(c(fbeta_mat[,c(1,5,9,13)]), met)
colnames(fintercept) <- c("Intercept", "Method")
fbeta_1 <- data.frame(c(fbeta_mat[,c(2,6,10,14)]), met)
colnames(fbeta_1) <- c("Beta_1", "Method")
fbeta_2 <- data.frame(c(fbeta_mat[,c(3,7,11,15)]), met)
colnames(fbeta_2) <- c("Beta_2", "Method")
fbeta_3 <- data.frame(c(fbeta_mat[,c(4,8,12,16)]), met)
colnames(fbeta_3) <- c("Beta_3", "Method")
par(mfrow=c(2,2))
```

```
pirateplot(formula = Intercept ~ Method,
          data = fintercept,
          pal = "google",
          theme = 3,
          ylab = "",
          main = expression(paste(hat(beta)[0], " estimates for Full Dataset")) )
abline(h = 4.432716, col = "red", lwd = 2)
pirateplot(formula = Beta_1 ~ Method,
          data = fbeta_1,
          pal = "google",
          theme = 3,
          ylab = "",
          main = expression(paste(hat(beta)[1], " estimates for Full Dataset")) )
abline(h = 0.537663, col = "red", lwd = 2)
pirateplot(formula = Beta_2 ~ Method,
          data = fbeta_2,
          pal = "google",
          theme = 3,
          ylab = "",
          main = expression(paste(hat(beta)[2], " estimates for Full Dataset")) )
abline(h = -0.589283, col = "red", lwd = 2)
pirateplot(formula = Beta_3 ~ Method,
          data = fbeta_3,
          pal = "google",
          theme = 3,
          ylab = "",
          main = expression(paste(hat(beta)[3], " estimates for Full Dataset")) )
abline(h = -0.129025, col = "red", lwd = 2)
##Beanplots of variance components
fsigma_mat <- full$sigma
fsigma_re <- data.frame(c(fsigma_mat[,c(1,2,3,4)]), met)
colnames(fsigma_re) <- c("Sigma_re", "Method")
par(mfrow=c(1,1))
pirateplot(formula = Sigma_re ~ Method,
          data = fsigma_re,
          pal = "google",
          theme = 3,
          main = expression(paste(hat(sigma)[re], " estimates for Full Dataset")) )
abline(h = sqrt(c(summary(full_params)$varcor$Nest)), col = "red", lwd = 2)

######Half Data
###Beanplots for Estimated Fixed Effects
half_coefs <- summary(half$model)$coefficients
hbeta_mat <- half$beta
met <- c(replicate(250, "PQL"), replicate(250, "LA"),replicate(250, "AGHQ"),replicate
    (250, "JAGS"))
hintercept <- data.frame(c(hbeta_mat[,c(1,5,9,13)]), met)
colnames(hintercept) <- c("Intercept", "Method")
hbeta_1 <- data.frame(c(hbeta_mat[,c(2,6,10,14)]), met)
colnames(hbeta_1) <- c("Beta_1", "Method")
hbeta_2 <- data.frame(c(hbeta_mat[,c(3,7,11,15)]), met)
colnames(hbeta_2) <- c("Beta_2", "Method")
hbeta_3 <- data.frame(c(hbeta_mat[,c(4,8,12,16)]), met)
colnames(hbeta_3) <- c("Beta_3", "Method")
```

```
par(mfrow=c(2,2))
pirateplot(formula = Intercept ~ Method,
          data = hintercept,
          pal = "google",
          theme = 3,
          ylab = "",
          main = expression(paste(hat(beta)[0], " estimates for Half Dataset")) )
abline(h = half_coefs[1,1], col = "red", lwd = 2)
pirateplot(formula = Beta_1 ~ Method,
          data = hbeta_1,
          pal = "google",
          theme = 3,
          ylab = "",
          main = expression(paste(hat(beta)[1], " estimates for Half Dataset")) )
abline(h = half_coefs[2,1], col = "red", lwd = 2)
pirateplot(formula = Beta_2 ~ Method,
          data = hbeta_2,
          pal = "google",
          theme = 3,
          ylab = "",
          main = expression(paste(hat(beta)[2], " estimates for Half Dataset")) )
abline(h = half_coefs[3,1], col = "red", lwd = 2)
pirateplot(formula = Beta_3 ~ Method,
          data = hbeta_3,
          pal = "google",
          theme = 3,
          ylab = "",
          main = expression(paste(hat(beta)[3], " estimates for Half Dataset")) )
abline(h = half_coefs[4,1], col = "red", lwd = 2)
##Beanplots of variance components
hsigma_mat <- half$sigma
hsigma_re <- data.frame(c(hsigma_mat[,c(1,2,3,4)]), met)
colnames(hsigma_re) <- c("Sigma_re", "Method")
par(mfrow=c(1,1))
pirateplot(formula = Sigma_re ~ Method,
          data = hsigma_re,
          pal = "google",
          theme = 3,
          main = expression(paste(hat(sigma)[re], " estimates for Half Dataset")) )
abline(h = sqrt(c(summary(half$model)$varcor$Nest)), col = "red", lwd = 2)

######Quarter Data
###Beanplots for Estimated Fixed Effects
quart_coefs <- summary(quart$model)$coefficients
qbeta_mat <- quart$beta
met <- c(replicate(250, "PQL"), replicate(250, "LA"),replicate(250, "AGHQ"),replicate
    (250, "JAGS"))
qintercept <- data.frame(c(qbeta_mat[,c(1,5,9,13)]), met)
colnames(qintercept) <- c("Intercept", "Method")
qbeta_1 <- data.frame(c(qbeta_mat[,c(2,6,10,14)]), met)
colnames(qbeta_1) <- c("Beta_1", "Method")
qbeta_2 <- data.frame(c(qbeta_mat[,c(3,7,11,15)]), met)
colnames(qbeta_2) <- c("Beta_2", "Method")
qbeta_3 <- data.frame(c(qbeta_mat[,c(4,8,12,16)]), met)
```

```
colnames(qbeta_3) <- c("Beta_3", "Method")
par(mfrow=c(2,2))
pirateplot(formula = Intercept ~ Method,
          data = qintercept,
          pal = "google",
          theme = 3,
          ylab = "",
          main = expression(paste(hat(beta)[0], " estimates for Quarter Dataset")) )
abline(h = quart_coefs[1,1], col = "red", lwd = 2)
pirateplot(formula = Beta_1 ~ Method,
          data = qbeta_1,
          pal = "google",
          theme = 3,
          ylab = "",
          main = expression(paste(hat(beta)[1], " estimates for Quarter Dataset")) )
abline(h = quart_coefs[2,1], col = "red", lwd = 2)
pirateplot(formula = Beta_2 ~ Method,
          data = qbeta_2,
          pal = "google",
          theme = 3,
          ylab = "",
          main = expression(paste(hat(beta)[2], " estimates for Quarter Dataset")) )
abline(h = quart_coefs[3,1], col = "red", lwd = 2)
pirateplot(formula = Beta_3 ~ Method,
          data = qbeta_3,
          pal = "google",
          theme = 3,
          ylab = "",
          main = expression(paste(hat(beta)[3], " estimates for Quarter Dataset")) )
abline(h = quart_coefs[4,1], col = "red", lwd = 2)

##Beanplots of variance components
qsigma_mat <- quart$sigma
qsigma_re <- data.frame(c(qsigma_mat[,c(1,2,3,4)]), met)
colnames(qsigma_re) <- c("Sigma_re", "Method")
par(mfrow=c(1,1))
pirateplot(formula = Sigma_re ~ Method,
          data = qsigma_re,
          pal = "google",
          theme = 3,
          main = expression(paste(hat(sigma)[re], " estimates for Quarter Dataset")) )
abline(h = sqrt(c(summary(quart$model)$varcor$Nest)), col = "red", lwd = 2)

######Quarter Data (HC(0.5) Prior)
###Beanplots for Estimated Fixed Effects
quart2_coefs <- summary(quart2$model)$coefficients
q2beta_mat <- quart2$beta
met <- c(replicate(250, "PQL"), replicate(250, "LA"),replicate(250, "AGHQ"),replicate
    (250, "JAGS"))
q2intercept <- data.frame(c(q2beta_mat[,c(1,5,9,13)]), met)
colnames(q2intercept) <- c("Intercept", "Method")
q2beta_1 <- data.frame(c(q2beta_mat[,c(2,6,10,14)]), met)
colnames(q2beta_1) <- c("Beta_1", "Method")
q2beta_2 <- data.frame(c(q2beta_mat[,c(3,7,11,15)]), met)
```

```
colnames(q2beta_2) <- c("Beta_2", "Method")
q2beta_3 <- data.frame(c(q2beta_mat[,c(4,8,12,16)]), met)
colnames(q2beta_3) <- c("Beta_3", "Method")
par(mfrow=c(2,2))
pirateplot(formula = Intercept ~ Method,
          data = q2intercept,
          pal = "google",
          theme = 3,
          ylab = "",
          main = expression(paste(hat(beta)[0], " estimates for Quarter Dataset (Half-
              Cauchy(0.5) Prior for ", hat(sigma)[re],")")) )
abline(h = quart2_coefs[1,1], col = "red", lwd = 2)
pirateplot(formula = Beta_1 ~ Method,
          data = q2beta_1,
          pal = "google",
          theme = 3,
          ylab = "",
          main = expression(paste(hat(beta)[1], " estimates for Quarter Dataset (Half-
              Cauchy(0.5) Prior for ", hat(sigma)[re],")")) )
abline(h = quart2_coefs[2,1], col = "red", lwd = 2)
pirateplot(formula = Beta_2 ~ Method,
          data = q2beta_2,
          pal = "google",
          theme = 3,
          ylab = "",
          main = expression(paste(hat(beta)[2], " estimates for Quarter Dataset (Half-
              Cauchy(0.5) Prior for ", hat(sigma)[re],")")) )
abline(h = quart2_coefs[3,1], col = "red", lwd = 2)
pirateplot(formula = Beta_3 ~ Method,
          data = q2beta_3,
          pal = "google",
          theme = 3,
          ylab = "",
          main = expression(paste(hat(beta)[3], " estimates for Quarter Dataset (Half-
              Cauchy(0.5) Prior for ", hat(sigma)[re],")")) )
abline(h = quart2_coefs[4,1], col = "red", lwd = 2)

##Beanplots of variance components
q2sigma_mat <- quart2$sigma
q2sigma_re <- data.frame(c(q2sigma_mat[,c(1,2,3,4)]), met)
colnames(q2sigma_re) <- c("Sigma_re", "Method")
par(mfrow=c(1,1))
pirateplot(formula = Sigma_re ~ Method,
          data = q2sigma_re,
          pal = "google",
          theme = 3,
          main = expression(paste(hat(sigma)[re], " estimates for Quarter Dataset (Half-
              Cauchy(0.5) Prior for ", hat(sigma)[re],")")) )
abline(h = sqrt(c(summary(quart2$model)$varcor$Nest)), col = "red", lwd = 2)
```