UNIVERSITY OF CALIFORNIA
RIVERSIDE

A Bayesian Longitudinal Trend Analysis of Count Data With Gaussian Processes

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Applied Statistics

by

Samantha VanSchalkwyk

September 2020

The Dissertation of Samantha VanSchalkwyk is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

I am grateful to my advisor, without whose help, I would not have been here. I would also like to thank Dr. Manuela Martins-Green, Jane Kim, and Dr. James Borneman for providing the motivating data sets, as well as for their productive discussions for analysis. The data analyzed in this manuscript were obtained from the National Center for Biotechnology Information (NCBI)s Sequence Read Archive (SRA) under the BioProject Accession Number PRJNA623025. This work was supported by 1R21 AI138188-01 to Dr. Manuela Martins-Green.

To my parents for all the support.

ABSTRACT OF THE DISSERTATION

A Bayesian Longitudinal Trend Analysis of Count Data With Gaussian Processes

by

Samantha VanSchalkwyk

Doctor of Philosophy, Graduate Program in Applied Statistics
University of California, Riverside, September 2020
Dr. Daniel R. Jeske, Chairperson

The context of comparing two different groups of subjects that are measured repeatedly over time is considered. Our specific focus is on highly variable count data which have a non-negligible frequency of zeros and have time trends that are difficult to characterize. These challenges are often present when analyzing bacteria or gene expression data sets. Traditional longitudinal data analysis methods, including Generalized Estimating Equations, can be challenged by the features present in these types of data sets. We propose a Bayesian methodology that effectively confronts these challenges. A key feature of the methodology is the use of Gaussian Processes to flexibly model the time trends. Inference procedures based on both sharp and interval null hypotheses are discussed, including for the important hypotheses that test for group differences at individual time points. The proposed methodology is illustrated with next generation sequencing data sets corresponding to two different experimental conditions.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The data sets to which the proposed methodology is aimed to be applied generally measure a genomic feature for two or more groups. Each group has a number of subjects which are measured at multiple discrete time points, which may or may not coincide. Since the same subject is measured more than once, this introduces correlation on the observed counts of the same subjects at different time points. The number of dependent variables is often tens of thousands of this genomic feature, which can be genes or bacteria, and will be referred to as tags throughout this thesis. These next generation sequencing (NGS) data sets, which include RNA-sequencing (RNA-seq) data sets, produce non-negative count data. Often data are considerably overdispersed, meaning the variance is much larger than the mean.

Statistical models for longitudinal data sets such as Generalized Estimating Equations (GEE) (Liang and Zeger, 1986) and generalized linear mixed models which are not designed for NGS data sets often fail to converge. Often the data sets have many zeros, and especially when the ratio of zeros to non-zeros is high, this can become a significant obstacle to model convergence. An additional complexity with NGS data sets is that the time trends are unpredictable, and an appropriate method of addressing these time trends is difficult to choose. Sometimes counts are expected to grow or shrink for a time before achieving an equilibrium due to biological interaction. A flexible model made specifically for these longitudinal NGS data sets would be helpful for identifying differences in group trends over time.

Before NGS data sets, microarray experiments were commonly used. There were some methods developed to detect differential expression in these data sets, such as limma (Smyth, 2005) and GPTwoSample (Stegle et al., 2010). These models are based on Gaussian distributions. However, when NGS data sets started to replace microarray data sets, new models were needed since NGS produces non-negative count data. The limma model added a method called voom (Law et al., 2014) to transform counts logarithmically. However, to avoid taking the logarithm of zero, all zero counts were offset, and this becomes problematic with NGS data sets which have a high number of zeros.

Some models have been developed to analyze NGS data sets using the Negative Binomial (NB) distribution. These include edgeR (Robinson et al., 2009) and DESeq2 (Anders and Huber, 2010; Love et al., 2014), which are built to detect differential expression between groups. However, both of these models are only designed for data sets where subjects are measured at one time point, so they are not suitable for analysis on longitudinal data sets. A model called ZIBR (Chen and Li, 2016) relies on normalization procedures for the NB data and thus has eliminates information about the magnitude of the unnormalized

	Models							
Feature	edgeR	DESeq2	limma/ voom	ZIBR	GPTwo- Sample	DyNB	GEE	Proposed Model
1	✗	✗	✓	✓	✓	✓	✓	✓
2	✓	✓	✗	✗	✗	✓	✓	✓
3	n.a.	n.a.	✓	✓	✓	✗	✓	✓
4	n.a.	n.a.	✗	✗	✓	✗	✓	✓
5	n.a.	n.a.	✗	✗	✓	✓	✗	✓
6	✓	✓	✓	✓	✓	✓	✗	✓

Table 1.1: Models and features. Features 1–6 are: longitudinal, designed for counts, non-synchronous time allowed, inference at each time point, non-parametric time trend, and robust convergence.

data. Another model called DyNB (Äijö et al., 2014), also using the NB distribution, was developed for longitudinal NGS data sets, however this model is not built to handle asynchronous time points, nor does it provide inference at individual time points. See Table 1.1 for a summary of the features of each model discussed. Each of these models will be discussed in more detail in Chapter 3. The main contribution of this thesis is a flexible model fit to sparse, highly variable, longitudinal count data with a robust specification of the time trend.

The rest of this dissertation is organized as follows. Chapter 2 introduces methods used in model building and inference. In Chapter 3, existing models relevant to the NGS application are described. In Chapter 4, the proposed model is introduced. In Chapter 5, we discuss inference procedures implemented after model fitting. In Chapter 6, we describe a simulation study for the proposed model. In Chapter 7, results of model fitting on a wound healing data set are summarized. Chapter 8 provides conclusions and direction for future work. The Appendix includes a comparison study between compositional and raw data, supplementary material for plots, and relevant R code.

Chapter 2

Model building and Inference Methods

In this chapter, methods are introduced that will be used to build the model discussed in Chapter 4. Markov Chain Monte Carlo methods are discussed in detail, which will be used to collect draws from the posterior distribution of interest. Another important component of the proposed model are Gaussian Processes, which can be incorporated into models to flexibly specify a time trend. A goodness-of-fit method for evaluating the fit of the Negative Binomial distribution, designed in particular for NGS data sets, is explained. Some methods to perform model inference for the proposed model are also explored.

2.1 Markov Chain Monte Carlo

As the application of frequentist methods on our data sets of interest resulted in model convergence issues, we explored a Bayesian alternative with a Markov Chain

Monte Carlo (MCMC) algorithm. When computing a posterior distribution is not analytically tractable, MCMC methods can be used to gain draws from a posterior distribution. Here we introduce Markov chains, the MCMC method, the Metropolis-Hastings algorithm, component-wise sampling, and a stopping rule that can be used to determine how long the algorithm needs to be run.

2.1.1 Markov Chains

A Markov chain is a dependent sequence of random variables or random vectors $X_1, X_2, \ldots$ with the property that the conditional distribution of X_{n+1} given $X_1, \ldots, X_n$ depends only on X_n [Flegal, 2016]. If the conditional distribution of X_{n+1} given X_n is the same for all n, the Markov chain is said to have stationary transition probabilities, which is a property of every Markov chain used in an MCMC algorithm. A Markov chain is stationary if its initial distribution is stationary.

Markov chains are categorized by a state space, transition probabilities, and an initial state across the state space. The state space is the set of all possible positions that the random variables can travel to in the Markov chain. The transition probabilities define how each random variable travels within the state space. An initial state across the state space is the set of initial states that components of a Markov chain are given.

2.1.2 MCMC method

Define f to be the probability distribution we would like to explore, which is also called the stationary distribution. In our proposed model, f will be the posterior distribution of interest. If the Markov chain is stationary, then every iterate of the chain X_i

has the same marginal distribution, also called the equilibrium distribution. If chain is not stationary but has a unique equilibrium distribution, which is the case with MCMC chains, then the marginal distribution X_i converges to the equilibrium distribution as $i \longrightarrow \infty$. When independent, identically distributed observations are unavailable, a Markov chain with stationary distribution f can be utilized. Resulting draws of the MCMC can then be used to summarize f with expectations or quantiles, for example.

Certain properties of MCMC chains are used to verify assumptions that will be used later to identify stopping criteria of the chain. A state that the chain returns to with probability 1 is said to be recurrent. If the expected time to return to the state is finite, it is non null. A chain is irreducible if for all i, j pairs there exists $m > 0$ such that $P(X_{m+n} = i | X_n = j) > 0$. A chain is period $r > 1$ if it can only return to its present state X_t at times $t + cr$, for some constant c. A chain is aperiodic if it does not have any period $r > 1$. If the chain is irreducible and aperiodic, then f is unique.

A chain is ergodic if it is irreducible, aperiodic, and all its states are non null and recurrent. Let $x \in \mathtt{X}$, where $\mathtt{X}$ is the support of f, and let A be a measurable set. We will use $P^n(x, A) = P(X_{n+i} \in A | X_i = x)$ to denote what is called an n-step Markov transition kernel. Let $M(x)$ be a nonnegative function and $\|\cdot\|$ be the total variation norm. Polynomial ergodicity of order $m \geq 0$ occurs when $\|P^n(x, \cdot) - f(\cdot)\| \leq M(x) n^{-m}$ (Jones 2004).

Suppose we have draws $X_1, \ldots, X_n$ from a unique equilibrium distribution, and we would like to compute the expectation $\theta = E[g(X_i)]$ with respect to the equilibrium distribution. If $E[g(X_i)] < \infty$, then

$$\hat{\theta} = \frac{1}{n} \sum_{i=1}^{n} g(X_i) \xrightarrow{\text{a.s.}} \theta \tag{2.1}$$

as $n \to \infty$ by the strong law of large numbers (Flegal 2016). If for some $\delta > 0$, g has $2 + \delta$ moments under f, and the Markov chain is polynomially ergodic of order $m > (2+\delta)/\delta$, the Markov chain central limit theorem holds for the approximate sampling distribution of the Monte Carlo error, Σ (Jones 2004). That is, for a chain with p components, there exists a $p \times p$ positive-definite matrix, Σ, such that as $n \to \infty$,

$$\sqrt{n}(\hat{\theta} - \theta) \xrightarrow{\text{d}} N_p(0, \Sigma). \tag{2.2}$$

Monte Carlo error is considered later when assessing stopping criteria for chains.

2.1.3 Evaluating Performance of a MCMC algorithm

A MCMC algorithm should be run until the chain converges on the posterior distribution of interest. The algorithm will do so in the limit, but we cannot determine how many iterations are needed for this convergence to occur before running the chain. However, as only a finite number of iterations can be run, it is necessary to identify when it is reasonable to conclude the chain has converged on the posterior distribution. A stopping rule to determine when a chain has sufficiently converged is discussed in Section 2.1.6.

Another concern while assessing performance of the MCMC algorithm is whether the chain has good mixing. A chain mixes well if it explores the state space without getting stuck at a particular value too long, and efficiently reaches convergence. Successfully mixing chains will have parameters that fluctuate around the values they converge to. See Figure 2.1 for examples of chains with different mixing properties. We will come back to this example in the following section.

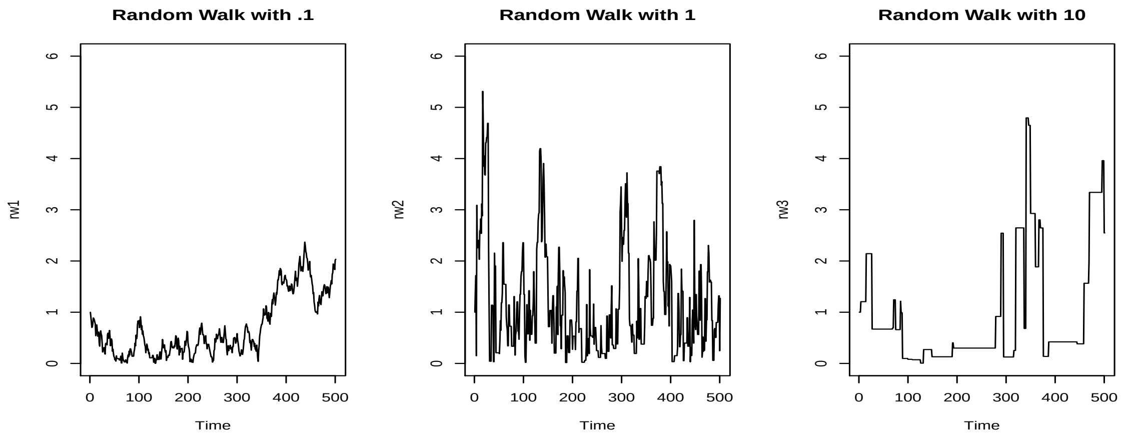


Figure 2.1: The figure on the left shows poor mixing due to the inefficiency of reaching convergence, where the proposal variance is $\sigma = 0.1$. The figure on the right shows poor mixing due to the parameter getting stuck for too long without exploring the state space, where $\sigma = 10$. The figure in the middle shows reasonable mixing, with $\sigma = 1$.

2.1.4 Metropolis-Hastings algorithm

The Metropolis-Hastings (MH) algorithm can be used to generate draws from an MCMC chain. The algorithm requires choice of a proposal distribution, denoted as g, which controls transition probabilities of the chain. To begin, set $X_0 = x_0$, where x_0 are the initial states of the chain. The MH algorithm generates X_{t+1} given $X_t = x_t$ as follows:

1. Sample a candidate value $x^* \sim g(\cdot|x_t)$

2. Compute the MH ratio $R(x_t, X^*)$, where

$$R(x_t, X^*) = \frac{f(x^*)g(x_t|x^*)}{f(x_t)g(x^*|x_t)} \tag{2.3}$$

3. Set $X_{t+1} = \begin{cases} x_t & \text{w.p. } \min\{R(x_t, x^*), 1\} \\ x^* & \text{otherwise} \end{cases}$

The choice of the proposal distribution will control the performance of the chain, such as mixing within the state space.

As a special case of the MH algorithm, if $g(x^*|x_t) = g(x^*)$, this yields an independence chain. In this case, the proposal does not depend on the current state. The resulting MH ratio reduces to

$$R(x_t, x^*) = \frac{f(x^*)g(x_t)}{f(x_t)g(x^*)}. \tag{2.4}$$

Random walk algorithms can also be thought of as a special case of the MH algorithm. In a random walk algorithm, candidate values are generated such that $X^* = X_t + \epsilon$, where $\epsilon \sim h(\cdot)$. The resulting proposal distribution has the property $g(x * |x_t) = h(x^* - x_t)$. If h is a symmetric, zero mean distribution, then the MH ratio simplifies to

$$R(x_t, x^*) = \frac{f(x^*)}{f(x_t)} \tag{2.5}$$

because $h(x^* - x_t) = h(x_t - x^*)$. Common choices of h include the Normal distribution and the Uniform distribution.

Recall that in Section 2.1.3 we considered mixing properties of three chains, with chain values shown in Figure 2.1. These three chains were random walk chains, with $f \sim Exp(1)$ and $g \sim N(0, \sigma^2)$. We know from (2.5) that the MH ratio is

$$R(x_t, x^*) = \frac{\exp(-x^*)}{\exp(-x_t)} = \exp(x_t - x^*) \tag{2.6}$$

for $x^* \geq 0$. Each of the chains begins at an initial value of 1 and is run for 500 iterations. The value chosen for σ, the proposal variance, affects the efficiency of mixing. For $\sigma = 0.1$, mixing is poor due to slow convergence, while for $\sigma = 10$ mixing is poor because the chain gets stuck too often. When $\sigma = 1$ mixing is most appropriate, and the state space is explored well. Therefore, the proposal variance for parameters in the chain is important and must be chosen carefully so that mixing is reasonable and convergence is reached efficiently.

2.1.5 Component-wise sampling

In MCMC chains that have many dimensions, chain mixing can be slow, prolonging convergence. Component-wise sampling is a way to speed mixing by isolating each parameter and updating their values individually. Let $\boldsymbol{\theta}$ be the vector of parameters that will be updated in the MCMC chain. The sampling scheme for component-wise sampling is implemented as follows:

1. Set initial values for all parameters, $\boldsymbol{\theta}^0$.
2. Sample a candidate value θ_1^* for θ_1.
3. At iteration i, compute the MH ratio with θ_1^*, keeping other components as their most recently updated values in the chain.
4. Set $\theta_1^i = \theta_1^*$ if accepted, or $\theta_1^i = \theta_1^{i-1}$ if rejected.
5. Repeat steps 2) through 4) for $\theta_2, \ldots, \theta_N$ (for N-dimensional θ).
6. Repeat steps 2) through 5) for each iteration through the chain.

Each component of the vectors $\boldsymbol{\lambda}$ and $\boldsymbol{\beta}$ uses component-wise sampling.

2.1.6 Stopping rules

Vats et al. (2019) develop a stopping rule for the MCMC chain based on a multivariate effective sample size in order to address correlation across different components, or parameters, of the chain. To introduce this topic, we first consider another correlation present in MCMC chains called autocorrelation, which measures the correlation of the chain values that are i iterations apart, for any i (Kruschke, 2015). The number of iterations i is also referred to as the lag. We use the effective sample size (ESS) to measure the amount of independent information that exists in autocorrelated MCMC chains. The ESS is the sample size that we would have in a chain with no autocorrelation that yielded the same information. In a univariate setting, the ESS is

$$\text{ESS} = N \Big/ \left(1 + 2\sum_{i=1}^{\infty} \text{ACF}(i)\right), \tag{2.7}$$

where ACF(i) is the autocorrelation of the chain at lag i, and N is the length of the chain.

Extending now to a multivariate setting, we estimate the multivariate ESS, which is more appropriate for chains that collect draws for multiple components, as

$$\text{E\hat{S}S} = n\left\{\frac{\det(\Lambda_n)}{\det(\Sigma_n)}\right\}^{1/p}, \tag{2.8}$$

where Λ_n is the sample covariance matrix of the chain, p is the number of components in the chain, and Σ_n is a strongly consistent estimator of Σ (Vats et al., 2019).

The aim of the stopping rule is to terminate the chain when the Monte Carlo standard error Σ is small compared to the variability in the target distribution, which can be evaluated with (2.8). Vats et al. (2019) derive a stopping rule of the form

$$\text{E\hat{S}S} \geq \frac{2^{2/p}\pi}{[p\Gamma(p/2)]^{2/p}} \frac{\chi^2_{1-\alpha,p}}{\epsilon^2} \tag{2.9}$$

for relative precision $\epsilon > 0$ and some choice of confidence level, α.

2.2 Gaussian Processes

Another approach to modeling unpredictable time trends is with Gaussian Processes (GPs). GPs extend Gaussian distributions to infinite dimensions, such that every finite subset follows a multivariate Gaussian distribution. In our case, the GP will be indexed by time, so the GP for a finite subset of time points with K unique time points is defined as

$$\mathbf{F}|\mathbf{m}, \gamma \sim MVN(\mathbf{m}, T(\gamma)), \tag{2.10}$$

where $\mathbf{F}$ is the process, T is the covariance matrix, $\gamma = \{\sigma_1, \sigma_2\}$ are hyperparameters, and $\mathbf{m}$ is the mean of the process. For a squared exponential function, σ_1 and σ_2 are called the length-scale and signal variance hyperparameters, respectively (Rasmussen and Williams, 2006). The squared exponential function defines the (k, k')th element of the matrix T as

$$t_{k,k'}(\sigma_1, \sigma_2) = \sigma_1^2 \exp\left\{ -\frac{1}{2}\frac{(k-k')^2}{\sigma_2^2} \right\}. \tag{2.11}$$

For simplicity in the following section, denote the resulting covariance matrix as $T(\mathbf{k}, \mathbf{k})$ and components as $t_{k,k'}$.

2.2.1 Gaussian Processes for Normally Distributed Data

Let $\mathbf{y} = \{y_1, y_2, \ldots, y_n\}$ be observed function values measured at times given by $\mathbf{k} = \{k_1, k_2, \ldots, k_n\}$. Define the likelihood of $\mathbf{y}$ as

$$f(\mathbf{y}|\boldsymbol{\theta}) = MVN(\boldsymbol{\mu}(\mathbf{k}), T(\mathbf{k}, \mathbf{k})), \tag{2.12}$$

where $\boldsymbol{\theta} = \{\boldsymbol{\beta}, \sigma_1, \sigma_2\}$, σ_1 and σ_2 are used to calculate the covariance function as shown in (2.11), and $\boldsymbol{\beta}$ are parameters that define the mean of $\mathbf{y}$ at locations $\mathbf{k}$. If we believe there is noise associated with the observed values, we could write the covariance function as $T(\mathbf{k},\mathbf{k})+\sigma_3^2\mathbf{I}$. In this case the parameter vector $\boldsymbol{\theta}$ would also include σ_3, the noise variance.

For a location k_* with unknown corresponding value y_* which is independent of the observed data, the joint distribution of the observed data and y_* is

$$p\begin{pmatrix}\mathbf{y}\\ y_*\end{pmatrix} \sim N\left[\begin{pmatrix}\boldsymbol{\mu}(\mathbf{k})\\ \mu(k_*)\end{pmatrix}, \begin{pmatrix}T(\mathbf{k},\mathbf{k}) & T(\mathbf{k},k_*)\\ T(k_*,\mathbf{k}) & t(k_*,k_*)\end{pmatrix}\right], \tag{2.13}$$

where $T(\mathbf{k},k_*)$ is the column vector formed from $t(k_1,k_*),\ldots,t(k_n,k_*)$ and $T(k_*,\mathbf{k})$ is its transpose. The conditional distribution of $Y_*|\mathbf{Y}=\mathbf{y}$ is Gaussian with mean and variance given by

$$m_* = \boldsymbol{\mu}(k_*) + T(k_*,\mathbf{k})T(\mathbf{k},\mathbf{k})^{-1}(\mathbf{y}-\boldsymbol{\mu}(\mathbf{k})) \tag{2.14}$$

$$\sigma_*^2 = T(k_*,k_*) - T(k_*,\mathbf{k})T(\mathbf{k},\mathbf{k})^{-1}T(\mathbf{k},k_*). \tag{2.15}$$

To see an example of how GPs can be used both to model the time trend as well as interpolate between observed time points, consider data from group A that is simulated to take the form $y_{rkA} = -48+44k-12k^2+k^3+\epsilon_A$, where $\epsilon_A \sim N(0,\sigma_{eA}^2)$ and $\sigma_{eA}=20$. Data from group B were simulated as $y_{rkB} = 20+17k-24k^2+3k^3+\epsilon_B$, where $\epsilon_B \sim N(0,\sigma_{eB}^2)$ and $\sigma_{eB}=25$. Both groups have $r=4$ subjects with repeated measurements at time points $k=$ 0, 2, 4, 6, 8, and 10.

Using a squared exponential covariance function, as well as noise variance, we can estimate the parameters of interest, which involves estimating σ_1,σ_2 for each group, as well as σ_{eA}, and σ_{eB} by maximizing the log likelihood. In other words, for group A, the covariance matrix can be modeled with $C_A := C(\sigma_{1A},\sigma_{2A},\sigma_{eA}^2) = T(\mathbf{k},\mathbf{k})+\sigma_{eA}^2\mathbf{I}$, and the covariance matrix for group B can be modeled with $C_B := C(\sigma_{1B},\sigma_{2B},\sigma_{eB}^2) = T(\mathbf{k},\mathbf{k})+\sigma_{eB}^2\mathbf{I}$. Then

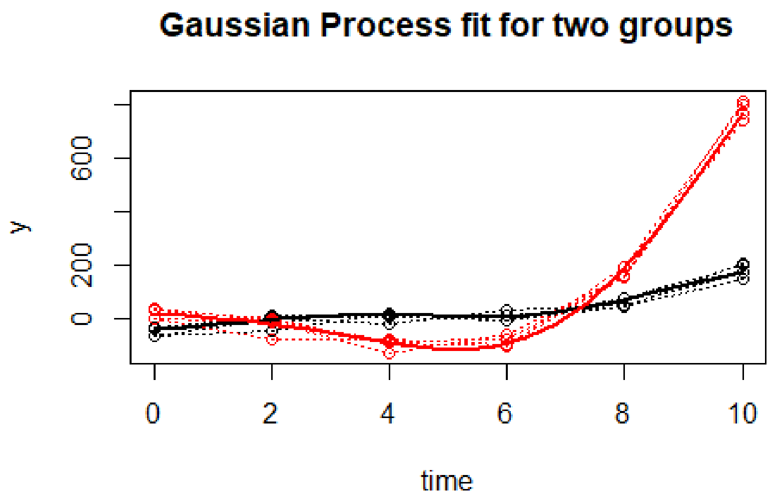


Figure 2.2: Data here were simulated to have subjects from two groups, shown as black or red lines. Dotted lines connect measurements on the same subject, and solid lines show the GP fit, interpolating between observed time points.

minimize the negative of the log likelihood, which is $\sum_{r=1}^{4}[\frac{1}{2}\mathbf{y}_{rA}^{T}C_{A}^{-1}\mathbf{y}_{rA} + \frac{1}{2}\log(\det(C_{A}))]$ for group A and $\sum_{r=1}^{4}[\frac{1}{2}\mathbf{y}_{rB}^{T}C_{B}^{-1}\mathbf{y}_{rB} + \frac{1}{2}\log(\det(C_{B}))]$ for group B. Estimates came out to $\hat{\sigma}_{1A} = 96.58$, $\hat{\sigma}_{2A} = 2.69$, $\hat{\sigma}_{eA}^{2} = 26.84$, $\hat{\sigma}_{1B} = 629.80$, $\hat{\sigma}_{2B} = 3.73$, and $\hat{\sigma}_{eB}^{2} = 59.34$. Those estimates can then be used to estimate the covariance matrix both for the observed points and for any interpolated points over time.

Assuming a zero mean for the observed points, even if the data do not have a zero mean, the GP can still generate a mean that fits with the data. Using (2.14), the GP fit can be found for the original data points as well as for interpolated data between observed time points, where $\boldsymbol{\mu}(k_{*}) = 0$, $\boldsymbol{\mu}(\mathbf{k}) = \mathbf{0}$, and C_A and C_B are used in place of the T covariance matrix for groups A and B, respectively. See Figure 2.2 for a visual of two zero-mean GPs fit to two groups of subjects.

In the previous example, data were simulated to have normally distributed error. However, the application of interest involves NGS data sets, which are non-negative count data, so assuming normally distributed error is not appropriate. The proposed model in following chapters will use the NB distribution to model the data, as there is often overdispersion. Therefore we can consider how to incorporate GPs into a model built for NB data.

2.2.2 Gaussian Processes for Negative Binomial Data

While normally distributed data can use GPs to directly specify the mean, the mean of the NB distribution is strictly non-negative. The DyNB model, which will be discussed further in Chapter 3, uses a GP as the mean of a NB distribution, and rejects all negative MCMC draws of the GP. Another way to handle the inconsistency is by specifying the log of the mean as the GP. That is, let

$$f(y|\theta) = NB(\boldsymbol{\lambda}(\boldsymbol{\theta}), \phi), \tag{2.16}$$

where

$$\log \boldsymbol{\lambda}|\boldsymbol{\theta} \sim MVN(\boldsymbol{\mu}(\mathbf{k}), T(\mathbf{k}, \mathbf{k})) \tag{2.17}$$

and $\boldsymbol{\theta} = \{\boldsymbol{\beta}, \sigma_1, \sigma_2\}$, where these parameters are defined as they were in Section 2.2.1. In this formulation, the GP is still able to flexibly model the time trend, while eliminating the need to reject all negative values from the GP.

2.3 Negative Binomial Goodness-of-fit

The proposed model in Chapter 4 relies on the assumption that the data follow a Negative Binomial (NB) distribution. As our data are overdispersed, the NB distribution is preferred to the Poisson distribution, which requires the same mean and variance. However, we should still make sure that the NB distribution is an appropriate fit for the data to ensure that the model assumption is satisfied.

Mi et al. (2015) propose a goodness-of-fit (GOF) test for NB model adequacy. They use Pearson residuals and compute a p-value based on simulation-based null sampling distributions. A more traditional GOF test based on χ^2 null distributions for residuals is not appropriate with small means, which are often present in NGS data sets. Their method applies to a single tag and is not built for a longitudinal context.

Thee method relies on Pearson residuals, which are computed as $b_r = (y_r - \hat{\mu}_r)/\hat{s}_r$, where y_r is the count for subject r, $\hat{\mu}_r$ is the estimated NB mean and $\hat{s}_r$ is the estimated NB standard deviation of y_r from the model being tested, for $r = 1, \ldots, n$. They simulate a large number of NB data sets of the same size as the observed data, using the estimated mean and standard deviation for simulation. Then they fit the same NB regression model and extract the ordered Pearson residuals from each simulated data set. A Pearson statistic is computed as the sum of squared Pearson residuals. A p-value can be obtained by finding the proportion of simulated samples that produce a Pearson statistic as extreme or more extreme than the observed one.

Another test statistic that Mi et al. (2015) propose is computed as the sum of squared differences of the ordered Pearson residuals from their sampling distribution medians. The sampling distribution medians are available since the simulations provided sampling distributions for each ordered residual. A sum of squared vertical distance GOF p-value uses this statistic instead to assess which simulated statistics are as extreme or more extreme than the observed one. The p-value is found with the following algorithm:

1. Fit a NB regression model to the data $\mathbf{Y}^{(0)} = (Y_1, \ldots, Y_n)^T$. Estimate the dispersion parameter $\hat{\phi}$ and regression coefficients $\hat{\beta}^{(0)}$. Calculate Pearson residuals $\mathbf{b}^{(0)} = (b_1^{(0)}, \ldots, b_n^{(0)})$ and mean vector $\hat{\mu}^{(0)}$.

2. For $h = 1, \ldots, H$:

   - Simulate a random vector $\mathbf{Y}^{(h)}$ from $NB(\hat{\mu}^{(0)}, \hat{\phi})$.
   - Compute and retain Pearson residuals $\mathbf{b}^{(h)}$.

3. Find the median of the Monte Carlo sampling distribution for each ordered residual, denoted by $\hat{b}_{(r)}^{50}$.

4. Compute the sum of squared deviations of ordered residuals from the medians of their sampling distributions, denoted as $d^{(h)} = \sum_{r=1}^{n}(b_{(r)}^{(h)} - \hat{b}_{(r)}^{50})^2$, for the observed data ($h = 0$) and for the simulated samples ($h = 1, \ldots, H$). Compute a Monte Carlo GOF test p-value as

$$p^{MC} = \frac{\sum_{h=1}^{H} I(d^{(h)} \geq d^{(0)}) + 1}{H + 1}. \tag{2.18}$$

The Pearson GOF p-value is computed the same way, but replacing the test statistic with the sum of squared residuals.

2.4 Inference Methods

In this section, inference methods which will be revisited in Chapter 5 are described, including the ROPE procedure and second-generation p-values.

2.4.1 ROPE Procedure

Kruschke (2015) described a procedure built for credible intervals to test if a null value is plausible. The idea is to begin by creating a region of practical equivalence (ROPE) for the null value. The width of the ROPE depends on practical interpretation, but affects the frequency of rejection of the null value.

Once the limits of the ROPE are chosen, it is compared with the credible interval. If the ROPE and the credible interval are disjoint, the null value is rejected. Alternatively, if the ROPE completely contains the credible interval, the null value is said to be accepted for practical purposes. In the case that the ROPE and the credible interval have some overlap but neither of the above situations applies, the decision about the null value is withheld.

2.4.2 Second-Generation p-values

Second-generation p-values (SGPV) were introduced by Blume et al. (2019) as an alternative to more traditional p-values. While the method is designed for use with confidence intervals, credible intervals can be used as well. The SGPV uses the overlap of the interval null hypothesis and the credible region to quantify agreement between the null hypothesis and the data. When the SGPV is close to 1, data are compatible with the null hypothesis. Consider a fixed time point k. Let $I = [\beta_k^l, \beta_k^u]$ be the marginal credibility interval for β_k whose length is given by $|I| = \beta_k^u - \beta_k^l$. Denote the interval null by H_0 and its length by $|H_0|$. The SGPV is

$$p_\delta = \begin{cases} \frac{|I \cap H_0|}{|I|} & \text{when } |I| \leq 2|H_0| \\ \frac{1}{2}\frac{|I \cap H_0|}{|H_0|} & \text{when } |I| > 2|H_0| \end{cases}, \tag{2.19}$$

where $I \cap H_0$ is the overlap of the two intervals.

When the $|I| \leq 2|H_0|$, the interval estimate is considered precise in relation to the null hypothesis. In this situation, the SGPV is the fraction of the interval that is contained within the null hypothesis. Otherwise, it is possible that the interval is wide enough to extend on either side of H_0, which would yield a deceivingly small SGPV without the correction factor which replaces the denominator $|I|$ with $2|H_0|$. In this case where $|I| > 2|H_0|$, the SGPV is bounded by 1/2. As a result, inconclusive data have a SGPV near 1/2, while a SGPV near 1 indicates that the data support H_0.

Chapter 3

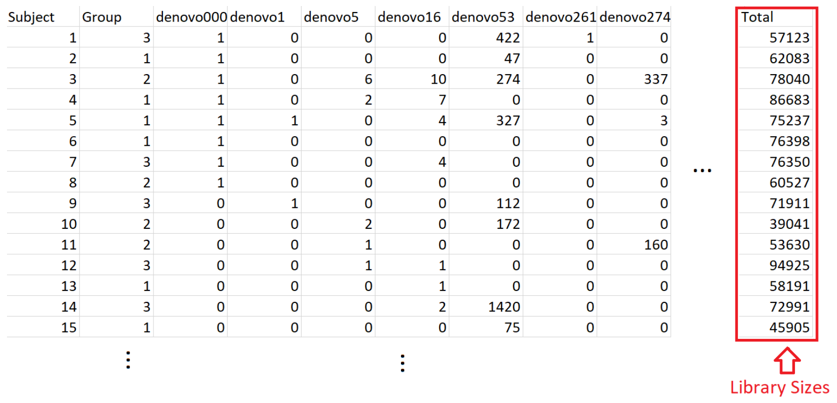
Review of Existing Literature

The existing models to determine if there is differential expression in NGS data sets all have some drawbacks that will be addressed in this thesis. The models considered here include edgeR, DESeq2, limma combined with voom, ZIBR, GPTwoSample, DyNB, and GEE. Certain features of edgeR, GPTwoSample, and DyNB will be incorporated into the proposed model discussed in Chapter 4.

3.1 Non-longitudinal Models

3.1.1 edgeR

The R package edgeR (Robinson et al., 2009) was developed to model non-longitudinal gene expression. It uses a generalized linear model with Negative Binomial (NB) responses, which allows for overdispersion. For a NB random variable Y with mean μ and dispersion parameter ϕ, edgeR uses the parameterization of the NB distribution with



Subject	Group	denovo000	denovo1	denovo5	denovo16	denovo53	denovo261	denovo274	…	Total
1	3	1	0	0	0	422	1	0		57123
2	1	1	0	0	0	47	0	0		62083
3	2	1	0	6	10	274	0	337		78040
4	1	1	0	2	7	0	0	0		86683
5	1	1	1	0	4	327	0	3		75237
6	1	1	0	0	0	0	0	0		76398
7	3	1	0	0	4	0	0	0		76350
8	2	1	0	0	0	0	0	0		60527
9	3	0	1	0	0	112	0	0		71911
10	2	0	0	2	0	172	0	0		39041
11	2	0	0	1	0	0	0	160		53630
12	3	0	0	1	1	0	0	0		94925
13	1	0	0	0	1	0	0	0		58191
14	3	0	0	0	2	1420	0	0		72991
15	1	0	0	0	0	75	0	0		45905
⋮				⋮						

Figure 3.1: Visual of Library Sizes for NGS Data set

probability mass function given by

$$f(y;\mu,\phi) = P(Y = y) = \frac{\Gamma(y+\phi^{-1})}{\Gamma(\phi^{-1})\Gamma(y+1)}\left(\frac{1}{1+\mu\phi}\right)^{\phi^{-1}}\left(\frac{\mu}{\phi^{-1}+\mu}\right)^{y} \tag{3.1}$$

so that $E(Y) = \mu$ and $Var(Y) = \mu(1+\phi\mu)$. Note that the Poisson distribution is a special case when $\lim\limits_{\phi\to 0} f(y;\mu,\phi)$.

Let y_{rt} be the count of the rth subject for tag t. The edgeR model assumes that the counts follow a NB distribution

$$y_{rt} \sim NB(\mu_{rt}, \phi_t), \tag{3.2}$$

where ϕ_t is the dispersion parameter.

Define the library size to be the sum of the counts of a subject across all tags. A visual of the library size can be seen in Figure 3.1, with data provided by James Borneman, PhD (see Acknowledgments). If we let λ_{rt} be the relative abundance of tag t for subject r, and let $m_r = \sum_t y_{rt}$ be the library size of the rth subject, then the mean of y_{rt} is expressed as $\mu_{rt} = m_r\lambda_{rt}$ and $\log\mu_{rt} = \log m_r + \log\lambda_{rt}$.

When there are two groups, let $\log \lambda_{rt} = \beta_{0t} + \beta_{1t} U_r$ where

$$U_r = \begin{cases} 0 & \text{if subject } r \text{ is in group A} \\ 1 & \text{if subject } r \text{ is in group B} \end{cases}. \tag{3.3}$$

Differential expression for tag t is tested with the hypotheses

$$H_0 : \beta_{1t} = 0 \text{ vs. } H_1 : \beta_{1t} \neq 0, \tag{3.4}$$

where H_0 indicates lack of differential expression in tag t and H_1 indicates that differential expression is present in tag t. Testing can also be performed when there are more than two groups by utilizing additional indicator variables in the definition of $\log \lambda_{rt}$.

Consider a test for differential expression between two groups. A Wald test simply divides the maximum likelihood estimate (MLE) $\hat{\beta}_{1t}$ by the standard error, with the asymptotic variance obtained from the Fisher information matrix. The test statistic is then compared to the Normal distribution. An exact test is also implemented by edgeR by creating what is called "pseudodata" and calculating the probability of observing counts as or more extreme than the counts observed.

With edgeR, information is shared across tags to estimate the dispersion parameter, which is thought to stabilize estimation when there are a small number of samples. Without this information sharing approach, model fitting is often prevented by model convergence issues due to instability of the dispersion parameter. The weighted conditional log-likelihood for ϕ_t is a weighted combination of the individual and common log-likelihoods. If $\ell_t(\phi)$ is the single-tag profile log-likelihood for ϕ given the sum of observations for tag t in each group, and $\ell_C(\phi) = \sum_{t=1}^{T} \ell_t(\phi)$ is the common log-likelihood, where T is the total number of tags, the weighted profile log-likelihood (WL) for ϕ_t is

$$WL(\phi_t) = \ell_t(\phi_t) + \alpha \ell_C(\phi_t), \tag{3.5}$$

where α is the weight given to the common log-likelihood.

The value of α determines how much information is being shared across tags to estimate the dispersion parameters. If α is chosen to be large, individual tag-wise contributions to the log-likelihood are outweighed by the common log-likelihood, resulting in a common dispersion parameter for all tags. If $\alpha = 0$, the result is a tag-wise estimate, so no information is being shared across tags. When the estimates of ϕ are so extreme that they prevent model fit convergence, this could become problematic. Otherwise, if α is chosen between these two extremes, the result yields tag-wise estimates that are between the individual and common estimates. In this scenario, the dispersion parameters will be prevented from being extreme values, but will also take information from the data for each tag for estimation.

There are numerous benefits to using the edgeR model. Since the data has so much variability, by using the NB distribution which allows the variance to be much larger than mean, the model more accurately addresses the variability than a Poisson model. The edgeR model allows the user to test thousands of tags for differential expression with just a few lines of code. Conducting all tests at once is beneficial because there are often data sets with a large number of dependent variables being inputted into edgeR, and the output provides ordered p-values of the tags for the specified test. Additionally, a multiple testing adjustment is built in to control the false discovery rate, which is important since there are often so many tags being tested. One significant benefit of edgeR is that the estimation procedure allows information about the dispersion parameter to be shared across tags, and stabilization of the dispersion parameter is crucial to ensuring model convergence. Another benefit, discussed in Appendix A, is that there is no normalization to the library size, so information is not lost. Incorporating the library size into the mean ensures that the counts are considered relative to other counts within their own library, which is important because counts are not comparable across samples. Finally, an R package `edgeR` has been implemented, which makes the model accessible to interested users.

The main drawback to edgeR is that it does not handle repeated measures data. The model is built for cross-sectional data, but data sets with multiple time points are outside the realm of edgeR functionality.

3.1.2 DESeq and DESeq2

Similar to the edgeR model, DESeq (Anders and Huber, 2010) is a model that uses the NB distribution to identify differential expression between groups. They use the same parameterization of the NB distribution as edgeR, but make three assumptions. First, DESeq assumes that the mean is $\mu_{rt} = q_{t,U_r} s_r$, where q_{t,U_r} is proportional to the expected value of the true abundance of tag t under condition U_r, and s_r is called a size factor. The size factor represents the sampling depth of library r. Second, the variance of counts is given by $\sigma_{rt}^2 = \mu_{rt} + s_r^2 v_{t,U_r}$, where $s_r^2 v_{t,U_r}$ is referred to as raw variance. Third, they assume that the per-tag raw variance parameter v_{t,U_r} is a smooth function of q_{t,U_r}, or $v_{t,U_r} = v_U(q_{t,U_r})$.

A few tags may have large counts that have a strong influence on the total read count. As a result, the ratio of total read counts can be a poor estimate of the ratios $E(Y_{rt})/E(Y_{r't})$ of expected counts for tag t in different subjects r and r'. Therefore, DESeq takes the median of the ratios of observed counts to estimate the size factors, which are estimated as

$$\hat{s}_r = \underset{t}{\text{median}} \frac{y_{rt}}{\left(\prod_{r=1}^{n} y_{rt} \right)^{1/n}}. \tag{3.6}$$

The size factors defined here allow counts to be comparable across samples while still modeling counts instead of ratios.

The model is implemented in an R package called `DESeq`, and its successor `DESeq2` was also developed. DESeq2 uses shrinkage estimators to model dispersion and fold change. DESeq2 showed that it had higher overall precision than edgeR with tags that were not differentially expressed. These models, as well as edgeR, are suitable for identifying differential expression in non-longitudinal NGS data sets.

3.2 Longitudinal Models

3.2.1 limma/voom

Linear models for microarray data (limma) [Smyth, 2005] is a package in R for detecting differential expression in data sets collected from microarray experiments. Microarray experiments are becoming widely replaced with NGS technologies. NGS has become more predominantly used for various reasons, since NGS tends to perform better at detecting differential expression [Zhao et al., 2014]. The `voom` function takes counts from NGS data and converts them to log2-counts-per-million (logCPM) so that limma can be applied to NGS data.

Limma assumes a linear model such that $E(d_t) = X\beta_t$, where d_t is a $N \times 1$ vector of logCPM values for tag t, for N total observations of all of the subjects, X is the $N \times p$ design matrix for p parameters, and β_t is a $p \times 1$ vector of coefficients. The variance-covariance matrix for d_t is block diagonal, allowing for independence for different subjects.

Since limma was originally designed for microarray data, which was continuous data, an adjustment was necessary to apply the method to NGS data sets, which is count data. Therefore, voom was created, which transforms the NGS data into logCPM values by calculating

$$r_{rt} = \log_2\left(\frac{y_{rt} + 0.5}{m_r + 1.0} \times 10^6\right), \tag{3.7}$$

where m_r is the library size for subject r, where the counts are offset by 0.5 to avoid taking the logarithm of zero and to reduce variability in low expression genes, and library sizes are offset by 1 so that the ratio is strictly less than 1 but strictly greater than zero [Law et al., 2014]. The logCPM values and derived weights are inputted into limma's linear modeling for differential expression analysis.

With limma, longitudinal data sets where subjects are measured over time can be analyzed to account for the correlation on observations of the same subjects. Since it has been adapted from its original design of microarray analysis to extend to NGS data analysis through the use of voom, it can at least be used for more recent data sets, since microarray technologies are widely becoming outdated. However, voom's data transformation involves adding an offset of 0.5 to all counts to avoid taking the logarithm of zero. There are often so many zeros in NGS data, which makes the normality assumption of the voom values seem questionable. That is, all zeros will map to a very large and negative number through the log transformation, so the problem has just changed form. Finally, counts are normalized by their library sizes, which loses information about the magnitude of the counts. Taking the ratio of the count to its library size removes the model's knowledge of the original scale of the count, which may impact inference in a negative way.

3.2.2 ZIBR

Another method for analyzing longitudinal data was developed in an R package called ZIBR [Chen and Li, 2016]. Zero-inflated Beta Regression (ZIBR) uses random effects to address correlation from repeated measures. It creates compositional data (i.e., proportions) by dividing counts by total sequence count in the sample, which is the library size. ZIBR has a two-part logistic-Beta regression model. There is a logistic component to model the absence or presence of a tag in the samples, and a Beta component that is used to model the abundance of the tag, conditional on it being present.

A clear benefit of ZIBR is that it addresses repeated measures and allows for the corresponding correlation. It also allows covariates affecting the presence or absence of a tag to be different from the covariates affecting abundance, which is a unique feature that directly addresses the fact that there are such a high quantity of zeros in these data sets.

The drawback of using the ZIBR package is that normalizing counts to their relative abundance results in ratios that can lose information (i.e., $\frac{10}{100} \equiv \frac{1000}{10000}$). The standard errors in the models will be affected, which can have an effect on the statistical significance of the tests for differential expression. Refer to Appendix A for an comparative example using ZIBR and edgeR.

3.2.3 GPTwoSample

A model called GPTwoSample was developed by Stegle et al. (2010) to test for differential expression in microarray data. The model has the benefit of incorporating a time trend into the model with Gaussian Processes so that it can be applied to data with repeated measures. The model has the benefit of allowing for inference at each time point. Additionally, GPTwoSample claims robustness to outliers by incorporating a mixture model that allows for noisier observations.

GPTwoSample relies on a comparison of a shared model and two independent models. The shared model assumes that the mean of observations from the two groups being compared were drawn from an identical shared distribution. The independent models allow the model to be fit twice, once to each group. To determine whether differential expression is present, the shared model is compared to the two independent models for each tag with a Bayes factor.

For microarray data, instead of counts, the observations are referred to as expression levels. Let y_{rk} be the expression level for subject r at the kth time point, where $r = 1, \dots, R$ and $k \in \{1, \dots, K\}$. Denote the time points at which the rth subject is measured as $K_r \subseteq \{1, \dots, K\}$. Let $\mathbf{f} \sim N(\mathbf{0}, T(\sigma_1, \sigma_2))$ be a GP prior, where $T(\sigma_1, \sigma_2)$ is the squared exponential covariance function, such that the (k, k')th element of the matrix T is given by

$$t_{k,k'}(\gamma) = \sigma_1^2 \exp\left\{ -\frac{1}{2}\frac{(k-k')^2}{\sigma_2^2} \right\}. \tag{3.8}$$

The posterior distribution over $\mathbf{f}$ for the shared model is given by

$$P(\mathbf{f}|\theta_S, \mathbf{y}) \propto N(\mathbf{f}|\mathbf{0}, T(\sigma_1, \sigma_2)) \prod_{r=1}^{R} \prod_{k \in K_r} N(y_{rk}|f_k, \sigma_r^2), \tag{3.9}$$

where σ_r is the noise level for observations of subject r and $\theta_S = \{\sigma_1, \sigma_2, \{\sigma_r\}_{r=1}^R\}$ denotes the set of all hyperparameters for the shared model. A similar posterior distribution exists for each of the independent models, where θ_S is replaced by θ_I, the GP prior $\mathbf{f}$ is modeled separately for each group, yielding $\mathbf{f}^A$ and $\mathbf{f}^B$, and the product is taken over only replicates from one group for each model.

Inference is first carried out with the Bayes factor to identify differentially expressed tags. Let H_S and H_I denote the competing hypotheses of the shared and independent models, respectively. For the shared model, the probability of the observed data, integrating out θ_S, is

$$P(\mathbf{y}) = \int P(\mathbf{y}|\theta_S)P(\theta_S)d\theta_S, \tag{3.10}$$

where $\mathbf{y}$ are all data points from both groups. For the independent models, the probability of the observed data is

$$P(\mathbf{y}_A) = \int P(\mathbf{y}_A|\theta_I)P(\theta_I)d\theta_I \tag{3.11}$$

$$P(\mathbf{y}_B) = \int P(\mathbf{y}_B|\theta_I)P(\theta_I)d\theta_I, \tag{3.12}$$

where $\mathbf{y}_A$ and $\mathbf{y}_B$ are all data from groups A and B, respectively. The Bayes factor can be computed as

$$\text{BF} = \frac{P(\mathbf{y}_A)P(\mathbf{y}_B)}{P(\mathbf{y})}. \tag{3.13}$$

Stegle et al. (2010) acknowledge that microarray data may contain outliers that would not be well modeled by Gaussian noise. They therefore consider a mixture model of the form

$$P(y_{rk}|f_k, \theta) = \pi_0 N(y_{rk}|f_k, \sigma_r^2) + (1 - \pi_0)N(y_{rk}|f_k, \sigma_{\text{inf}}^2), \tag{3.14}$$

where $\theta = \{\{\sigma_r\}_{r=1}^R\}$, π_0 is the probability of a regular observation and $(1 - \pi_0)$ is the probability of an outlier. The noise variance of the outliers, σ_{inf}^2 is much larger than the noise variance of regular observations. The formation in (3.14) can be used to compute the Bayes factor in (3.13).

Once it is determined that a tag has differential expression, the model performs inference at each time point to identify when tags are differentially expressed. To answer this, they design a switching model between the shared and the independent models. Binary switches z_k are defined at each time point to determine if the shared or the independent models are preferred, where $z_k = 0$ indicates the shared model, and $z_k = 1$ indicates the independent models. Let $P(\mathbf{Z}) = \prod_{k=1}^K \text{Bernoulli}(z_k|0.5)$, which assigns equal prior probability to the shared and independent models. The joint probability of the GPs and model parameters is given by

$$\begin{aligned} P(\mathbf{f}, \mathbf{f}^A, \mathbf{f}^B, \mathbf{Z}|\mathbf{y}, \theta_S, \theta_I) \propto & P(\mathbf{f}|\sigma_1, \sigma_2) P(\mathbf{f}^A|\sigma_1, \sigma_2) P(\mathbf{f}^B|\sigma_1, \sigma_2) P(\mathbf{Z}) \times \\ & \prod_{r \in S_A} \prod_{k \in K_r} \left[N(f_k|y_{r,k}, \sigma_r^2) \right]^{I(z_k=0)} \left[N(f_k^A|y_{r,k}, \sigma_r^2) \right]^{I(z_k=1)} \times \\ & \prod_{r \in S_B} \prod_{k \in K_r} \left[N(f_k|y_{r,k}, \sigma_r^2) \right]^{I(z_k=0)} \left[N(f_k^B|y_{r,k}, \sigma_r^2) \right]^{I(z_k=1)}, \end{aligned} \quad (3.15)$$

where S_A and S_B are the set of subjects from groups A and B, respectively, and $P(\mathbf{f}|\sigma_1, \sigma_2)$, $P(\mathbf{f}^A|\sigma_1, \sigma_2)$, and $P(\mathbf{f}^B|\sigma_1, \sigma_2)$ are the independent GP priors. The joint probability in (3.15) does not make use of the mixture model in (3.14). GPTwoSample performs inference on their model by using a variational approximation, in contrast with the DyNB model discussed in Section 3.2.4 and the proposed model discussed in Chapter 4, which use an MCMC algorithm, described in Section 2.1.

GPTwoSample offers various flexibilities and benefits. Incorporation of GPs into their model allows for flexible time trend modeling. The mixture model to accommodate outliers is useful for improving model fitting for the data. The switching parameters, z_k, are particularly useful for identifying whether differential expression is present at each time point. These features are all desirable, but since GPTwoSample is built for microarray data sets, an alternative is needed for NGS data sets.

3.2.4 DyNB

A model called DyNB (Äijö et al., 2014) was developed to assess differential expression in time course NGS data sets. The model uses a Negative Binomial likelihood and integrates Gaussian Processes into its time trend. A Markov Chain Monte Carlo method is used to gather posterior samples of the parameters of the model, and ultimately Bayes Factors are used to make decisions about differential expression for each tag. DyNB also draws from DESeq (Anders and Huber, 2010) by using size factors on the Gaussian Process samples for variance estimation, which is done to make read counts comparable between different NGS runs.

The GP with hyperparameters $\boldsymbol{\gamma} = \sigma_1, \sigma_2\}$ is defined as

$$\mathbf{F}|\boldsymbol{\xi}, \boldsymbol{\gamma} \sim MVN(\boldsymbol{\xi}, T(\gamma)), \tag{3.16}$$

where $\mathbf{F}$ represents the process, $\boldsymbol{\xi}$ is the mean of the process, and T is the covariance matrix. The GP in (3.16) is used as the prior distribution for the mean of the data. The data are modeled as $y_{rk}|f_k \sim NB(f_k, \phi_k)$ with subjects r and time points $k = 1, \ldots, K$ where ϕ_k is the dispersion parameter and f_k is a realization of the random process in (3.16) at the kth time point. It is assumed that subjects have measurements at synchronous time points. The squared exponential covariance function is used. Prior distributions for $\boldsymbol{\theta} = \{\boldsymbol{\xi}, \gamma, \{\phi_k\}_{k=1}^K\}$ are denoted here as $p(\boldsymbol{\theta})$.

The conditional likelihood of the data can be written as

$$p(\mathbf{y}|\mathbf{f}, \boldsymbol{\theta}) = \prod_{\substack{r \in S \\ k \in \{1, \ldots, K\}}} \frac{\Gamma(y_{rk} + \phi_k^{-1})}{\Gamma(y_{rk} + 1)\Gamma(\phi_k^{-1})} \times \left(\frac{1}{1 + f_k \phi_k}\right)^{\phi_k^{-1}} \left(\frac{f_k}{\phi_k^{-1} + f_k}\right)^{y_{rk}}, \tag{3.17}$$

where the model is fit three times, once with replicates from both groups, then with replicates from each group separately. The marginal density of $\mathbf{y}$ is defined as

$$p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{f}, \boldsymbol{\theta}) p(\mathbf{f}, \boldsymbol{\theta}) d\mathbf{f} d\boldsymbol{\theta}. \tag{3.18}$$

The above equation is estimated using a harmonic mean of evaluations of $p(\mathbf{y}|\mathbf{f}, \boldsymbol{\theta})$ over the posterior draws of $(\mathbf{f}, \boldsymbol{\theta})$ from an MCMC algorithm as proposed by Newton & Raftery (1994).

Finally, a Bayes factor calculated based on the estimate of (3.18) is given by

$$\mathrm{BF} = \frac{p(\mathbf{y}_A) p(\mathbf{y}_B)}{p(\mathbf{y})}, \tag{3.19}$$

where $\mathbf{y}_A$ is the data from group A and $\mathbf{y}_B$ is the data from group B. For the independent models, represented in the numerator of the Bayes factor, different models are fit to the data from each group, while the shared model fits one model to all of the data. The Bayes factor then provides evidence for either the independent or the shared model.

Notice that DyNB uses realizations of a GP to model the mean of negative binomial counts. Clearly, when draws from a GP are negative, they would not function as the mean of a negative binomial distribution, since this must be positive. To account for this, in the code supplied by DyNB, the GP draws are truncated so that no negative counts are used. The main drawback of this model is that it is not capable of analyzing data in which subjects were measured at asynchronous time points, so an alteration is necessary to perform analysis for data sets such as the one described in Chapter 7.

The modeling approach in DyNB is limited by challenges with Bayes factors when subjective information is not available, and also does not readily provide an indication of the direction or trend of group differences over time. For example, when counts in one group trend higher than counts in the other group, Bayes factors do not indicate which group trended higher, nor do they explain when group differences occurred. The proposed methodology discussed in Chapter 4 provides inference for group difference overall and at each time point, and results allow direct interpretation of the direction of group difference.

3.2.5 GEE

Generalized Estimating Equations (GEE) (Liang and Zeger, 1986) can be used to estimate and make inference about group and time trend parameters in data sets with repeated measures. GEE offers certain model flexibilities which make it an appealing choice for NGS data sets. For example, it can be implemented by formulating models for the marginal mean and variance of the data. Consequentially, no likelihood function is required. Additionally, GEE does not require that the correlation matrix of the repeated measures within a subject be precisely specified.

Laird (1989) provides a useful introduction of GEEs. Assume we have observations Y_{rk}, where each observation has a corresponding $p \times 1$ vector of covariates $X_{rk} =$

$\{x_{rk1}, \ldots, x_{rkp}\}'$ so that

$$E(Y_{rk}) = \mu_{rk} = g(X_{rk}^T \beta) \tag{3.20}$$

and

$$g^{-1}(\mu_{rk}) \equiv \ell(\mu_{rk}) = X_{rk}\beta \tag{3.21}$$

for some suitable link function $\ell(\cdot)$. If Y_{rk} are count data, a natural link function to use is the log, so that

$$\log \mu_{rk} = X_{rk}^T \beta \Rightarrow \mu_{rk} = e^{X_{rk}\beta}, \tag{3.22}$$

which ensures that $\mu_{rk} > 0$. For simplicity of notation, denote $E(Y_r) = \mu_r$ where $\ell(\mu_r) = X_r\beta$ denotes the vector $(\ell(\mu_{r1}, \ldots, \mu_{rK})^T$. Let V_r denote the covariance matrix of Y_r.

The generalized estimating equations are given by

$$S(\beta) = \sum_{r=1}^{R} D_r' V_r^{-1}(Y_r - \mu_r) = 0, \tag{3.23}$$

where $D_r = (\delta\mu_r / \delta\beta)$ (Liang and Zeger, 1986). Let $K_r \subseteq \{1, \ldots, K\}$ denote the time points at which the rth subject is measured. The $p \times K_r$ matrix of partial derivatives of the mean with respect β is

$$D_r' = \begin{bmatrix} \frac{x_{r11}}{g'(\mu_{r1})} & \cdots & \frac{x_{rK_r1}}{g'(\mu_{rK_r})} \\ \vdots & & \vdots \\ \frac{x_{r1p}}{g'(\mu_{r1})} & \cdots & \frac{x_{rK_rp}}{g'(\mu_{rK_r})} \end{bmatrix}. \tag{3.24}$$

Let $B_r(\alpha)$ be a "working" correlation matrix which is specified by the parameters α. Denote

$$V_r = \text{var}(Y_r) = \psi A_r^{1/2} W_r^{-1/2} B_r(\alpha) W_r^{-1/2} A_r^{1/2}, \tag{3.25}$$

where $A_r = \{\text{diag} V(\mu_{rk})\}$ and W_r is a $K_r \times K_r$ diagonal matrix of weights, which may be set equal to one for all r and k. The mean model, given by (3.20), and the variance structure in (3.25) are used to proceed with the estimation algorithm, where the variance is assumed to be a known function of the mean. A range of models can be chosen for the working correlation structure, including unstructured. It is estimated with an iterative fitting process using Pearson residuals, given by

$$e_{rk} = \frac{y_{rk} - \mu_{rk}}{\sqrt{v_{rk}/w_{rk}}}. \tag{3.26}$$

For an unstructured working correlation matrix, where

$$Corr(Y_{rk}, Y_{rk'}) = \begin{cases} 1 & k = k' \\ \\ \alpha_{kk'} & k \neq k' \end{cases}, \tag{3.27}$$

the SAS procedure PROC GENMOD estimates the working correlation structure with

$$\hat{\alpha}_{kk'} = \frac{1}{(R-p)\psi} \sum_{r=1}^{R} e_{rk} e_{rk'}. \tag{3.28}$$

The parameter ψ is then estimated as

$$\hat{\psi} = \frac{1}{R-p} \sum_{r=1}^{R} \sum_{k=1}^{K_r} e_{rj}^2. \tag{3.29}$$

Define

$$I_0 = \sum_{r=1}^{R} \frac{\delta \mu_r'}{\delta \beta} \hat{V}_r^{-1} \frac{\delta \mu_r}{\delta \beta}. \tag{3.30}$$

To fit the model, PROC GENMOD uses the following algorithm:

1. Obtain an initial estimate for β using a generalized linear model that assumes independence.

2. Compute the working correlation structure B based on standardized residuals, the current β, and the assumed structure of B.

3. Estimate the covariance matrix, given by $V_r = \psi A_r^{1/2} W_r^{-1/2} \hat{B}_r(\alpha) W_r^{-1/2} A_r^{1/2}$.

4. Update β such that $\beta_{i+1} = \beta_i + [I_0]^{-1}\Bigg[\sum_{r=1}^{R} \frac{\delta \mu_r'}{\delta \beta} V_r^{-1}(Y_r - \mu_r)\Bigg]$.

5. Iterate steps 2-4 until convergence.

A model-based estimator of $Cov(\hat{\beta})$ is given by

$$\Sigma_m(\hat{\beta}) = I_0^{-1}, \tag{3.31}$$

and is a consistent estimator of the covariance matrix of β if the mean model and the working correlation matrix are correctly specified. Alternatively, the estimator

$$\Sigma_e = I_0^{-1} I_1 I_0^{-1} \tag{3.32}$$

is called the empirical estimator of the covariance matrix of $\hat{\beta}$, where

$$I_1 = \sum_{r=1}^{R} \frac{\delta \mu_r'}{\delta \beta} \hat{V}_r^{-1} \text{Cov}(Y_r) V_r^{-1} \frac{\delta \mu_r}{\delta \beta}. \tag{3.33}$$

The empirical estimator is a consistent estimator of the covariance matrix of β even if the working correlation matrix is misspecified, meaning $\text{Cov}(Y_r) \neq V_r$. To compute Σ_e, estimates are used for β and ψ, and $\text{Cov}(Y_r)$ is estimated as $(Y_r - \mu_r)(Y_r - \mu_r)'$.

It is of interest to perform inference with the GEE model to test for differential expression between groups over time, as well as to identify overall group or time differences. In PROC GENMOD, generalized score statistics are used, which are motivated by Boos (1992) and Rotnitzky and Jewell (1990). For a $j \times p$ contrast matrix L', we would like to test $L'\beta = \mathbf{0}$. Let $\tilde{\beta}$ denote the regression parameters that result from solving the estimating equations under the restricted model $L'\beta = \mathbf{0}$, and $S(\tilde{\beta})$ denote the values of the estimating equations at $\tilde{\beta}$ (see (3.23)). Then the generalized score statistic is

$$T = S(\tilde{\beta})'\Sigma_m L(L'\Sigma_e L)^{-1} L'\Sigma_m S(\tilde{\beta}). \tag{3.34}$$

Then p-values can be found based on an asymptotic Wald test for T based on a chi-squared distribution with j degrees of freedom.

For the NGS application, the mean is expressed as $\mu_{rkt} = m_{rk}\lambda_{rkt}$ for subject r at time point k for tag t, where $m_{r,k} = \sum_t y_{rkt}$ is the library size, which is specific to each subject and time point. Group and time trend structure can be specified by incorporating indicator variables into a parameterization of $\log \lambda_{rkt}$. An asymptotic Wald test can be used to determine if the time trends are the same for each group.

When implemented on data sets such as our NGS application described in Section 7, model convergence is an issue for sparse tags. The root cause of model convergence issues is that unlike edgeR, the GEE fitting algorithm does not share data across tags. Issues related to the difficulty of GEE convergence were discussed in a symposium talk given by the author at the University of California, Riverside in September, 2018 [VanSchalkwyk, 2018].

Chapter 4

Proposed Model

It is of interest to detect differences between groups A and B measured on subjects across time. To address the unpredictable time trend, we incorporate a GP prior into the mean of the NB distribution. We use the probability mass function for the NB distribution given by (3.1).

We consider two different model formulations. One allows the dispersion parameter of the NB distribution to be common within each group. The other formulation is a richer model, which allows for a unique estimate of the dispersion parameter for each group at each time point. While this allows for more flexibility in specifying the dispersion parameter, this extension may not often be necessary.

4.1 Model Formulation: Common Dispersion

For simplicity in notation, we describe the model for one arbitrary tag. Let y_{rk} be the observation for subject r at the kth time point. Let $r \in \{S_A, S_B\}$, where S_A denotes subjects from group A and S_B denotes subjects from group B, and $K_r \subseteq \{1, \dots, K\}$ denote the time points at which the rth subject is measured. The posterior distribution of all the model parameters can be written as

$$P(\boldsymbol{\lambda}, \boldsymbol{\beta}, \sigma_1, \sigma_2, \phi_A, \phi_B | \mathbf{y}) \propto \prod_{r \in S_A} \prod_{k \in K_r} NB(y_{rk}; m_{rk}\lambda_k, \phi_A) \prod_{r \in S_B} \prod_{k \in K_r} NB(y_{rk}; m_{rk}\lambda_k\beta_k, \phi_B)$$
$$P(\boldsymbol{\beta})P(\boldsymbol{\lambda}|\sigma_1, \sigma_2)P(\sigma_1)P(\sigma_2)P(\phi_A)P(\phi_B), \tag{4.1}$$

where m_{rk} is the library size, which is specific to each subject and time point, and is defined as the sum of counts observed at time k for subject r across all of the tags. Here, $P(\cdot)$ denotes a prior for the indicated parameter. Note that the group means are $m_{rk}\lambda_k$ and $m_{rk}\lambda_k\beta_k$ for groups A and B, respectively. Going forward, we define $\boldsymbol{\theta} = (\boldsymbol{\lambda}, \boldsymbol{\beta}, \sigma_1, \sigma_2, \phi_A, \phi_B)$. Priors on these parameters are defined in Section 4.1.1.

The mean specification in (4.1) is designed to mimic the edgeR mean structure by incorporating the library size. Counts are therefore modeled relative to their measurement totals, which is important because counts are only interpretable relative to this library size. Including the library size in the mean is a preferred option to taking ratios of counts and library sizes, as it retains the magnitude of the data. See Appendix A for more details.

Notice from (4.1) that the difference between group means for groups A and B is quantified by the $K \times 1$ vector $\boldsymbol{\beta}$. These $\boldsymbol{\beta}$ parameters will therefore be used to perform inference on group differences over time in Chapter 5. For now, note that when $\beta_t < 1$, the mean for group B is lower than the mean for group A at time t. When $\beta_t > 1$, the mean for group B is higher than the mean for group A at time t. When $\beta_t = 1$, means are equivalent for both groups at time t.

4.1.1 Prior Distributions

The $K \times 1$ vector for $\boldsymbol{\lambda}$ has a multivariate log-normal prior distribution, equivalent to a GP prior for $\log \boldsymbol{\lambda}$, that is, $\log \boldsymbol{\lambda} \sim \text{MVN}(\mathbf{0}, T(\sigma_1, \sigma_2))$ where the elements in the covariance matrix $T(\sigma_1, \sigma_2)$ are given by (3.8). Prior information is not always available, so we use non-informative priors for the other parameters of the model. A multivariate log-normal prior distribution is used for $\boldsymbol{\beta}$, so that $\log \boldsymbol{\beta}$ follows a zero-mean multivariate normal distribution with a large variance. Then σ_1, σ_2, ϕ_A and ϕ_B have non-informative gamma priors with appropriate and specific hyperparameters. Non-informative uniform priors on ϕ_A and ϕ_B were also implemented, and the resulting posterior draws were very similar in both cases.

4.1.2 Fitting Algorithm

To sample from the posterior distribution in (4.1), we use a Markov Chain Monte Carlo (MCMC) algorithm. More specifically, we use the Metropolis Hastings (MH) algorithm, which requires the choice of proposal distributions. For proposal distributions, each component of $\boldsymbol{\theta}$ is generated via a random walk. For candidate values $\boldsymbol{\theta}^*$ and draws from iteration $i-1$ of the chain denoted as $\boldsymbol{\theta}^{i-1}$, the Metropolis Hastings (MH) ratio is

$$R(\boldsymbol{\theta}^{i-1}, \boldsymbol{\theta}^*) = \frac{f(\boldsymbol{\theta}^*|y)}{f(\boldsymbol{\theta}^{i-1}|y)}. \tag{4.2}$$

The variance of the proposal distribution for each parameter should be used as a lever to achieve proper mixing of the MCMC chain. Proposal distributions for components of $\boldsymbol{\beta}$ and $\boldsymbol{\lambda}$ are log-normal, equivalent to normal distributions for components of $\log \boldsymbol{\beta}$ and

$\log \boldsymbol{\lambda}$. The complete set of proposal distributions are:

$$\log \beta_k^* \sim \mathrm{N}(\log \beta_k^{i-1}, \sigma_{\beta_k}^2), k = 1, 2, \ldots, K \tag{4.3}$$

$$\log \lambda_k^* \sim \mathrm{N}(\log \lambda_k^{i-1}, \sigma_{\lambda_k}^2), k = 1, 2, \ldots, K \tag{4.4}$$

$$\sigma_1^* \sim \mathrm{Unif}(\sigma_1^{i-1} - a, \sigma_1^{i-1} + a) \tag{4.5}$$

$$\sigma_2^* \sim \mathrm{Unif}(\sigma_2^{i-1} - b, \sigma_2^{i-1} + b) \tag{4.6}$$

$$\phi_A^* \sim \mathrm{Unif}(\phi_A^{i-1} - c, \phi_A^{i-1} + c) \tag{4.7}$$

$$\phi_B^* \sim \mathrm{Unif}(\phi_B^{i-1} - d, \phi_B^{i-1} + d), \tag{4.8}$$

where $\sigma_{\beta_k}, \sigma_{\lambda_k}$, a, b, c, and d are constants for $k = 1, \ldots, K$, and are chosen in the tuning phase of the algorithm where the goal is to achieve a target acceptance rate. Since σ_1, σ_2, ϕ_A and ϕ_B all take strictly positive values, candidate values for these parameters which are non-positive are rejected with probability 1.

Component-wise sampling can be helpful for high-dimensional chains when mixing is slow or satisfactory acceptance rates are difficult to achieve. We implement component-wise sampling for each of the parameters of the model with the sampling scheme described in Section 2.1.5.

There are some challenges introduced with the MCMC method, but each can be attended to. For example, running the chain is time consuming, and must be done for each tag. However, running tags in parallel on a computing cluster can alleviate computing time concerns. Also, the chain should be run until a minimum effective sample size (which depends on the number of parameters) is reached (Vats et al., 2019). To avoid gathering too few draws and needing to restart, after a specified number of iterations are run, the code checks every 100 iterations whether this condition is met as stopping-point criteria. Another concern is that parameters for the proposal distribution must be chosen. After fitting the model to a few tags using a manual process, reasonable ranges of proposal distribution parameters will become apparent.

The proposed model is designed to accommodate missing data. That is, it is not assumed that there are measurements at each time point for all subjects, nor is it assumed that there are an equal number of subjects in each group. These are both limitations that exist in the DyNB method (Äijö et al., 2014). However, it is necessary to have data from subjects from both groups at each time point in order to detect difference in groups. The proposed model is formulated to perform inference on the $\boldsymbol{\beta}$ parameters, which quantify differences between groups at each time point, so having data at each time point from both groups is necessary.

4.2 Model Formulation: Time-Varying Dispersion

We would now like to formulate a model which will allow the dispersion to vary over time within each group. Utilizing much of the same notation from Section 4.1, the posterior distribution can be written as

$$\begin{aligned} P(\boldsymbol{\lambda}, \boldsymbol{\beta}, \sigma_1, \sigma_2, \boldsymbol{\phi}_A, \boldsymbol{\phi}_B | \mathbf{y}) \propto \prod_{r \in S_A} \prod_{k \in K_r} NB(y_{rk}; m_{rk}\lambda_k, \phi_{kA}) \prod_{r \in S_B} \prod_{k \in K_r} NB(y_{rk}; m_{rk}\lambda_k\beta_k, \phi_{kB}) \\ P(\boldsymbol{\beta}) P(\boldsymbol{\lambda}|\sigma_1, \sigma_2) P(\sigma_1) P(\sigma_2) P(\boldsymbol{\phi}_A) P(\boldsymbol{\phi}_B), \qquad (4.9) \end{aligned}$$

where now $\boldsymbol{\phi}_A$ and $\boldsymbol{\phi}_B$ are K-dimensional vectors of dispersion parameters.

4.2.1 Prior Distributions

For this richer model, we use the same prior distributions as Section 4.1.1 for all of the common parameters of the two model formulations (i.e., $\boldsymbol{\beta}, \boldsymbol{\lambda}, \sigma_1, \sigma_2$). Prior distributions for each component of the vectors $\boldsymbol{\phi_A}$ and $\boldsymbol{\phi}_B$ are specified as gamma priors, as were the common ϕ_A and ϕ_B parameters.

4.2.2 Fitting Algorithm

Using the random walk chain is done in a very similar way as in Section 4.1.2, but now the proposal distributions must be specified for each component of the vectors of dispersion parameters. Proposal distributions for the dispersion parameters are given by

$$\phi^*_{Ak} \sim \text{Unif}(\phi^{i-1}_{Ak} - c_k, \phi^{i-1}_{Ak} + c_k) \tag{4.10}$$

$$\phi^*_{Bk} \sim \text{Unif}(\phi^{i-1}_{Bk} - d_k, \phi^{i-1}_{Bk} + d_k), \tag{4.11}$$

where c_k and d_k are constants for $k = 1, \ldots, K$. Proposed values of ϕ_{Ak} or ϕ_{Bk} less than zero are rejected with probability 1.

Chapter 5

Model Inference

Testing for differences between the two groups involves inference on the parameters β_k, for $k = 1, \ldots, K$. Bayes factors are commonly used for Bayesian hypothesis testing, which is exemplified by the DyNB method in Section 3.2.4. Bayes factors are designed for model selection, and we could have tried to implement the Bayes factor for the posterior draws of our model. However, Bayes factors are not advised for use with noninformative prior distributions, as prior distributions significantly impact the outcome. For our purposes, we do not assume subjective prior knowledge about group differences in the data, as our analysis is exploratory. Therefore, we prefer an alternative approach to testing for group differences.

While using a Bayesian methodology to construct the model in Chapter 4, we would like to capitalize on the information available from the posterior distribution. Stern (2005) discusses a preference for using the posterior distribution to perform Bayesian analysis, compared with model selection methods such as the Bayes factor. However, a hypothesis testing framework is not widely accepted by Bayesian statisticians, so it is not obvious how to perform inference in the context of interest. Our inference methods, discussed in this chapter, integrate information from the posterior draws of $\boldsymbol{\beta}$ to conduct hypothesis tests to identify group differences. It is our belief that such a test is unavoidable for making decisions for each tag in the data set, since we would like to provide an answer to whether or not group differences are present.

Following McShane et al. (2019) and references therein, which discuss the fact that sharp null hypotheses of zero effect are often implausible, we develop inference methods that can accommodate either sharp or interval null hypotheses. We begin by constructing a hyper-rectangular joint $100(1-\alpha)\%$ credible region of the K-dimensional $\boldsymbol{\beta}$ vector. Group difference is inferred when at least one dimension of the credible region is disjoint from the null hypothesis of choice. We then integrate this hyper-rectangular joint credible region into a method which calculates what we refer to as a 'pseudo Bayesian p-value', which provides a global significance test for group difference for each tag. When the global significance test rejects, we proceed to individual significance tests of group difference at each time point.

A sharp null, $H_0 : \boldsymbol{\beta} = \mathbf{1}$, would consist of testing whether each dimension of the hyper-rectangular joint credible region includes 1. An interval null hypothesis is constructed to be an interval of indifference, or in other words a region of practical equivalence (ROPE) (Kruschke, 2015), which was introduced in Section 2.4.1. The amount we allow each β_k to differ from 1 and be considered inconsequential dictates the width of the ROPE. The ROPE hypothesis testing procedure is designed for a one-dimensional interval null. We extend the ROPE procedure to higher dimensions by rejecting the global significance test of group difference if any dimension of the credible region is disjoint from its corresponding dimension specified by the ROPE.

5.1 Hyper-Rectangular Joint Credible Region

We would like to be able to contain our $\boldsymbol{\beta}$ parameters in a credible region which has the appropriate joint $100(1-\alpha)\%$ posterior coverage probability. Building marginal $100(1-\alpha)\%$ credible intervals for each dimension would lead to joint coverage under $100(1-\alpha)\%$, while if we considered Bonferroni-corrected marginal credible intervals this would lead to joint coverage over $100(1-\alpha)\%$. We utilize an approach motivated by Robertson et al. (2019), which searches $\alpha^* \in [\alpha/K, \alpha]$ to find marginal intervals with size $100(1-\alpha^*)\%$ that collectively create a joint $100(1-\alpha)\%$ hyper-rectangular credible region.

The hyper-rectangular joint credible region is found by constructing $100(1-\alpha^*)\%$ credible intervals for each of the K dimensions and counting how many of the MCMC draws of these $\boldsymbol{\beta}$ parameters are within the resulting credible region. Once this coverage is satisfactorily close to $100(1-\alpha)\%$, we stop and use that α^* as a calibrating value that provides approximate joint coverage of $100(1-\alpha)\%$.

5.2 Global Test

We propose the following scheme to calculate what we refer to as 'pseudo Bayesian p-values' to identify whether group difference exists for each tag:

1. Given:

    - A set of $\alpha \in (0, 1]$ from which to search for the pseudo Bayesian p-value
    - A null hypothesis set corresponding to a sharp null or a K-dimensional interval null

2. Begin with the smallest α, say α^*

3. Construct a $100(1 - \alpha^*)\%$ hyper-rectangular joint credible interval

4. Identify whether any dimension of the credibility interval is disjoint with the corresponding dimension in the null hypothesis set

    - If so, stop, α^* is the pseudo Bayesian p-value
    - Otherwise, use the next largest α and repeat steps 3 and 4
    - If you reach $\alpha = 1$, set the pseudo p-value equal to 1

Due to the discreteness of the search algorithm, the actual pseudo p-value will be less than or equal to the stopping value α^*.

The scheme above results in pseudo Bayesian p-values which are smallest for a point null hypothesis and increase as the null hypothesis set expands in size, as long as the point null is a subset of the null interval. Since it should be easier to reject a point null than an interval null that contains that point, this agrees with intuition. Referring to Figure 5.1, which illustrates a two parameter setting, notice that smaller values of α result in larger joint credible regions. When $\alpha = 0.001$ or 0.005, neither rectangular joint credible interval is disjoint from either null hypothesis set shown. When $\alpha = 0.01$, the sharp null at

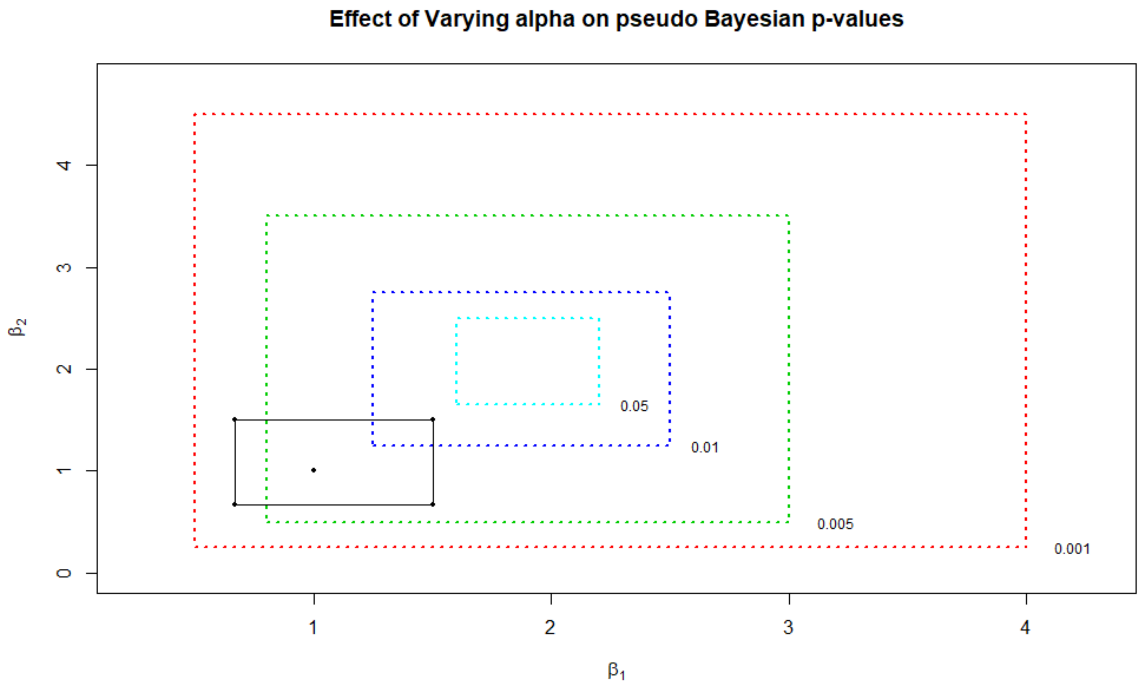


Figure 5.1: The effect of varying α while computing the pseudo Bayesian p-value in a two parameter setting is shown here. Two null hypothesis sets are shown; one for $(\beta_1, \beta_2) =$ (1,1), shown as a point at (1,1), and one for $(\beta_1, \beta_2) \in (1/1.5, 1.5) \times (1/1.5, 1.5)$, shown as a solid line black box. Rectangular joint credible intervals are plotted in red, green, dark blue, and light blue, with $\alpha = 0.05, 0.01, 0.005$, and 0.001 respectively, where the value of α is shown at the bottom right corner of each rectangular joint credible interval.

$(\beta_1, \beta_2) = (1, 1)$ is rejected, but the interval null, where $(\beta_1, \beta_2) \in (1/1.5, 1.5) \times (1/1.5, 1.5)$, is not. Thus, for the null hypothesis $H_0 : (\beta_1, \beta_2) = (1, 1)$, the pseudo Bayesian p-value is between 0.005 and 0.01. For $\alpha = 0.05$, the interval null hypothesis set is disjoint from the null interval, so the pseudo Bayesian p-value for the null hypothesis $H_0 : (\beta_1, \beta_2) \in (1/1.5, 1.5) \times (1/1.5, 1.5)$ is between 0.01 and 0.05.

Tags which yield global test pseudo Bayesian p-values less than a chosen threshold, such as $\alpha = 0.05$, could be considered to have statistically significant group differences over time. Moreover, if interval nulls are used, it could be said the groups have statistically significant practical differences over time. Note that values of α that are used to search for the pseudo Bayesian p-value should be carefully chosen to account for specific significance levels of interest.

5.2.1 Sharp Null Hypotheses

To use a sharp null for a global test, we would test

$$H_0 : \boldsymbol{\beta} = \mathbf{1}^K \text{ vs. } H_1 : \text{ at least one } \beta_k \neq 1, \tag{5.1}$$

where $\mathbf{1}^K$ is a K-dimensional vector of 1's.

5.2.2 Interval Null Hypotheses

To use an interval null for a global test, we would test

$$H_0 : \boldsymbol{\beta} \in (1/c, c)^K \text{ vs. } H_1 : \text{ at least one } \beta_k \notin (1/c, c) \tag{5.2}$$

for some constant $c > 0$, where $(1/c, c)^K$ is a K-dimensional hypercube with endpoints $(1/c, c)$ in each dimension. Note that an interval null is synonymous with the term ROPE. Using the interval null hypothesis in conjunction with the hyper-rectangular joint credible region would extend the ROPE testing procedure to multiple dimensions.

If a change of means by a factor of 1.5, for example, is considered negligible, this can be used to define the interval null. Specifically, that would mean the hypothesis test is

$$H_0 : \boldsymbol{\beta} \in \left(\frac{1}{1.5}, 1.5\right)^K \text{ vs. } H_1 : \text{ at least one } \beta_k \notin \left(\frac{1}{1.5}, 1.5\right). \tag{5.3}$$

Since the posterior draws of $\boldsymbol{\beta}$ are on the log scale, this is equivalent to testing

$$H_0 : \log \boldsymbol{\beta} \in (-.405, .405)^K \text{ vs. } H_1 : \text{ at least one } \log \beta_k \notin (-.405, .405). \tag{5.4}$$

5.3 Individual Time Point Tests

When the global test from Section 5.2 rejects, we proceed to individual tests for group differences at each time point. The calibrated hyper-rectangular joint credible region has built-in multiple testing protection, so the same region is appropriate for tests at each time point. One option is to use the ROPE testing procedure described by Kruschke (2015) for each dimension, leading to a decision at each time point. The results from such a test, assuming the hyper-rectangular joint credible intervals are still used, would be equivalent to observing which dimensions of the hyper-rectangular joint credible region were disjoint from the null space in the global test. Another option is to use the SGPV method (Blume et al., 2019) introduced in Section 2.4.2.

5.4 Tests of Interaction

A useful result would be to gain insight on how groups differ over time. There are two tests of interaction that can be conducted using the output of MCMC draws. One test is for whether the difference in group intensities is the same over time, and another tests for whether ratios of the means are the same over time.

5.4.1 Differences

It may be of interest to identify whether the difference in means is constant over time. Since the data sets of interest are compositional, that is, counts are only interpretable relative to their respective library sizes, we can conduct this test using intensities which eliminate the need for the library size to be included in the test. Define $c_k = \lambda_k(1 - \beta_k)$, which is a difference of the means of groups A and B, eliminating the library size term. Testing for consistent difference in group intensities over time is a test of

$$H_0 : c_1 = c_2 = \cdots = c_K \text{ vs. } H_1 : \text{ not } H_0. \tag{5.5}$$

The test can be carried out by forming the $K-1$ contrasts of the c_k's and testing if they are jointly zero. MCMC draws of $\log \boldsymbol{\lambda}$ and $\log \boldsymbol{\beta}$ can be used to construct a hyper-rectangular joint credible region for the contrasts, and a pseudo Bayesian p-value can be found as described in Section 5.2.

5.4.2 Ratios

Testing for consistent ratios of group means over time is a test of

$$H_0 : \beta_1 = \beta_2 = \cdots = \beta_K \text{ vs. } H_1 : \text{ not } H_0. \tag{5.6}$$

The test can be carried out by forming the $K-1$ contrasts of the β_k's and testing if they are jointly zero. MCMC draws of $\log \boldsymbol{\beta}$ can be used to construct a hyper-rectangular joint credible region for the contrasts, and a pseudo Bayesian p-value can be found as described in Section 5.2.

5.4.3 Comment

Though they are both tests of interaction, the tests for differences of group intensities and ratios of group means may produce different results. Chapter 6 will introduce a setting where data are simulated to have parallel differences over time (refer to Figure 6.1). Data sets with this mean structure should not reject the test for differences of group intensities, since the difference stays constant over time between the two groups. Using

GEE, a related test could be performed on these data sets using a log link function and testing for group by time interaction. In the same mean structure scenario, the test for difference of ratios should reject, since the parallel group lines change over time, impacting the ratios. The related test with GEE would use the identity link function to test for group by time interaction, and is implemented in the following chapter.

The discrepancy between expected results for this particular mean structure is useful for understanding the difference in the two types of interaction tests. When the log link function was used to test for interaction with GEE, 99.9% of the simulated data sets indicated group by time interaction. While this result makes sense for testing difference of intensities, the traditional understanding of an interaction test as identifying non-parallel lines would make this an unusual result. Using ratios to test the difference instead yields more comprehensible results in such a context.

Chapter 6

Simulation Study

While a Bayesian machinery is used to develop this model, we are interested in exploring frequentist properties of the model by using repeated sampling so that we can evaluate performance of the hypothesis test for group differences. To explore the statistical power of the proposed model, various data generation settings were considered (see Table 6.1). For each data set scenario, 1000 data sets were simulated. To generate the data, the means from Table 6.1 were used as the mean of the NB distribution, specified as $\mu_{rk} = m_{rk}\lambda_k$ for time points $k = 1, \ldots, K$. The size parameter for the NB distribution was specified as $\frac{1}{\phi_C}$, for $C = \{A, B\}$. The library sizes were set to 10,000 for each subject at all time points, which is within the range of library sizes from the data set in Chapter 7. All simulated data sets have 4 time points at days 0, 2, 4, and 8. Some data sets have 10 subjects per group, while others have 40 subjects per group. Dispersion parameters for each group, ϕ_A and ϕ_B, were set to either .1 or 2, which was motivated by preliminary analysis of the data in Chapter 7 using edgeR.

			Means Group A				Means Group B			
Scenario	Group Sizes	ϕ_A, ϕ_B	D0	D2	D4	D8	D0	D2	D4	D8
1	10	.1, .1	400	600	800	1000	400	600	800	1000
2	10	.1, .1	400	600	600	600	400	600	1200	1800
3	10	.1, .1	400	600	1200	1800	400	600	600	600
4	10	.1, .1	400	600	800	1000	1200	1400	1600	1800
5	10	.1, .1	400	400	400	400	400	600	800	1000
6	10	.1, .1	200	200	200	200	200	1000	2000	3000
7	10	2, 2	400	600	800	1000	400	600	800	1000
8	10	2, 2	400	600	600	600	400	600	1200	1800
9	10	2, 2	400	600	1200	1800	400	600	600	600
10	10	2, 2	400	600	800	1000	1200	1400	1600	1800
11	10	2, 2	400	400	400	400	400	600	800	1000
12	10	2, 2	200	200	200	200	200	1000	2000	3000
13	40	.1, .1	400	600	800	1000	400	600	800	1000
14	40	.1, .1	400	600	600	600	400	600	1200	1800
15	40	.1, .1	400	600	1200	1800	400	600	600	600
16	40	.1, .1	400	600	800	1000	1200	1400	1600	1800
17	40	.1, .1	400	400	400	400	400	600	800	1000
18	40	.1, .1	200	200	200	200	200	1000	2000	3000
19	40	2, 2	400	600	800	1000	400	600	800	1000
20	40	2, 2	400	600	600	600	400	600	1200	1800
21	40	2, 2	400	600	1200	1800	400	600	600	600
22	40	2, 2	400	600	800	1000	1200	1400	1600	1800
23	40	2, 2	400	400	400	400	400	600	800	1000
24	40	2, 2	200	200	200	200	200	1000	2000	3000

Table 6.1: Settings for Simulated Data. D0, D2, D4 and D8 represent day 0, day 2, day 4, and day 8, respectively.

Six distinct mean trends were chosen to assess the power of tests from Section 5. A visual of the mean structures is shown in Figure 6.1. Notice that mean structure 1 has no difference between groups at any time point. Scenarios with this structure should allow us to measure the Type I Error for different combinations of sample size and variation, imposed by ϕ_A and ϕ_B. Mean structures 2 and 3 reverse the direction of group difference, allowing us to assess power for values of β that are greater than 1, as well as less than 1. Mean structure 4 has a parallel difference between groups over time. We expect more power

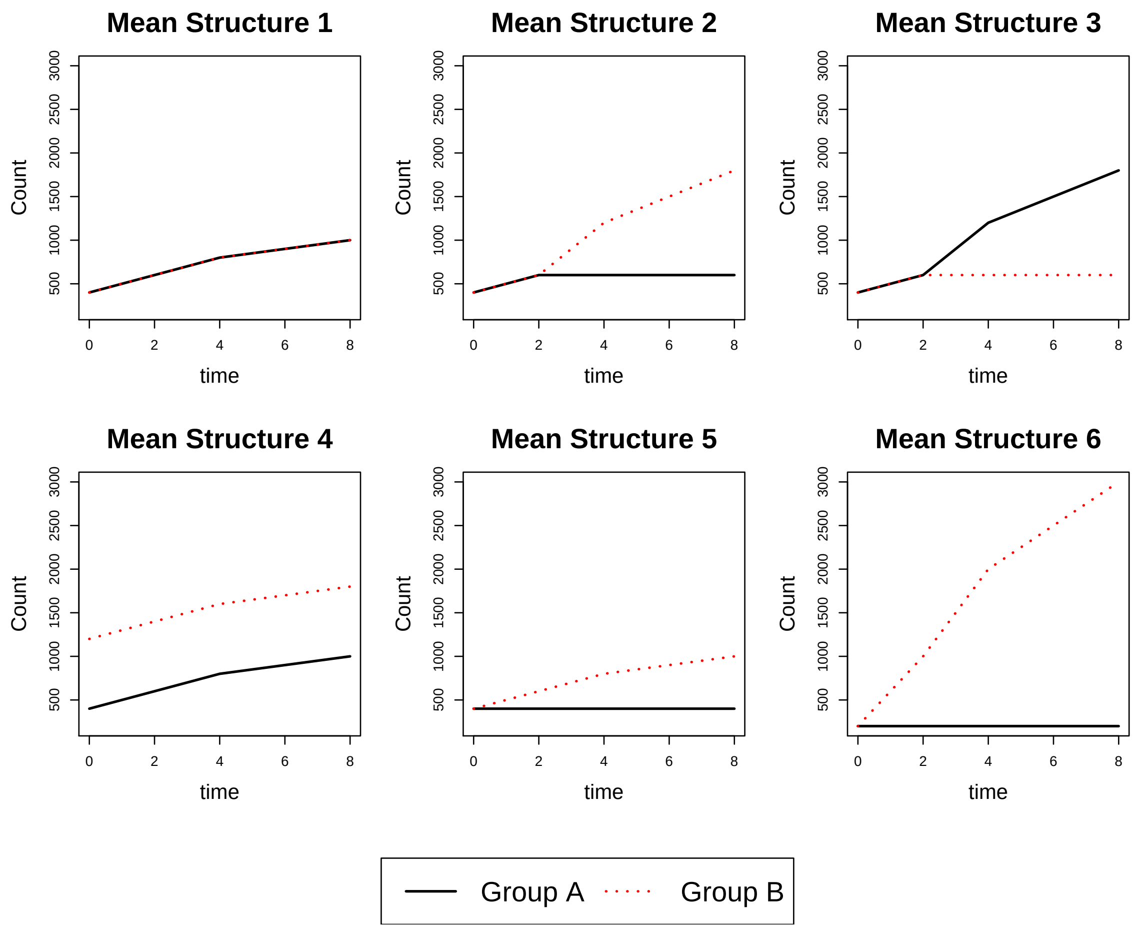


Figure 6.1: Mean structures from simulation settings described in Table 6.1. Mean structures $i = 1, \ldots, 6$ correspond to scenarios $i + 6c$, for $c = 0, 1, 2, 3$.

to detect changes in mean structure 6 due to the larger magnitude of group differences, especially in later time points.

6.1 GEE Results

Power for the global test of group differences from running GEE on the simulated data is shown in Table 6.2. GEE was fit with an identity link function and a non-parametric time trend using dummy variables for each day. Results in the table were based on asymptotic Wald tests, as described in Section 3.2.5. The column for Group × Time is a test for equality of difference of means over time. With few exceptions, the GEE model was able to converge successfully for the simulated data sets within each scenario. Overall, GEE shows strong power to detect significant group and time differences at a 5% significance level, as well as difference in means. Power is weaker for simulated scenarios 7-12 and 19-24 because ϕ_A and ϕ_B were higher in those scenarios, introducing higher levels of variation. The fact that scenarios 13-24 show higher power than scenarios 1-12 is due to the increased sample size for those scenarios. Scenarios 6, 12, 18, and 24, which have higher effect sizes compared to the other scenarios, generally yield stronger power. Some false rejection rates are high for scenarios 1-12. Due to the smaller sample sizes per group in these scenarios, the asymptotic properties of the GEE parameter estimators most likely have not quite taken effect.

The following code can be used to run GEE in R:

```
data1.1 <- read.csv("Data1.1.csv", fileEncoding="UTF-8-BOM")
library(geepack)
gee1.1 <- geeglm(Count ~ factor(Group) + factor(Day) +
                    factor(Group):factor(Day) + offset(log(Total)),
                    id = Mouse, data = data1.1,
                    family = poisson, corstr = "unstructured")
summary(gee1.1)
anova(gee1.1)
```

The above code produces the same model estimates as the SAS GENMOD procedure. Running the `anova()` function on the `geeglm` object gives results for a Wald statistic test of group difference, time difference, and an interaction test between group and time.

			% Rejected		
Scenario	n	ϕ_A, ϕ_B	Group	Time	Group × Time
1	10	.1, .1	**10.0**	100	**12.9**
2	10	.1, .1	98.8	100	100
3	10	.1, .1	98.8	100	100
4	10	.1, .1	99.9	99.9	**18.1**
5	10	.1, .1	99.8	95.8	100
6	10	.1, .1	96.3	100	100
7	10	2, 2	**8.9**	50.5	**11.6**
8	10	2, 2	44.3	51.1	34.4
9	10	2, 2	46.4	48.8	36.8
10	10	2, 2	63.4	28.0	**17.4**
11	10	2, 2	46.8	24.0	28.0
12	10	2, 2	97.7	70.9	95.0
13	40	.1, .1	**6.2**	100	**5.9**
14	40	.1, .1	100	100	100
15	40	.1, .1	99.9	100	100
16	40	.1, .1	100	100	**7.1**
17	40	.1, .1	100	100	100
18	40	.1, .1	97.5	100	100
19	40	2, 2	**5.0**	97.5	**6.0**
20	40	2, 2	93.3	98.9	85.1
21	40	2, 2	93.9	98.9	82.7
22	40	2, 2	99.4	57.7	**6.7**
23	40	2, 2	94.1	52.1	56.6
24	40	2, 2	99.9	100	100

Table 6.2: Simulation Results for GEE. The group size is denoted here as n. Numbers in bold indicate that the test should not be rejected in that case.

6.2 Proposed Methodology Results

Tables 6.3 and 6.4 show the rejection rates for the global test of group difference for each data set scenario, as well as the rejection rates at each time point. Effect size is determined by the values of β_k, $k \in \{1, 2, 3, 4\}$, and the further $\log \beta_k$ is from 1, the higher the effect size. For each scenario, H_0 is rejected if the hyper-rectangular joint 95% credible interval is disjoint from the null region. Although pseudo-Bayesian p-values were not computed, they would have produced identical results for these tables by using 0.05

as a rejection boundary. From these tables, it is evident that with a larger effect size, the proposed model more easily rejects H_0. Larger sample size and lower values of ϕ_A and ϕ_B also improve true rejection rates. There are no instances of high rates of false rejection.

		ϕ_A,	Proportion Rejected									
Scen.	n	ϕ_B	Global	Time 0	β_1	Time 2	β_2	Time 4	β_3	Time 8	β_4
1	10	.1, .1	0.027	0.010	1	0.003	1	0.006	1	0.008	1
2	10	.1, .1	1.000	0.012	1	0.026	1	0.987	2	1.000	3
3	10	.1, .1	1.000	0.012	1	0.008	1	0.952	1/2	1.000	1/3
4	10	.1, .1	1.000	1.000	3	1.000	7/3	0.991	2	0.816	9/5
5	10	.1, .1	1.000	0.007	1	0.725	3/2	0.994	2	0.999	5/2
6	10	.1, .1	1.000	0.014	1	1.000	5	1.000	10	1.000	15
7	10	2, 2	0.022	0.008	1	0.006	1	0.007	1	0.003	1
8	10	2, 2	0.166	0.010	1	0.004	1	0.075	2	0.094	3
9	10	2, 2	0.142	0.009	1	0.007	1	0.026	1/2	0.105	1/3
10	10	2, 2	0.255	0.090	3	0.144	7/3	0.067	2	0.012	9/5
11	10	2, 2	0.145	0.006	1	0.033	3/2	0.073	2	0.048	5/2
12	10	2, 2	0.990	0.010	1	0.645	5	0.898	10	0.868	15
13	40	.1, .1	0.030	0.011	1	0.005	1	0.007	1	0.009	1
14	40	.1, .1	1.000	0.008	1	0.026	1	1.000	2	1.000	3
15	40	.1, .1	1.000	0.016	1	0.015	1	1.000	1/2	1.000	1/3
16	40	.1, .1	1.000	1.000	3	1.000	7/3	1.000	2	1.000	9/5
17	40	.1, .1	1.000	0.011	1	0.999	3/2	1.000	2	1.000	5/2
18	40	.1, .1	1.000	0.009	1	1.000	5	1.000	10	1.000	15
19	40	2, 2	0.029	0.010	1	0.010	1	0.005	1	0.005	1
20	40	2, 2	0.839	0.021	1	0.014	1	0.427	2	0.727	3
21	40	2, 2	0.799	0.014	1	0.003	1	0.217	1/2	0.738	1/3
22	40	2, 2	0.961	0.806	3	0.752	7/3	0.475	2	0.142	9/5
23	40	2, 2	0.748	0.013	1	0.170	3/2	0.446	2	0.486	5/2
24	40	2, 2	1.000	0.011	1	1.000	5	1.000	10	1.000	15

Table 6.3: Simulation Results for Proposed Model. The group size is denoted here as n. The value of β is given for each time point. Rejections are anticipated unless $\beta = 1$. The table shows results for the global test of $H_0 : \boldsymbol{\beta} = 1$ by computing a hyper-rectangular joint 95% credible interval, and marginal proportions of rejections at each time point.

Table 6.5 shows the bias and root mean squared error (RMSE) of the estimates of $\beta_1, \beta_2, \beta_3, \beta_4$, as well as the average run time that data sets in each scenario took to achieve the minimum effective sample size. As expected, the bias and RMSE tend to be

		ϕ_A,	Proportion Rejected								
Scen.	n	ϕ_B	Global	Time 0	β_1	Time 2	β_2	Time 4	β_3	Time 8	β_4
1	10	.1, .1	0.000	0.000	1	0.000	1	0.000	1	0.000	1
2	10	.1, .1	0.966	0.000	1	0.000	1	0.227	2	0.961	3
3	10	.1, .1	0.970	0.000	1	0.000	1	0.121	1/2	0.967	1/3
4	10	.1, .1	0.992	0.979	3	0.824	7/3	0.311	2	0.031	9/5
5	10	.1, .1	0.795	0.000	1	0.004	3/2	0.324	2	0.727	5/2
6	10	.1, .1	1.000	0.000	1	1.000	5	1.000	10	1.000	15
7	10	2, 2	0.004	0.003	1	0.000	1	0.001	1	0.000	1
8	10	2, 2	0.040	0.004	1	0.000	1	0.011	2	0.027	3
9	10	2, 2	0.044	0.002	1	0.001	1	0.009	1/2	0.032	1/3
10	10	2, 2	0.066	0.017	3	0.033	7/3	0.018	2	0.002	9/5
11	10	2, 2	0.029	0.002	1	0.002	3/2	0.014	2	0.011	5/2
12	10	2, 2	0.926	0.004	1	0.358	5	0.741	10	0.700	15
13	40	.1, .1	0.000	0.000	1	0.000	1	0.000	1	0.000	1
14	40	.1, .1	1.000	0.000	1	0.000	1	0.871	2	1.000	3
15	40	.1, .1	1.000	0.000	1	0.000	1	0.826	1/2	1.000	1/3
16	40	.1, .1	1.000	1.000	3	1.000	7/3	0.947	2	0.356	9/5
17	40	.1, .1	1.000	0.000	1	0.004	3/2	0.972	2	1.000	5/2
18	40	.1, .1	1.000	0.000	1	1.000	5	1.000	10	1.000	15
19	40	2, 2	0.000	0.000	1	0.000	1	0.000	1	0.000	1
20	40	2, 2	0.275	0.000	1	0.000	1	0.058	2	0.237	3
21	40	2, 2	0.320	0.000	1	0.000	1	0.021	1/2	0.303	1/3
22	40	2, 2	0.447	0.300	3	0.187	7/3	0.075	2	0.016	9/5
23	40	2, 2	0.145	0.000	1	0.003	3/2	0.060	2	0.088	5/2
24	40	2, 2	1.000	0.000	1	0.967	5	1.000	10	1.000	15

Table 6.4: Simulation Results for Proposed Model. The group size is denoted here as n. The value of β is given for each time point. Rejections are anticipated unless $\beta = 1$. The table shows results for the global test of $H_0 : \boldsymbol{\beta} \in (1/1.5, 1.5)$ using a hyper-rectangular joint 95% credible interval, and marginal proportions of rejections at each time point.

highest for the scenarios 6-12 with lower sample sizes per group and higher variation and lowest for scenarios 13-18 with higher sample sizes per group and lower variation. The average run time per data set across scenarios ranged from 34.1 minutes to 101.4 minutes, and altogether the model run on these 24,000 simulated data sets takes about 24 days on a high performance computing cluster using 100 cores, which allowed for 50 data sets to be analyzed simultaneously.

	bias				RMSE				Avg Run
Scenario	$\log\beta_1$	$\log\beta_2$	$\log\beta_3$	$\log\beta_4$	$\log\beta_1$	$\log\beta_2$	$\log\beta_3$	$\log\beta_4$	Time
1	0.039	-0.022	-0.029	0.017	0.145	0.122	0.137	0.146	54.3 min
2	0.058	-0.086	0.016	0.014	0.153	0.152	0.132	0.149	58.2 min
3	-0.002	0.047	-0.061	0.018	0.144	0.137	0.159	0.149	35.2 min
4	0.036	-0.021	-0.022	0.027	0.145	0.119	0.136	0.145	34.1 min
5	0.022	0.000	-0.009	0.017	0.131	0.119	0.125	0.135	39.6 min
6	0.018	-0.011	-0.020	0.020	0.143	0.119	0.126	0.139	44.6 min
7	0.194	-0.024	-0.054	0.045	0.627	0.545	0.560	0.629	69.1 min
8	0.193	-0.075	-0.025	0.047	0.645	0.560	0.575	0.619	99.3 min
9	0.173	0.078	-0.123	0.081	0.630	0.572	0.595	0.609	93.5 min
10	0.202	0.010	-0.013	0.037	0.614	0.586	0.578	0.617	73.2 min
11	0.093	-0.004	-0.036	0.109	0.604	0.541	0.553	0.633	79.7 min
12	0.095	0.016	0.019	0.100	0.624	0.541	0.558	0.638	60.8 min
13	0.008	-0.012	-0.006	0.008	0.070	0.062	0.066	0.073	62.0 min
14	0.019	-0.038	0.018	-0.003	0.072	0.084	0.074	0.072	61.6 min
15	-0.009	0.026	-0.019	0.003	0.071	0.078	0.078	0.073	60.7 min
16	0.013	-0.012	-0.005	0.007	0.072	0.063	0.064	0.069	61.1 min
17	0.009	-0.008	-0.008	0.004	0.067	0.060	0.063	0.068	80.7 min
18	0.009	-0.004	-0.004	0.008	0.069	0.060	0.062	0.069	85.4 min
19	0.096	-0.036	-0.055	0.040	0.321	0.272	0.274	0.309	97.4 min
20	0.116	-0.086	-0.010	0.026	0.327	0.283	0.285	0.313	96.6 min
21	0.066	0.056	-0.103	0.030	0.317	0.279	0.324	0.327	83.7 min
22	0.068	-0.036	-0.063	0.034	0.321	0.262	0.283	0.318	90.6 min
23	0.028	-0.022	-0.026	0.041	0.297	0.268	0.275	0.305	100.8 min
24	0.039	-0.007	-0.032	0.012	0.294	0.268	0.271	0.319	101.4 min

Table 6.5: Bias and RMSE for Proposed Methodology

Chapter 7

Example Data set

The motivating data set for this proposed methodology contains 37 mice with chronic wounds and 40 mice with non-chronic wounds. Referring to Section 4.1, we will treat mice with non-chronic wounds as group A, and those with chronic wounds as group B. Mice with chronic wounds are considered chronic due to application of oxidative stress (Kim et al., 2020), and are coded as 1 in Figure 7.2. Most mice are measured at days 0, 1, 2, 3, 5, 10, 15, and 20, with some missing data. Bacteria counts for each sample over time are gathered. The 100 bacteria with the highest summed count over all mice and time points were considered for analysis, since, as can be seen in the data, nonzero counts quickly become sparse as their summed count decreases. Empirical mean plots for the twelve most abundant bacteria are shown in Figure 7.1. The data set is shown in abbreviated form in Figure 7.2, where rows of the data correspond to the samples taken on the subjects, and columns labeled "Bacteria1" to "Bacteria100" are the tags. The library size for this example data set is shown as the right-most column.

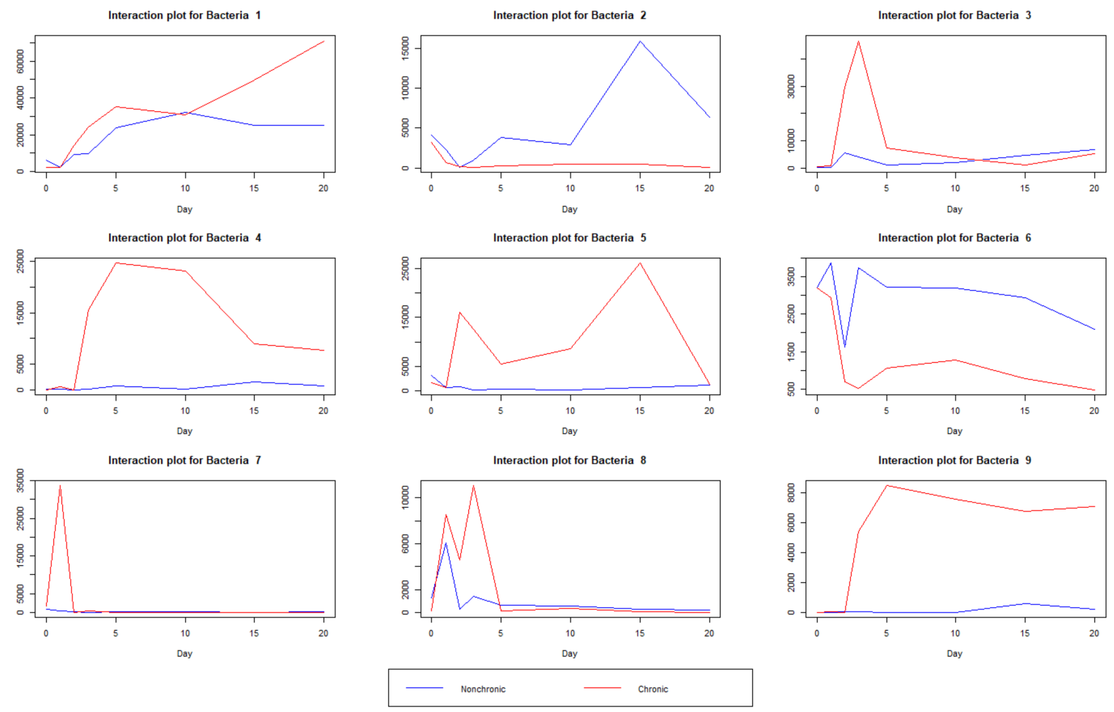
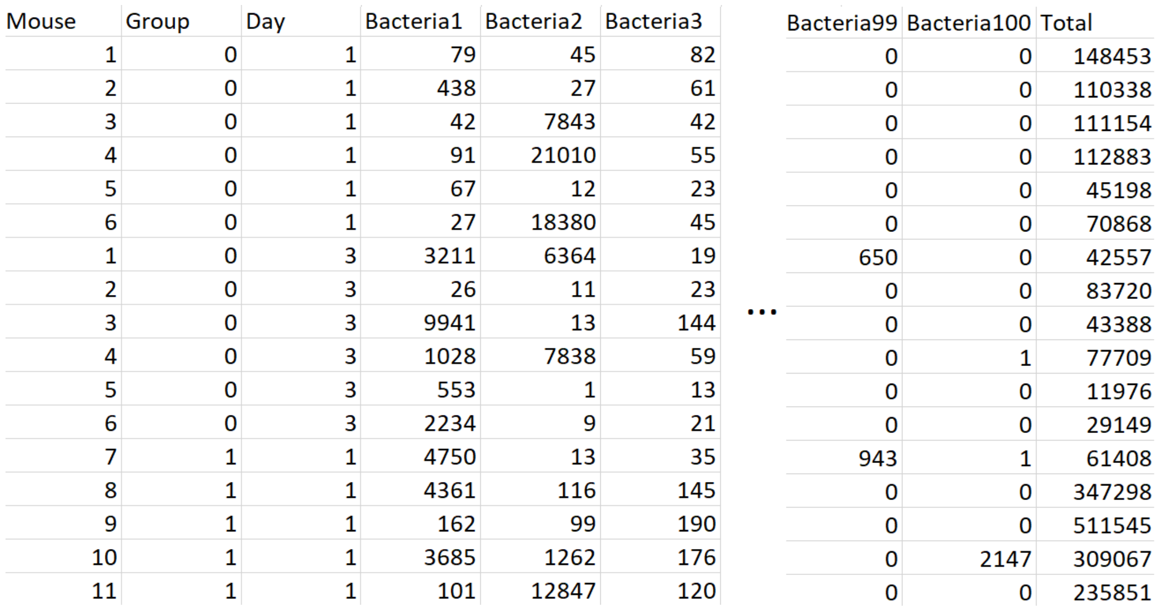


Figure 7.1: Empirical mean plots



Mouse	Group	Day	Bacteria1	Bacteria2	Bacteria3		Bacteria99	Bacteria100	Total
1	0	1	79	45	82		0	0	148453
2	0	1	438	27	61		0	0	110338
3	0	1	42	7843	42		0	0	111154
4	0	1	91	21010	55		0	0	112883
5	0	1	67	12	23		0	0	45198
6	0	1	27	18380	45		0	0	70868
1	0	3	3211	6364	19		650	0	42557
2	0	3	26	11	23	...	0	0	83720
3	0	3	9941	13	144		0	0	43388
4	0	3	1028	7838	59		0	1	77709
5	0	3	553	1	13		0	0	11976
6	0	3	2234	9	21		0	0	29149
7	1	1	4750	13	35		943	1	61408
8	1	1	4361	116	145		0	0	347298
9	1	1	162	99	190		0	0	511545
10	1	1	3685	1262	176		0	2147	309067
11	1	1	101	12847	120		0	0	235851

Figure 7.2: Repeated measures data set

The objective of the data analysis is to discover bacteria which behave differently between groups over time in the hopes of discovering possible probiotics, which are bacteria that promote wound healing. These probiotics become candidates to heal wounds for human patients, in particular those with diabetes. Identifying pathogens, or bacteria which prevent wound healing, would also be helpful. Diabetic patients, who often face difficulty with wound healing, are analogous to the mice with chronic wounds in the experiments, while non-diabetic patients are analogous to the mice with non-chronic wounds. It is hypothesized that probiotic bacteria could flourish in a non-chronic wound to improve wound healing, while pathogenic bacteria dominate in chronic wounds. That hypothesis would need to be tested in future validation experiments, as the experiments discussed here only identify candidates for probiotics and pathogens. The data set was provided by Dr. Manuela Martins-Green and PhD student Jane Kim.

7.1 Existing Model Fitting

The GEE model discussed in Chapter 3 was applied to this data set. GEE was fit with an identity link function and a non-parametric time trend using dummy variables for each day. The resulting model convergence problems arising in each case largely motivated the proposed model in Chapter 4. Ultimately, only 39 of the models fit to the 100 most abundant bacteria converged with GEE. With a sample size of 37 mice with chronic wounds and 40 mice with non-chronic wounds, some significant tests were produced from models that did converge. However, the 39% success rate was not an adequate outcome that will help advance the science. The GEE model would have been a good choice if it had succeeded in yielding converging models more often, but since this became such a persistent problem, it is apparent that an alternative is needed to remedy this issue.

7.2 Proposed Model Fitting

The proposed methodology discussed in Chapter 4 was implemented on the wound healing data set as well. In this section we describe the prior distributions used to implement the proposed model, provide ranges of proposal variances that worked well for the wound healing data, and summarize the results.

7.2.1 Priors

The priors for σ_1 and σ_2 were $\Gamma(10/256, 1/256)$ and $\Gamma(1/2, 1/8)$, where $\Gamma(s, t)$ is a gamma distribution with mean s/t and variance s/t^2. The prior for $\log \boldsymbol{\lambda}$ was MVN($\mathbf{0}$, $T(\sigma_1, \sigma_2)$) for given values of σ_1 and σ_2, where elements of T are defined in (3.8). The prior for $\log \boldsymbol{\beta}$ was MVN$(\mathbf{0}, 50I)$. Prior distributions for ϕ_A and ϕ_B were both $\Gamma(5/8, 1/16)$. Each distribution was chosen to have a large enough variance so that the prior would not strongly influence posterior draws.

7.2.2 Proposals

While fitting the model to the data, reasonable values for proposal variances for each parameter needed to be chosen. Proposal variances for $\log \boldsymbol{\lambda}$ and $\log \boldsymbol{\beta}$ were often adjusted for each time point. Referring to Section 4.1.2, proposal variances ranged from 1 to 6 for $\sigma^2_{\beta_k}$, from 0.015 to 3 for $\sigma^2_{\lambda_k}$, from 1.5 to 10 for a, from 1.4 to 3 for b, from 0.3 to

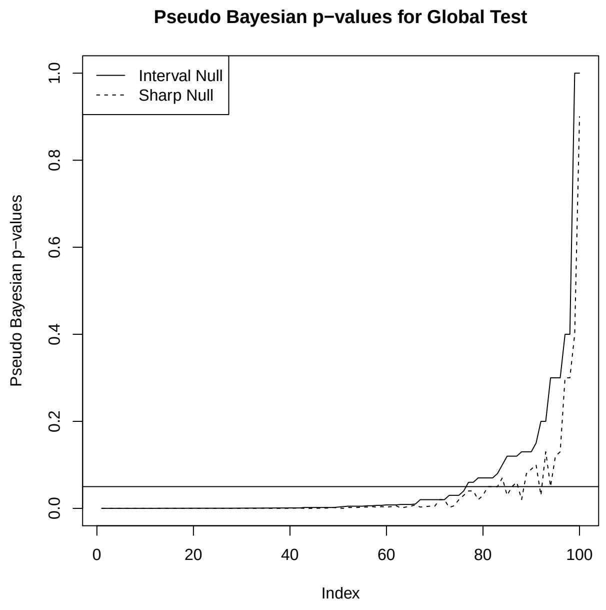


Figure 7.3: Pseudo Bayesian p-values for the 100 most abundant bacteria, sorted from smallest to largest value under the interval null. The interval null used was $H_0 : \beta \in (1/1.5, 1.5)$. A horizontal line is drawn at 0.05.

18 for c, and from 0.7 to 20 for d. These ranges should provide a good starting point for fitting the model to other NGS data sets.

7.2.3 Results

We implemented the global test for group difference on the wound healing data, which includes the pseudo Bayesian p-values as well as the hyper-rectangular joint credible regions. We looked at results for a point null hypothesis for each time point ($H_0 : \boldsymbol{\beta} = \mathbf{1}$), and a null interval allowing for 50% change of means as practically equivalent ($H_0 : \boldsymbol{\beta} \in (\frac{1}{1.5}, 1.5)$). These yielded pseudo Bayesian p-values less than 0.05 for 88 and 76 bacteria, respectively, and each of the 76 bacteria that rejected the interval null also rejected the point null. See Figure 7.3 for a visual of these pseudo Bayesian p-values. Note that the lines in the figure for the interval null and the sharp null may meet, but will not cross, since the sharp null is a subset of the interval null (see Section 5.2).

Figure 7.4 shows the hyper-rectangular joint 95% credible intervals for the $\boldsymbol{\beta}$ parameters for the four most abundant bacteria. Notice that for bacteria 1 and 2, each marginal credible interval has some overlap with the point null, and thus also both null intervals. Hence there is no evidence to suggest that these bacteria behave differently for the mice with chronic and non-chronic wounds. Plots for bacteria 3 and 4 show some marginal intervals that not only exclude the point null value of 1, but are also disjoint from the null interval of (1/1.5, 1.5). However, the trend of intervals of β differ between these two bacteria. Bacteria 3 has marginal intervals that are larger than the null interval, suggesting that bacteria 3 is a potential pathogen, since the counts are higher in mice in the chronic group than the non-chronic group, so this bacteria flourishes in a chronic wound. Bacteria 4 has marginal intervals that are smaller than the null interval, suggesting that bacteria 4 is a potential probiotic since the counts get muted in the chronic group. Figure 7.4 demonstrates that output of our proposed model can be used not only to perform inference at each time point, but also to identify the direction of group difference over time.

Using the hyper-rectangular joint 95% credible regions, the SGPV was calculated at each time point. Figure 7.5 shows how bacteria cluster according to their SGPVs over time, and each bacteria is represented as a row in the heatmap. Many bacteria clustered toward the top of the heatmap have SGPVs near 0.5, meaning that these bacteria do not provide evidence that they affect wound healing. Another cluster around the center of the rows show smaller values of SGPVs in the last two or three time points, suggesting that

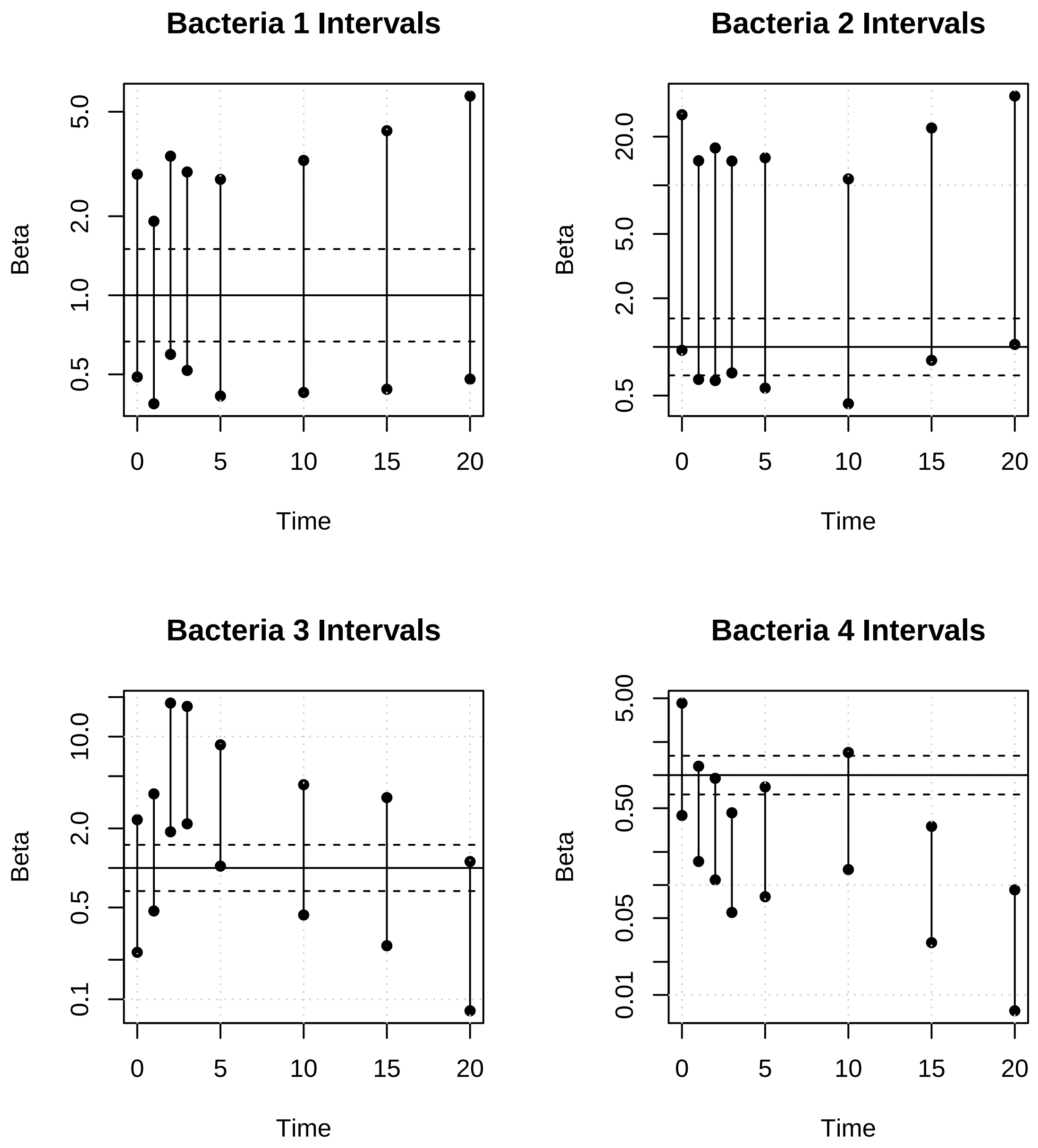


Figure 7.4: Semi-log plot with hyper-rectangular joint 95% credible intervals as well as horizontal lines indicating the location of $\beta = 1$ (solid) and $\beta = 1/1.5, 1.5$ (dashed). Pseudo Bayesian p-values for the global test of $H_0 : \boldsymbol{\beta} = \mathbf{1}$ are, in order, 0.9, 0.05, 0.0001, and 0.0001.

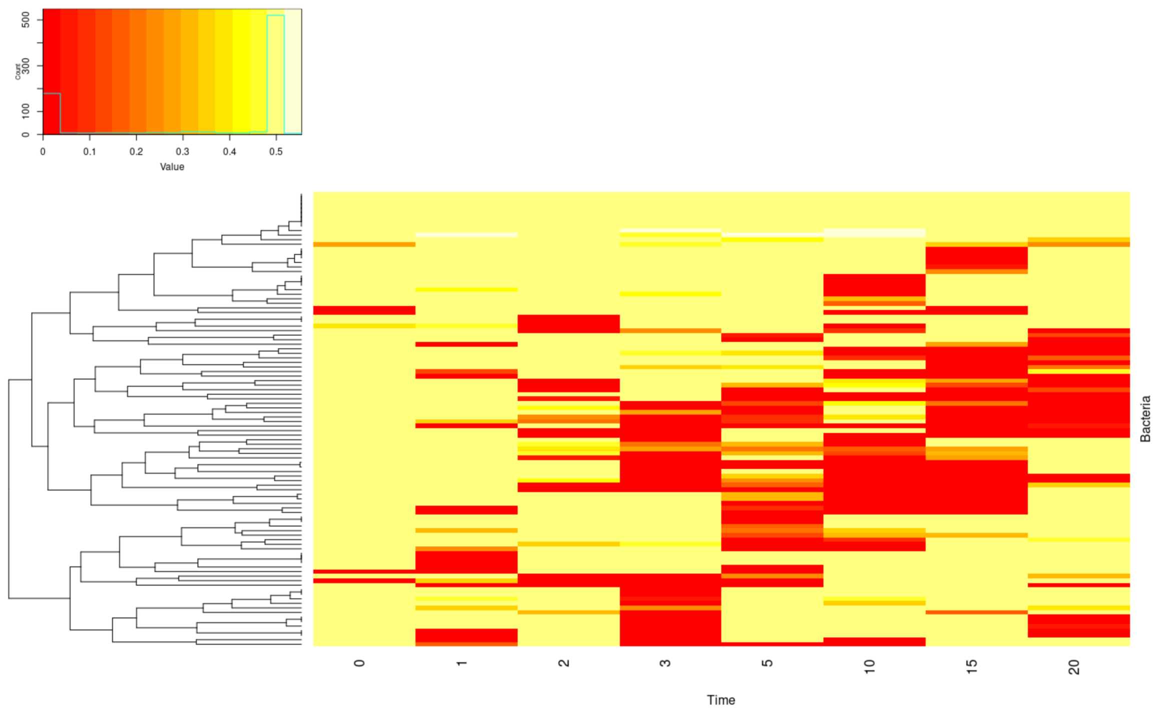


Figure 7.5: SGPVs were calculated at each time point for each bacteria. Tags were then clustered into similar groups according to their SGPVs over time. The heatmap contains 100 rows for 100 bacteria.

groups differ in these later time points for these bacteria. With some exceptions, bacteria typically do not show differences between groups at day 0, which makes sense because the bacteria are expected to behave similarly in the two groups before the experiment begins. Row labels for the names of each bacteria in the heatmap were too small to see in the figure and are provided in Appendix B.2.

The tests for interaction based on differences and ratios as described in Sections 5.4.1 and 5.4.2 are shown in Table 7.1 for the four most abundant bacteria. The two forms of the test for interaction can yield varying results, especially looking at the disparity in pseudo Bayesian p-values for bacteria 2 and 4, but both give similar inference conclusions.

	Interaction type	
Bacteria	Difference	Ratio
1	0.8000	0.8000
2	0.3000	0.9000
3	0.0030	0.0030
4	0.0100	0.0001

Table 7.1: Pseudo Bayesian p-values for interaction based on differences and ratios for the first 4 bacteria are shown.

Clustering bacteria which behave similarly could be useful for identifying probiotics or pathogens. Hypothesis testing is not required for clustering, and instead information from the posterior distribution is used to assess which bacteria have similar patterns of group differences over time. Figure 7.6 shows a dendrogram for the 100 bacteria, which are clustered based on means of posterior draws of $\log \boldsymbol{\beta}$. The dendrogram was formed with the `hclust` function in the `stats` package in `R`. Squared Euclidean distances are used in this clustering method to define the dissimilarity structure. Names corresponding to the numbered bacteria are provided in Appendix B.1.

Goodness-of-fit (GOF) of the proposed methodology can be assessed by combining MCMC draws of $\log \boldsymbol{\lambda}$ and $\log \boldsymbol{\beta}$ with library sizes, and then comparing these with empirical estimates from the data. Figure 7.7 shows jittered boxplots of data from the four most abundant bacteria for each group over time, with means from the proposed model plotted.

Another metric to assess GOF is described in Mi et al. (2015). Their method does not account for longitudinal data sets, so when we applied it to our data, we obtained a GOF p-value for each group at each day. The four most abundant bacteria yielded squared vertical distance GOF p-values that rejected adequacy for 10/64 tests at $\alpha = 0.01/16$, a Bonferroni-corrected threshold correcting for the number of groups (2) and the number of

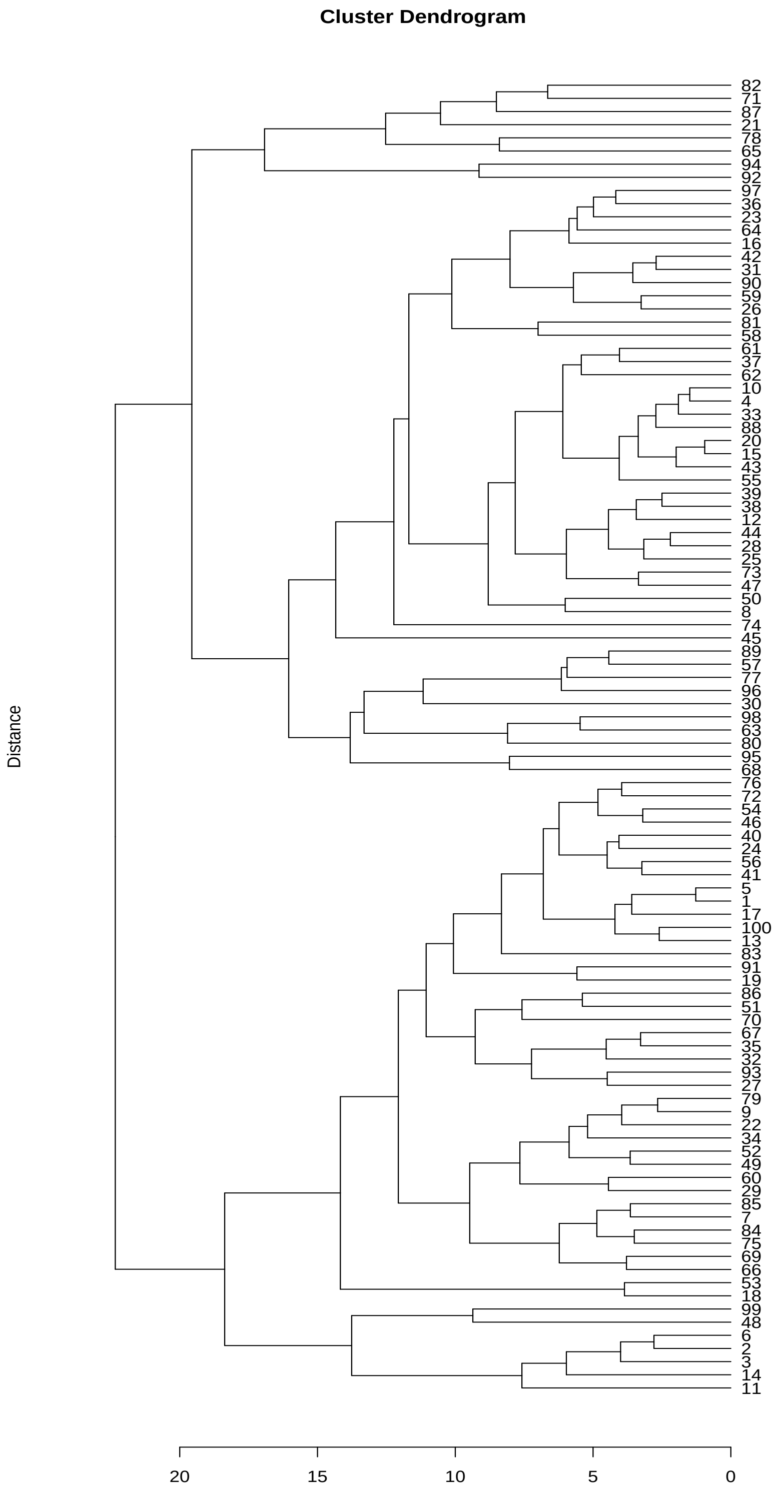


Figure 7.6: Dendrogram of the 100 bacteria, with posterior means of $\log \boldsymbol{\beta}$ used for clustering.

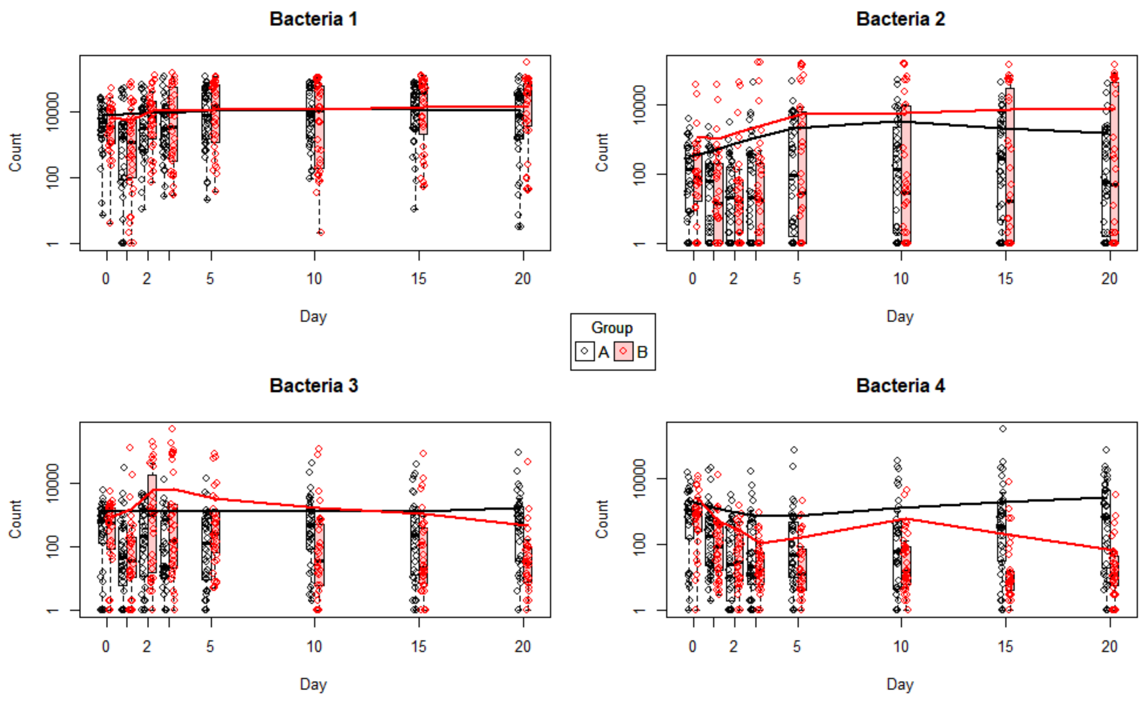


Figure 7.7: Fitted means from the proposed model for the four most abundant bacteria. Points in black indicate log-counts for non-chronic mice, while points in red indicate counts for chronic mice. There were 1, 109, 31, and 21 zero counts each that were set equal to 1 from these four bacteria in order to generate the plot. Groups A and B correspond to the non-chronic and chronic groups, respectively.

time points (8). The NB assumption is typically not violated via this metric, and when it is, generally an extremely large count causes the lack of fit.

Chapter 8

Conclusions

The inference methodology proposed in this paper extends the practitioner's toolbox when analyzing NGS data. The model is designed specifically for longitudinal count data sets, explicitly incorporates library sizes without normalizing to them, addresses model convergence via Bayesian methods, does not require synchronous measurements, and utilizes the GP prior that allows a robust characterization of time trends.

Additionally, our method of testing for group differences using a multi-dimensional version of ROPE allows for improved interpretation compared with more commonly used Bayesian hypothesis testing methods. By using posterior draws of model parameters to construct joint credible regions, we control interval coverage while retaining valuable information about parameter magnitudes, uncertainty, and direction. The method encourages users to consider how much group difference is considered practically equivalent by defining a null interval for the easily interpretable parameters which concern group difference. While information about the full posterior distribution of these parameters should not be ignored, summarizing them with joint credible intervals allows us to make a decision at each time point about whether there is a difference in groups, and if so which direction the difference exists.

Future work could include extending this model to test difference between more than two groups. Other models, such as edgeR, already have this functionality. Extending the model described in Chapter 4 to multiple groups would be a bit more involved, but should be possible through a more thorough specification of group means in the NB distribution. An extension to allow for other covariates that describe differences between subjects could be useful.

Additionally, different covariance structures could be investigated in the GP prior instead of the squared exponential function. Roberts et al. (2013) describe a wide assortment of alternative choices for covariance structures, including white noise, the rational quadratic kernel, Matérn, multiple inputs and outputs, periodic and quasi-periodic kernels, and changepoint functions. Our choice of the squared exponential function was motivated by the fact that it is a widely used covariance function for Gaussian Processes.

Another potential improvement to model fitting could be made by incorporating a mixture of NB distributions. While identifying instances of lack-of-fit, it was discovered that some outliers in the data set were impactful. We do not feel that these outliers should be ignored, however, as they may be informative data points. Rather, we could explore using a mixture model with a NB distribution that has a dispersion parameter which allows for these large data points, similar to the way that GPTwoSample used a mixture model to allow for outliers (Stegle et al., 2010).

Bibliography

Äijö, T., Butty, V., Chen, Z., Salo, V., Tripathi, S., Burge, C. B., Lahesmaa, R., and Lähdesmäki, H. (2014). Methods for time series analysis of RNA-seq data with application to human Th17 cell differentiation. *Bioinformatics*, 30(12):i113–i120.

Anders, S. and Huber, W. (2010). Differential expression analysis for sequence count data. *Genome Biology*, 11(R106).

Blume, J. D., Greevy, R. A., Welty, V. F., Smith, J. R., and Dupont, W. D. (2019). An Introduction to Second-Generation p-Values. *American Statistician*, 73(sup1):157–167.

Boos, D. D. (1992). On generalized score tests. *The American Statistician*, 46(4):327–333.

Chen, E. Z. and Li, H. (2016). A two-part mixed-effects model for analyzing longitudinal microbiome compositional data. *Bioinformatics*, 32(17):2611–2617.

Christensen, R., Johnson, W., Branscum, A., and Hanson, T. E. (2010). *Bayesian ideas and data analysis: An introduction for scientists and statisticians.* Press, CRC, Boca Raton, FL.

Ebden, M. (2015). Gaussian processes: A quick introduction. In *arXiv e-prints*, page arXiv:1505.02965.

Flegal, J. (2016). Statistics for data scientists: Monte carlo and mcmc simulations [slides]. Retrieved from http://faculty.ucr.edu/~jflegal/StatisticsForDataScientists.pdf.

Flegal, J. M., Hughes, J., Vats, D., and Dai, N. (2020). *mcmcse: Monte Carlo Standard Errors for MCMC*. Riverside, CA, Denver, CO, Coventry, UK, and Minneapolis, MN. R package version 1.4-1.

Genz, A., Bretz, F., Miwa, T., Mi, X., Leisch, F., Scheipl, F., and Hothorn, T. (2020). *mvtnorm: Multivariate Normal and t Distributions*. R package version 1.1-0.

Halekoh, U., Hjsgaard, S., and Yan, J. (2006). The r package geepack for generalized estimating equations. *Journal of Statistical Software*, 15/2:1–11.

Jones, G. L. (2004). On the Markov chain central limit theorem. *Prob. Surveys*, 1:299–320.

Kim, J. H., Ruegger, P. R., Lebig, E. G., VanSchalkwyk, S., Jeske, D. R., Hsiao, A., Borneman, J., and Martins-Green, M. (2020). High Levels of Oxidative Stress Create a Microenvironment That Significantly Decreases the Diversity of the Microbiota in Diabetic Chronic Wounds and Promotes Biofilm Formation. *Frontiers in Cellular and Infection Microbiology*, 10(June):1–20.

Kruschke, J. K. (2015). Bayesian Approaches to Testing a Point (Null) Hypothesis. *Doing Bayesian Data Analysis*, pages 143–191, 335–358.

Laird, N. M. (1989). Longitudinal Data Analysis for Counts and Binary Outcomes: Generalized Estimating Equations (GEE). *Analysis of Longitudinal and Cluster-Correlated Data*, 6(1988):96–109.

Law, C. W., Chen, Y., Shi, W., and Smyth, G. K. (2014). voom: precision weights unlock linear model analysis tools for RNA-seq read counts. *Genome Biology*, 15(29).

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

Love, M. I., Huber, W., and Anders, S. (2014). Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biology*, 15(12):1–21.

McShane, B. B., Gal, D., Gelman, A., Robert, C., and Tackett, J. L. (2019). Abandon Statistical Significance. *American Statistician*, 73(sup1):235–245.

Mi, G., Di, Y., and Schafer, D. W. (2015). Goodness-of-fit tests and model diagnostics for negative binomial regression of RNA sequencing data. *PLoS ONE*, 10(3):1–16.

Newton, M. A. and Raftery, A. E. (1994). Approximate Bayesian Inference with the Weighted Likelihood Bootstrap. *Journal of the Royal Statistical Society*, 56(1):3–48.

R Development Core Team (2011). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. Massachusetts Institute of Technology, Cambridge, Massachusetts.

Roberts, S., Osborne, M., Ebden, M., Reece, S., Gibson, N., and Aigrain, S. (2013). Gaussian processes for time-series modelling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1984):1–27.

Robertson, N., Flegal, J. M., Jones, G. L., and Vats, D. (2019). New visualizations for monte carlo simulations. In *arXiv e-prints*, page arXiv:1904.11912v1.

Robinson, M. D., McCarthy, D. J., and Smyth, G. K. (2009). edgeR: A Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics*, 26(1):139–140.

Robinson, M. D. and Smyth, G. K. (2007). Moderated statistical tests for assessing differences in tag abundance. *Bioinformatics*, 23(21):2881–2887.

Robinson, M. D. and Smyth, G. K. (2008). Small-sample estimation of negative binomial dispersion, with applications to SAGE data. *Biostatistics*, 9(2):321–332.

Rossi, P. (2019). *bayesm: Bayesian Inference for Marketing/Micro-Econometrics.* R package version 3.1-4.

Rotnitzky, A. and Jewell, N. P. (1990). Hypothesis testing of regression parameters in semiparametric generalized linear models for cluster correlated data. *Biometrika*, 77(3):485–497.

SAS Institute Inc. (2012). *SAS/STAT®* Software, Version 9.4 User's Guide. Cary, NC: SAS Institute Inc.

Smyth, G. K. (2004). Linear Models and Empirical Bayes Methods for Assessing Differential Expression in Microarray Experiments. *Statistical Applications in Genetics and Molecular Biology*, 3(1):1–25.

Smyth, G. K. (2005). limma: Linear Models for Microarray Data. *Bioinformatics and Computational Biology Solutions Using R and Bioconductor*, pages 397–420.

Stegle, O., Denby, K. J., Cooke, E. J., Wild, D. L., Ghahramani, Z., and Borgwardt, K. M. (2010). A Robust Bayesian Two-Sample Test for Detecting Intervals of Differential Gene Expression in Microarray Time Series. *Journal of Computational Biology*, 17(3):355–367.

Stern, H. S. (2005). Model inference or model selection: Discussion of Klugkist, Laudy, and Hoijtink (2005). *Psychological Methods*, 10(4):494–499.

VanSchalkwyk, S. (2018). Statistical Modeling and Analysis of Longitudinal Microbiome Data [slides]. UCR Microbiome Initiative, University of California, Riverside.

Vats, D., Flegal, J. M., and Jones, G. L. (2019). Multivariate output analysis for Markov chain Monte Carlo. *Biometrika*, 106(2):321–337.

Warnes, G. R., Bolker, B., Bonebakker, L., Gentleman, R., Huber, W., Liaw, A., Lumley, T., Maechler, M., Magnusson, A., Moeller, S., Schwartz, M., and Venables, B. (2020). *gplots: Various R Programming Tools for Plotting Data.* R package version 3.0.3.

Wickham, H., Hester, J., and Chang, W. (2020). *devtools: Tools to Make Developing R Packages Easier.* R package version 2.3.0.

Zeger, S. L., Liang, K.-Y., and Albert, P. S. (1988). Models for Longitudinal Data: A Generalized Estimating Equation Approach. *International Biometric Society*, 44(4):1049–1060.

Zhao, S., Fung-Leung, W. P., Bittner, A., Ngo, K., and Liu, X. (2014). Comparison of RNA-Seq and microarray in transcriptome profiling of activated T cells. *PLoS ONE*, 9(1).

Appendix

Appendix A Compositional vs. Raw Data

As NGS data produce data which are only interpretable relative to their respective library sizes, a common practice is to normalize counts to the library size. That is, ratios are computed by dividing counts by their library size, and the ratios are then used for analysis. Consequentially, information about the original magnitude of the data is lost. Here we consider a simulation of data sets which are analyzed for differential expression with edgeR and ZIBR to compare performance of analysis on unnormalized data to analysis on compositional data.

Non-longitudinal data is sufficient to compare compositional to raw data. One hundred data sets were simulated in four situations, allowing for group sizes of 10 each and 40 each, and a dispersion parameter of 0.1 and 2. Library sizes were simulated from a discrete uniform distribution with endpoints 10,000 and 40,000. There were 100 tags for each data set, 50 of which were differentially expressed, and the other 50 were not. Using the mean structure of edgeR, the relative abundance terms were chosen to sum to one across tags. Five sets of group differences were used, allowing for group A to be

1.5, 2, 5, 10, and 20 times as large as group B for 25 tags, and allowing for group B to be 1.5, 2, 5, 10, and 20 times as large as group A for 25 tags.

Results in Table A.1 show overall higher power with edgeR to detect group differences compared with ZIBR. Particularly when the group size is 10 and $\phi = 2$, it becomes evident with larger group differences that edgeR has higher power. Table A.2 shows Type I error rates for each simulation setting, where the nominal size is 5%. ZIBR p-values were FDR corrected to be comparable to edgeR results. Notice in comparing these tables that power is generally low in situations where the Type I error is low. All results in the table show Type I error below 5%.

	Group Size							
	10				40			
ϕ	0.1		2		0.1		2	
model / diff	edgeR	ZIBR	edgeR	ZIBR	edgeR	ZIBR	edgeR	ZIBR
1.5	65.5	44.5	0.3	0.2	100.0	99.8	2.1	0.2
2	99.5	96.4	0.8	0.4	100.0	100.0	28.6	1.7
5	100.0	100.0	43.1	4.92	100.0	100.0	99.7	70.8
10	100.0	100.0	88.5	18.0	100.0	100.0	100.0	97.0
20	100.0	100.0	98.7	34.3	100.0	100.0	100.0	99.8

Table A.1: Percent power for edgeR and ZIBR on simulated data to compare raw data analysis to compositional data analysis. The 'diff' column indicates the magnitude of group difference.

	Group Size							
	10				40			
ϕ	0.1		2		0.1		2	
model / diff	edgeR	ZIBR	edgeR	ZIBR	edgeR	ZIBR	edgeR	ZIBR
1.5	2.1	1.4	0.1	0.0	2.5	0.7	0.1	0.0
2	2.6	2.2	0.2	0.1	2.9	0.9	0.9	0.0
5	2.7	2.2	1.5	0.2	2.4	0.6	2.3	1.7
10	2.5	1.8	2.5	0.5	2.5	0.8	2.8	2.4
20	2.5	1.9	3.3	1.1	3.0	1.3	2.9	2.2

Table A.2: Type I error rates for edgeR and ZIBR on simulated data to compare raw data analysis to compositional data analysis. The 'diff' column indicates the magnitude of group difference.

Appendix B Bacteria Lists

B.1 Bacteria Names

1. *Enterobacter cloacae*
2. *Pseudomonas aeruginosa*
3. *Staphylococcus xylosus*
4. *Streptococcus* sp.
5. Enterobacteriaceae
6. *Corynebacterium frankenforstense*
7. *Enterococcus gallinarum*
8. *Exiguobacterium* sp.
9. *Turicibacter* sp.
10. *Cutibacterium acnes*
11. *Acinetobacter* sp.
12. *Lactobacillus johnsonii*
13. *Lactobacillus murinus*
14. *Bacillus paralicheniformis*
15. *Escherichia* sp.
16. *Weissella paramesenteroides*
17. *Bacillus* sp.
18. *Aerococcus urinaeequi*
19. *Enterococcus faecalis*
20. *Escherichia coli*
21. *Corynebacterium urealyticum*
22. *Lactobacillus reuteri*
23. *Faecalibaculum rodentium*
24. *Staphylococcus nepalensis*
25. *Helicobacter hepaticus*
26. *Massilia oculi*
27. *Streptococcus* sp.
28. *Staphylococcus epidermidis*
29. *Epulopiscium* sp.
30. *Lactobacillus crispatus*

31. *Pantoea agglomerans*
32. *Paenibacillus* sp.
33. *Achromobacter* sp.
34. *Massilia* sp.
35. Bacteroidales
36. *Staphylococcus capitis*
37. *Staphylococcus hominis*
38. *Muribaculum intestinale*
39. *Bacteroides uniformis*
40. *Alistipes finegoldii*
41. *Candidatus Arthromitus* sp.
42. *Pantoea* sp.
43. *Lactobacillus* sp.
44. *Prevotella denticola*
45. *Bacillus subtilis*
46. Bacteroidetes
47. Flavobacteriaceae
48. *Enterococcus hirae*
49. Lachnospiraceae
50. *Pantoea vagans*
51. *Erysipelothrix rhusiopathiae*
52. *Anaerostipes hadrus*
53. *Aerococcus* sp.
54. *Odoribacter splanchnicus*
55. *Ralstonia* sp.
56. *Bacteroides salanitronis*
57. *Clostridium saccharolyticum*
58. *Rothia mucilaginosa*
59. *Sporosarcina psychrophila*
60. Bacilli
61. *Geobacillus* sp.
62. *Streptococcus equinus*
63. *Echinicola* sp.
64. *Lawsonella clevelandensis*
65. *Salinicoccus* sp.
66. *Bacteroides massiliensis*
67. *Bacteroides caecimuris*
68. Erysipelotrichaceae
69. *Prevotella* sp.

70. *Desulfovibrio fairfieldensis*
71. *Corynebacterium propinquum*
72. *Lachnoclostridium Clostridium*
73. *Dyadobacter fermentans*
74. *Clostridium* sp.
75. *Porphyromonas endodontalis*
76. *Anaerotignum propionicum*
77. *Parabacteroides merdae*
78. *Marinilactibacillus* sp.
79. *Lachnoclostridium phocaeense*
80. *Aneurinibacillus soli*
81. *Lysinibacillus* sp.
82. *Bacillus glycinifermentans*
83. *Actinobacteria* sp.
84. *Prevotella buccae*
85. *Lactobacillus delbrueckii*
86. *Pantoea* sp.
87. *Corynebacterium choanis*
88. *Delftia* sp.
89. *Blautia hansenii*
90. *Mucinivorans hirudinis*
91. *Pantoea ananatis*
92. *Lactobacillus agilis*
93. *Porphyromonas gulae*
94. *Thauera aromatica*
95. *Porphyromonas asaccharolytica*
96. *Adlercreutzia equolifaciens*
97. *Pseudomonas* sp.
98. *Planococcus* sp.
99. *Geobacillus* sp.
100. Other

B.2 Row labels for Figure 7.5

Bacteria from top to bottom in Figure 3 are: 55, 48, 62, 70, 72, 73, 83, 98, 1, 5, 29, 2, 49, 34, 52, 91, 56, 93, 54, 46, 82, 21, 38, 76, 86, 65, 78, 74, 50, 39, 14, 81, 59, 23, 51, 100, 85, 60, 79, 84, 18, 90, 42, 22, 7, 26, 97, 9, 33, 10, 4, 89, 31, 87, 67, 20, 6, 15, 44, 12, 11, 77, 27, 53, 28, 30, 75, 66, 69, 96, 57, 58, 24, 41, 71, 13, 40, 25, 47, 17, 8, 45, 80, 92, 3, 94, 16, 99, 95, 43, 61, 19, 37, 68, 64, 88, 36, 32, 35, 63.

Appendix C Functions and Code

C.1 Manual

`mhNBsamp` *Longitudinal trend analysis of count data with Gaussian Processes*

The `mhNBsamp` function implements a Metropolis Hastings MCMC algorithm to get draws from the posterior shown in equation 10. The algorithm implemented provides draws for $\log \boldsymbol{\lambda}$ and $\log \boldsymbol{\beta}$ which can be exponentiated to get draws for $\boldsymbol{\lambda}$ and $\boldsymbol{\beta}$.

Usage

```
mhNBsamp(s1init, s2init, pAinit, pBinit,
         loglaminit, logbetainit, data, times, ...)
```

Arguments

`s1init`	Initial value for σ_1
`s2init`	Initial value for σ_2
`pAinit`	Initial value for ϕ_A
`pBinit`	Initial value for ϕ_B
`loglaminit`	Initial values for $\log \boldsymbol{\lambda}$. Length must be equal to the length of `times`.
`logbetainit`	Initial value for $\log \boldsymbol{\beta}$. Length must be equal to the length of `times`.
`data`	A matrix with six columns indicating measured time points for data A, measured time points for data B, counts for data A, counts for data B, library sizes for data A, and library sizes for data B. If there are a different number of measurements in the groups, extra rows can be filled with `NA`.
`times`	A sorted vector of unique time points in the data

`numiter`	Number of MCMC iterations to run before checking `multiESS` condition (see `mcmcse` package). Default is `1e5`.
`w1`	Proposal variance for $\log \boldsymbol{\lambda}$ parameters. Length must be equal to the length of `times`. Default is `rep(0.9, length(times)`.
`sigmasq`	Proposal variance for $\log \boldsymbol{\beta}$ parameters. Length must be equal to the length of `times`. Default is `rep(5, length(times))`.
`s1var`	Proposal variance for σ_1. Default is `3`.
`s2var`	Proposal variance for σ_2. Default is `2.5`.
`pAvar`	Proposal variance for ϕ_A. Default is `2.5`.
`pBvar`	Proposal variance for ϕ_B. Default is `2.5`.
`tune`	If `TRUE`, requires `multiESS` of 1. Useful for tuning proposal variances if `numiter` is set to a smaller number (e.g. `500`).

Value

`sigma1`	Vector of posterior draws of σ_1
`sigma2`	Vector of posterior draws of σ_2
`phiA`	Vector of posterior draws of ϕ_A
`phiB`	Vector of posterior draws of ϕ_B
`loglambda`	Matrix of posterior draws of $\log \boldsymbol{\lambda}$ (number of columns equal to the length of `times`).
`logbeta`	Matrix of posterior draws of $\log \boldsymbol{\beta}$ (number of columns equal to the length of `times`).

`covmatrix` *Covariance function to define the Gaussian Process*

The `covmatrix` function returns the squared exponential covariance matrix formed by σ_1, σ_2, and `times`. It is a supporting function for the `posterior` function.

Usage

```
covmatrix(s1, s2, times)
```

Arguments

`s1`	Value for σ_1
`s2`	Value for σ_2
`times`	A sorted vector of unique time points in the data, originally passed into the `mhNBsamp` function

Value

A $K \times K$ squared exponential matrix formed by `s1`, `s2`, and `times`

`get_proposal_distribution` *Samples proposals for parameters*

The `get_proposal_distribution` returns a candidate value for the requested parameter of interest using the inputted proposal variance. It is a supporting function for `mhNBsamp`.

Usage

```
get_proposal_distribution(parameter, lastvalue, s1var = s1var,
                          s2var = s2var, pAvar = pAvar, pBvar = pBvar)
```

Arguments

`parameter`	Parameter of interest to generate a candidate value
`lastvalue`	Last accepted value of the MCMC chain for the parameter of interest
`s1var`	Proposal variance for σ_1
`s2var`	Proposal variance for σ_2
`pAvar`	Proposal variance for ϕ_A
`pBvar`	Proposal variance for ϕ_B

Value

A candidate value for the specified parameter `posterior` *Computes and returns the log of the posterior distribution*

The `posterior` function computes and returns the log of the posterior distribution for inputted parameter values using a Negative Binomial distribution. Priors for all parameters are defined here. This is a supporting function for the `mhNBsamp` function.

Usage

```
posterior(s1, s2, pA, pB, loglambda, logbeta, yA, yB, times,
          loglibA, loglibB, loglamAtimes, loglamBtimes)
```

Arguments

`s1`	Most recently accepted or candidate value for σ_1
`s2`	Most recently accepted or candidate value for σ_2
`pA`	Most recently accepted or candidate value for ϕ_A
`pB`	Most recently accepted or candidate value for ϕ_B
`loglambda`	Most recently accepted or candidate values for $\log \boldsymbol{\lambda}$
`logbeta`	Most recently accepted or candidate values for $\log \boldsymbol{\beta}$
`yA`	Data from group A
`yB`	Data from group B
`times`	A sorted vector of unique time points in the data, originally passed into the `mhNBsamp` function
`loglibA`	Log library sizes for group A corresponding to `yA`
`loglibB`	Log library sizes for group B corresponding to `yB`
`loglamAtimes`	Times corresponding to measurements in `yA`
`loglamBtimes`	Times corresponding to measurements in `yB`

Value

The log of the posterior distribution, computed as the log of the priors plus the log likelihood `get_coverage` *Computes coverage probability of a joint credible interval*

The `get_coverage` function returns a coverage probability for posterior draws and a given α by computing a $100(1-\alpha)\%$ hyper-rectangular joint credible interval and counting how many posterior draws are jointly in the hyper-rectangular interval. It is currently built only for an 8-dimensional case, and would need to be adjusted if $K \neq 8$. This function is a supporting function for `find_alpha`.

Usage

```
get_coverage(X, alpha_MID)
```

Arguments

`X`	A 6 item list returned from `mhNBsamp`
`alpha_MID`	The value of α to use to build a hyper-rectangular joint credible interval

Value

The joint coverage of the $100(1-\alpha)\%$ hyper-rectangular joint credible interval

`find_alpha` *Finds the calibrating value of α to get the desired joint coverage*

The `find_alpha` function uses a bisection method to find the calibrating value of α that will produce the desired joint coverage in a hyper-rectangular joint credible interval. It can be called individually for a particular tag, but is also used as a supporting function within the `getBayesianpvalues` function to compute pseudo Bayesian p-values.

Usage

```
find_alpha(bacnum, alpha = 0.05, tol = 0.0001, n_max = 100)
```

Arguments

`bacnum`	The tag number. Assumes file is saved as `paste0("bac", bacnum, "results.RDS")`
`alpha`	The desired α to form the hyper-rectangular joint credible intervals. Default is `0.05`
`tol`	The tolerance allowed for coverage to differ from $(1 - \alpha)$. Default is `0.0001`
`n_max`	The maximum number of iterations to search for α. At default of `100`, this maximum is hardly ever used.

Value

`alpha`	The calibrating value of α to get the desired joint coverage
`iterations`	The number of iterations needed to find the calibrating α
`coverage`	The actual coverage yielded by the calibrating α

`getBayesianpvalues` *Computes pseudo Bayesian p-values*

The `getBayesianpvalues` function returns pseudo Bayesian *p*-values for a chosen null hypothesis set. It assumes an output file from `mhNBsamp` for each tag is saved as `paste0("bac", bacnum, "results.RDS")`.

Usage

```
getBayesianpvalues(testingUB = 1.5, numtags)
```

Arguments

`testingUB`	Defines null hypothesis set. Default is `1.5`. Use `1` for sharp null.
`numtags`	The number of tags in the data set

Value

A $1\times$ `numtags` vector of pseudo Bayesian $p-$values

`getsgpv` *Computes Second-Generation p-values*

The `getsgpv` function computes Second-Generation p-values. It assumes output from `mhNBsamp` for each tag is saved as `paste0("bac",bacnum,"results.RDS")`.

Usage

```
getsgpv(numtags, K, nullinterval = log(c(1/1.5, 1.5)))
```

Arguments

`numtags`	The number of tags in the data set
`K`	Total number of unique time points
`nullinterval`	Endpoints of one dimension of a null hypothesis interval set for $\log \boldsymbol{\beta}$. Default is `log(c(1/1.5, 1.5))`

Value

A `numtags`$\times$ `K` matrix of Second-Generation $p-$values

C.2 R Code

```
# Download and load packages:
# library(gplots)
# library(devtools) # devtools::install_github("gu-mi/NBGOF")
# library(NBGOF)
# library(geepack)

# Functions to run the model
# Main function:
mhNBsamp <- function(s1init, s2init, pAinit, pBinit, loglaminit,
                     logbetainit, data, times, numiter = 100000,
                     w1 = rep(.9, length(times)), sigmasq =
```

```r
                   rep(5, length(times)), s1var = 3, s2var = 2.5,
                   pAvar = 2.5, pBvar = 2.5, tune = FALSE){
library(mvtnorm)
library(bayesm)
suppressMessages(library(mcmcse))

if(tune){
  minIIDdraws = 1
}
else{
  minIIDdraws <- minESS(4 + 2*(length(loglaminit)))
}
print(paste0("Requires at minimum an effective sample size of ",
minIIDdraws, "."))

yA <- data[,3][!is.na(data[,3])]; yB <- data[,4][!is.na(data[,4])]
loglibA <- data[,5][!is.na(data[,5])]
loglibB <- data[,6][!is.na(data[,6])]
loglamAtimes <- data[,1][!is.na(data[,1])]
loglamBtimes <- data[,2][!is.na(data[,2])]

numtimes <- length(times)

s1vals <- c(s1init, rep(NA, 8e5)); s2vals <- c(s2init, rep(NA, 8e5))
pAvals <- c(pAinit, rep(NA, 8e5)); pBvals <- c(pBinit, rep(NA, 8e5))
loglvals <- matrix(c(loglaminit, rep(NA, 8e5*length(loglaminit))),
                   ncol = length(loglaminit), byrow = TRUE)
logbvals <- matrix(c(logbetainit, rep(NA, 8e5*length(logbetainit))),
                   ncol = length(logbetainit), byrow = TRUE)

# Save most recently updated version of each parameter
s1current <- s1init; s2current <- s2init; logbetacurrent <- logbetainit
pAcurrent <- pAinit; pBcurrent <- pBinit; loglcurrent <- loglaminit

# Record of how many candidate values get accepted for each component
s1accepted <- 0; s2accepted <- 0
logbetaaccepted <- rep(0, length(logbetainit))
pAaccepted <- 0; pBaccepted <- 0
loglaccepted <- rep(0, length(loglaminit))

# Keep trackers of how often each component yields an error
logbetatrackers <- rep(0, length(logbetainit))
s1tracker <- 0; s2tracker <- 0
pAtracker <- 0; pBtracker <- 0
logltracker <- rep(0, length(loglaminit))

# Error catching -Inf/-Inf (NaN) ratios
is.error <- function(x) inherits(x, "try-error")
```

```r
# Save numerator or denominator of acceptance ratio to speed up
# computation
# Initialize value
saveND <- posterior(s1init, s2init, pAinit, pBinit, loglaminit,
                    logbetainit, yA, yB, times, loglibA, loglibB,
                    loglamAtimes, loglamBtimes)

i <- 1
extraiter <- 0
repeat {
  i <- i + 1

  # add in component-wise sampler here for beta
  for(j in 1:length(logbetainit)){
    temp <- NaN
    while(is.error(try(if(min(temp, 0) > log(runif(1))){ },
                       silent = TRUE))){
      logbetastar <- rnorm(1, logbvals[i-1,j], sd = sqrt(sigmasq[j]))
      logbetacurrent[j] <- logbetastar
      tempNum <- posterior(s1vals[i-1], s2vals[i-1], pAvals[i-1],
                           pBvals[i-1], loglvals[i-1,], logbetacurrent,
                           yA, yB, times, loglibA, loglibB,
                           loglamAtimes, loglamBtimes)
      temp <- tempNum - saveND
      logbetatrackers[j] <- logbetatrackers[j] + 1
      if(logbetatrackers[j] > 2*(numiter+extraiter)){
        message("Taking too long on log beta;
                 tracker twice initial iteration number")
        break
      }
    }
    if(min(temp, 0) > log(runif(1))){ # accept
      logbvals[i,j] <- logbetastar
      logbetaaccepted[j] <- logbetaaccepted[j] + 1
      saveND <- tempNum
    }
    else{ # reject
      logbvals[i,j] <- logbvals[i-1,j]
      logbetacurrent[j] <- logbvals[i-1,j]
    }
  }

  s1ratio <- NaN
  while(is.error(try(if(min(s1ratio, 0) > log(runif(1))){ },
                     silent = TRUE))){
    s1star <- get_proposal_distribution("sigma1", 0, lastvalue =
                                        s1vals[i-1], s1var = s1var)
    if(s1star <= 0){ # reject if <= 0
```

```r
      s1ratio <- -Inf
      break
    }
    s1ratioNum <- posterior(s1star, s2vals[i-1], pAvals[i-1],
                            pBvals[i-1], loglvals[i-1,],
                            logbetacurrent, yA, yB, times,
                            loglibA, loglibB,
                            loglamAtimes, loglamBtimes)
    s1ratio <- s1ratioNum - saveND
    s1tracker <- s1tracker + 1
    if(s1tracker > 2*(numiter + extraiter)){
      message("Taking too long on s1;
               tracker is twice initial iteration number")
      break
    }
  }
  if(min(s1ratio, 0) > log(runif(1))){ # accept
    s1current <- s1star; s1vals[i] <- s1star
    s1accepted <- s1accepted + 1
    saveND <- s1ratioNum
  }
  else{ # reject
    s1current <- s1vals[i-1]; s1vals[i] <- s1vals[i-1]
  }

  s2ratio <- NaN
  while(is.error(try(if(min(s2ratio, 0) > log(runif(1))){ },
                     silent = TRUE))){
    s2star <- get_proposal_distribution("sigma2", 0, lastvalue =
                                        s2vals[i-1], s2var = s2var)
    if(s2star <= 0){ # reject if <= 0
      s2ratio <- -Inf
      break
    }
    s2ratioNum <- posterior(s1current, s2star, pAvals[i-1],
                            pBvals[i-1], loglvals[i-1,],
                            logbetacurrent, yA, yB, times,
                            loglibA, loglibB, loglamAtimes,
                            loglamBtimes)
    s2ratio <- s2ratioNum - saveND
    s2tracker <- s2tracker + 1
    if(s2tracker > 2*(numiter + extraiter)){
      message("Taking too long on s2;
               tracker is twice initial iteration number")
      break
    }
  }
  if(min(s2ratio, 0) > log(runif(1))){ # accept
```

```
      s2current <- s2star; s2vals[i] <- s2star
      s2accepted <- s2accepted + 1
      saveND <- s2ratioNum
    }
    else{ # reject
      s2current <- s2vals[i-1]; s2vals[i] <- s2vals[i-1]
    }

    pAratio <- NaN
    while(is.error(try(if(min(pAratio, 0) > log(runif(1))){ },
                          silent = TRUE))){
      pAstar <- get_proposal_distribution("phiA", 0, lastvalue =
                                            pAvals[i-1], pAvar = pAvar)
      if(pAstar <= 0){ # reject if <= 0
        pAratio <- -Inf
        break
      }
      pAratioNum <- posterior(s1current, s2current, pAstar, pBvals[i-1],
                              loglvals[i-1,], logbetacurrent, yA, yB,
                              times, loglibA, loglibB,
                              loglamAtimes, loglamBtimes)
      pAratio <- pAratioNum - saveND
      pAtracker <- pAtracker + 1
      if(pAtracker > 2*(numiter + extraiter)){
        message("Taking too long on phi A;
                 tracker twice init iter number")
        break
      }
    }
    if(min(pAratio, 0) > log(runif(1))){ # accept
      pAcurrent <- pAstar; pAvals[i] <- pAstar
      pAaccepted <- pAaccepted + 1
      saveND <- pAratioNum
    }
    else{ # reject
      pAcurrent <- pAvals[i-1]; pAvals[i] <- pAvals[i-1]
    }

    pBratio <- NaN
    while(is.error(try(if(min(pBratio, 0) > log(runif(1))){ },
                          silent = TRUE))){
      pBstar <- get_proposal_distribution("phiB", 0, lastvalue =
                                            pBvals[i-1], pBvar = pBvar)
      if(pBstar <= 0){ # reject if <= 0
        pBratio <- -Inf
        break
      }
      pBratioNum <- posterior(s1current, s2current, pAcurrent, pBstar,
                              loglvals[i-1,], logbetacurrent, yA, yB,
                              times, loglibA, loglibB,
                              loglamAtimes, loglamBtimes)
      pBratio <- pBratioNum - saveND
      pBtracker <- pBtracker + 1
      if(pBtracker > 2*(numiter + extraiter)){
        message("Taking too long on phi B;
                 tracker twice init iter number")
        break
      }
    }
    if(min(pBratio, 0) > log(runif(1))){ # accept
      pBcurrent <- pBstar; pBvals[i] <- pBstar
      pBaccepted <- pBaccepted + 1
      saveND <- pBratioNum
    }
    else{ # reject
      pBcurrent <- pBvals[i-1]; pBvals[i] <- pBvals[i-1]
    }

    # add in component-wise sampler here for log lambda
    for(j in 1:length(loglaminit)){
      temp <- NaN
      while(is.error(try(if(min(temp, 0) > log(runif(1))){ },
                             silent = TRUE))){
        loglstar <- rnorm(1, loglvals[i-1,j], w1[j])
        loglcurrent[j] <- loglstar
        tempNum <- posterior(s1current, s2current, pAcurrent, pBcurrent,
                             loglcurrent, logbetacurrent, yA, yB, times,
                             loglibA, loglibB, loglamAtimes,
                             loglamBtimes)
        temp <- tempNum - saveND
        logltracker[j] <- logltracker[j] + 1
        if(logltracker[j] > 2*(numiter + extraiter)){
          message("Taking too long on log lambda;
                   tracker twice init iter number")
          break
        }
      }
      if(min(temp, 0) > log(runif(1))){ # accept
        loglvals[i,j] <- loglstar
        loglaccepted[j] <- loglaccepted[j] + 1
        saveND <- tempNum
      }
      else{ # reject
        loglvals[i,j] <- loglvals[i-1,j]
        loglcurrent[j] <- loglvals[i-1,j]
      }
```

```r
    }

    # Check if we have collected enough draws
    if(i > numiter){
      if(tune){
        break
      }
      else if((i+1) %% 100 == 0){
        s1noNA <- s1vals[!is.na(s1vals)]
        s2noNA <- s2vals[!is.na(s2vals)]
        pAnoNA <- pAvals[!is.na(pAvals)]
        pBnoNA <- pBvals[!is.na(pBvals)]
        loglnoNA <- matrix(loglvals[!is.na(loglvals)],
                           ncol = ncol(loglvals))
        logbnoNA <- matrix(logbvals[!is.na(logbvals)],
                           ncol = ncol(logbvals))
        allparameters <- cbind(s1noNA, s2noNA, pAnoNA, pBnoNA, loglnoNA,
                               logbnoNA)
        if(multiESS(allparameters) >= minIIDdraws){
          break
        }
        else{
          extraiter <- extraiter + 1
        }
      }
      else{
        extraiter <- extraiter + 1
      }
    }
  }
  loglvals <- matrix(loglvals[!is.na(loglvals)], ncol = ncol(loglvals))
  logbvals <- matrix(logbvals[!is.na(logbvals)], ncol = ncol(logbvals))
  s1vals <- s1vals[!is.na(s1vals)]
  s2vals <- s2vals[!is.na(s2vals)]
  pAvals <- pAvals[!is.na(pAvals)]
  pBvals <- pBvals[!is.na(pBvals)]

  totaliter <- length(s1vals)
  message("Number of attempts per component:")
  message(paste(s1tracker, s1tracker, pAtracker, pBtracker, logltracker,
  sep = ", "))
  tracklbeta <- logbetatrackers[1]
  for(t in 2:length(logbetatrackers)){
   tracklbeta <- cat(tracklbeta, logbetatrackers[t], sep = ", ")
  }
  message(tracklbeta/totaliter)

  # When the chain ends, return the sampled posterior values
  return(list("s1" = s1vals, "s2" = s2vals, "pA" = pAvals, "pB" = pBvals,
              "log lambda" = loglvals, "log beta" = logbvals))
}

covmatrix <- function(s1, s2, times){
  K <- matrix(NA, nrow = length(times), ncol = length(times))
  for(i in 1:length(times)){
    for(j in 1:length(times)){
      K[i,j] <- (s1^2)*exp(-(1/2)*(times[i]-times[j])^2/(s2^2))
    }
  }
  return(K)
}
get_proposal_distribution <- function(parameter, lastvalue,
                                      s1var = s1var, s2var = s2var,
                                      pAvar = pAvar, pBvar = pBvar){
  if(parameter == "sigma1"){
    return(runif(1, lastvalue - s1var, lastvalue + s1var))
  }
  else if(parameter == "sigma2"){
    return(runif(1, lastvalue - s2var, lastvalue + s2var))
  }
  else if(parameter == "phiA"){
    return(runif(1, lastvalue - pAvar, lastvalue + pAvar))
  }
  else if(parameter == "phiB"){
    return(runif(1, lastvalue - pBvar, lastvalue + pBvar))
  }
}
posterior <- function(s1, s2, pA, pB, loglambda, logbeta, yA, yB, times,
                      loglibA, loglibB, loglamAtimes, loglamBtimes){
  sigma1 <- dgamma(s1, 10/(2^8), 1/(2^8))
  sigma2 <- dgamma(s2, 4/(2^3), 1/(2^3))
  phiA <- dgamma(pA, 10/(2^4), 1/(2^4))
  phiB <- dgamma(pB, 10/(2^4), 1/(2^4))
  loglambdaprior <- sum(lndMvn(loglambda, rep(0, length(loglambda)),
                              backsolve(chol(covmatrix(s1, s2, times)),
                                        diag(length(loglambda)))))
  logbetaprior <- sum(lndMvn(logbeta, rep(0, length(logbeta)),
                            backsolve(chol(50*diag(length(logbeta))),
                                      diag(length(logbeta)))))
  logpriors <- sum(log(sigma1), log(sigma2), log(phiA), log(phiB),
                   loglambdaprior, logbetaprior)

  # likelihood
  loglikelihood <- 0
  for(r in 1:length(yA)){
    for(t in 1:length(times)){ #These lines ensure correct component of
```

```
      if(loglamAtimes[r] == times[t]){ #log lambda aligns w/ time of obs
        loglikelihood <- loglikelihood +
                         log(dnbinom(yA[r], mu = exp(loglibA[r] +
                             loglambda[match(loglamAtimes[r], times)]),
                                     size = 1/pA))
      }
    }
  }
  for(r in 1:length(yB)){
    for(t in 1:length(times)){
      if(loglamBtimes[r] == times[t]){
        loglikelihood <- loglikelihood +
                         log(dnbinom(yB[r], mu = exp(loglibB[r] +
                             loglambda[match(loglamBtimes[r], times)] +
                             logbeta[match(loglamBtimes[r], times)]),
                             size = 1/pB))
      }
    }
  }

  return(logpriors+loglikelihood)
}

# pseudo-Bayesian p-values
get_coverage <- function(X, alpha_MID){
  total_MCMC_samples <- length(X[[1]])
  all_CIs_MID <- matrix(NA, nrow = 8, ncol = 2)
  for(i in 1:8){
    all_CIs_MID[i,] <- quantile(X[[6]][,i], probs = c(alpha_MID/2,
    1-(alpha_MID/2)))
  }
  coverage_MID <- 0
  for(i in 1:total_MCMC_samples){
    if(all_CIs_MID[1,1] < X[[6]][i,1]&&X[[6]][i,1] < all_CIs_MID[1,2] &&
       all_CIs_MID[2,1] < X[[6]][i,2]&&X[[6]][i,2] < all_CIs_MID[2,2] &&
       all_CIs_MID[3,1] < X[[6]][i,3]&&X[[6]][i,3] < all_CIs_MID[3,2] &&
       all_CIs_MID[4,1] < X[[6]][i,4]&&X[[6]][i,4] < all_CIs_MID[4,2] &&
       all_CIs_MID[5,1] < X[[6]][i,5]&&X[[6]][i,5] < all_CIs_MID[5,2] &&
       all_CIs_MID[6,1] < X[[6]][i,6]&&X[[6]][i,6] < all_CIs_MID[6,2] &&
       all_CIs_MID[7,1] < X[[6]][i,7]&&X[[6]][i,7] < all_CIs_MID[7,2] &&
       all_CIs_MID[8,1] < X[[6]][i,8]&&X[[6]][i,8] < all_CIs_MID[8,2]){
      coverage_MID <- coverage_MID + 1
    }
  }
  coverage_MID <- coverage_MID/total_MCMC_samples
  return(coverage_MID)
}
find_alpha <- function(bacnum, alpha = 0.05, tol = 0.0001, n_max = 100){
```

```r
  X <- readRDS(paste0("bac", bacnum, "results.RDS"))
  total_MCMC_samples <- length(X[[1]])
  # Bisection method. So find it at alpha/8, alpha/4, and alpha.
  # Then decide which half to use
  alpha_LB <- alpha
  alpha_MID <- alpha/4
  alpha_UB <- alpha/8
  coverage <- get_coverage(X, alpha_MID)
  conf <- 1-alpha
  j <- 1
  while(abs(coverage - conf) > tol & j <= n_max){
    j <- j + 1
    if(coverage - conf < 0){
      alpha_LB <- alpha_MID
      alpha_MID <- (alpha_LB + alpha_UB)/2
    }
    else if(coverage - conf > 0){
      alpha_UB <- alpha_MID
      alpha_MID <- (alpha_LB + alpha_UB)/2
    }
    else{
      print("Coverage is equal to conf level")
    }
    coverage <- get_coverage(X, alpha_MID)
  }
  return(list("alpha" = alpha_MID, "iterations" = j,
              "coverage" = coverage))
}
getBayesianpvalues <- function(testingUB = 1.5, numtags){
  pvalues <- vector("double", length = numtags)
  for(b in 1:numtags){
    MCMCoutput <- readRDS(paste0("bac", b, "results.RDS"))
    interval <- log(c(1/testingUB, testingUB))
    possibleCIsizes <- c(seq(.0001, .001, by = .0001),
                         seq(.002, .01, by = .001),
                         seq(.02, .2, by = .01),
                         seq(.3, 1, by = .1))
    for(i in 1:length(possibleCIsizes)){
      alpha <- find_alpha(b, alpha = possibleCIsizes[i])$alpha
      for(j in 1:8){
        # CI will contain the 100(1-alpha)% CIs for all 8 dimensions
        CI <- quantile(MCMCoutput[[6]][,j],
                  probs = c(alpha/2, 1 - alpha/2))
        # First check if endpoints of CI are between (1/tUB, tUB)
        # If CI LB between 1/tUB,tUB
        if((interval[1] < CI[1] && CI[1] < interval[2]) ||
           # or UB between 1/1.5,1.5
           (interval[1] < CI[2] && CI[2] < interval[2])){
        }# Then regions are not disjoint (do nothing)
        # Check also if (1/tUB,tUB) completely contained in CI
        # If CI LB < 1/tUB and tUB < CI UB
        else if(CI[1] < interval[1] && interval[2] < CI[2]){
        } # Then regions are not disjoint (do nothing)
        # Otherwise they are disjoint
        else{ # this is our p-value; stop heres
          pvalues[b] <- possibleCIsizes[i]
          break
        }
      }
      if(pvalues[b] != 1 && pvalues[b] != 0){
        break
      }
      if(i == length(possibleCIsizes)){
        pvalues[b] <- 1
      }
    }
    print(pvalues[b])
  }
  return(pvalues)
}

# Second Generation p-values
getsgpv <- function(numtags, K, nullinterval = log(c(1/1.5,1.5))){
    sgpv <- matrix(NA, ncol = K, nrow = numtags)
    nullwidth <- nullinterval[2] - nullinterval[1]
    for(i in 1:numtags){
     X <- readRDS(paste0("bac",i,"results.RDS"))
     for(j in 1:K){
        jointCI <- quantile(X[[6]][,j],
                            probs = c(alphas[i]/2, 1-alphas[i]/2))
        jointwidth <- jointCI[2] - jointCI[1]
        # First find |I and H_0|. If disjoint, overlap is 0
        if(jointCI[2] < nullinterval[1] || jointCI[1] > nullinterval[2]){
        overlap <- 0
        }
        # If null contains interval, overlap is width of interval
        else if(nullinterval[1] < jointCI[1] && jointCI[2] <
                nullinterval[2]){
         overlap <- jointwidth
         }
        # If interval contains null, overlap is width of null
        else if(jointCI[1] < nullinterval[1] && nullinterval[2] <
                jointCI[2]){
          overlap <- nullwidth
        }
        # If the overlap is not total but joint int below null, overlap
        # is upper limit of joint minus lower limit of null
        else if(jointCI[1] < nullinterval[1] && nullinterval[2] >
                jointCI[2]){
          overlap <- jointCI[2] - nullinterval15[1]
        }
        # If the overlap is not total but joint int above null, overlap
        # is upper limit of null minus lower limit of joint
        else if(nullinterval[1] < jointCI[1] && jointCI[2] >
                nullinterval[2]){
          overlap <- nullinterval15[2] - jointCI[1]
        }
        else{
          print("Error. Unanticipated overlapping.")
        }
        # If precise,
        if(jointwidth <= 2*nullwidth){
          sgpv[i,j] <- overlap/jointwidth
        }
        else{
          sgpv[i,j] <- (1/2)*overlap/nullwidth
        }
     }
    }
    return(sgpv)
}
```