5-4-2015

Classification and Multiple Hypothesis Testing in Microarray and RNA-Seq Experiments

Patrick B. Harrington
University of Connecticut - Storrs, PaddyBHarrington@gmail.com

Classification and Multiple Hypothesis Testing in Microarray and RNA-Seq Experiments

Patrick Harrington, Ph.D.

University of Connecticut, 2015

This thesis focuses on analyzing the type of data returned by two pieces of technology, the older and less expensive microarray, or the next generation sequencing data, RNA-Seq. Both devices return data that is extremely large in volume. Microarray analysis begins by finding genes of interest, which are called differentially expressed (DE). Genes are called DE controlling for some criteria, such as false discovery rate (FDR), and then clustered into groups. A method unifying these two steps was suggested, using a mixture of normal distributions with the appropriate EM algorithm. We compare this to a semi-parametric alternative to the unified method. We use simulation studies to compare these and other microarray analysis methods. We then look at next generation RNA-Seq data, with a focus on accounting for gene length. We introduce a hierarchical, log-linear negative binomial count model which incorporates gene length both into the parameter estimation and zero count inflation for this data. This hierarchical model allows borrowing counts information across genes efficiently and provides a Bayes factor criterion for screening for DE genes.We use real data to show a decrease in length bias when our method is compared to popular existing methods, as well as a simulation study to establish the effects of over and under fitting within our model, as well as the effect of fitting multiple DE types in a single model. We provide new methods for finding DE genes for microarray and RNA-Seq data, and illustrate their advantages using real and simulated data.

Classification and Multiple Hypothesis Testing in Microarray and RNA-Seq Experiments

Patrick Harrington

B.S., California Polytchnic State University, San Luis Obispo, California, 2007

A Dissertation

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

at the

University of Connecticut

2015

APPROVAL PAGE

Doctor of Philosophy Dissertation

Classification and Multiple Hypothesis Testing in Microarray and RNA-Seq Experiments

Presented by

Patrick Harrington, B.S.

Major Advisor _____
Lynn Kuo

Associate Advisor _____
Ming-Hui Chen

Associate Advisor _____
Zhiyi Chi

University of Connecticut

2015

To my darling son William Barry Harrington and my loving parents, Thomas and Karen Harrington

I am fortunate to have a great and loving set of friends and family in my life. None more supportive and loving than my parents, sister, and newborn. I would like to acknowledge each and every person who has helped me along my path to and beyond this current state, for the way in which they have helped me. Specifically, there are a handful of people to whom I am extremely indebted. I would like to thank Ming-Hui Chen for all of his help in so many different aspects of my studies. Beyond giving me enormous support throughout my entire time at UConn, he gave me attention and knowledge through the classes he taught me, the input he gave me on research, and consulting as well. Zhiyi Chi also gave me much support and help in classes, as well as great advice and direction outside of the classroom. Along with my advisor, these Professors have shown an amazing amount of patience and support, and been very accommodating. Without such great efforts, it is hard to imagine having made it through this program. And of course, by far the largest appreciation goes to my advisor, Lynn Kuo. The countless hours of support and help she has given me were absolutely the foundation on which this work is built. I know she must have endless support and patience, since she gave those to me. Without her direction and expertise in the areas of my thesis work, I would have been completely lost. Kind words could never properly express my feelings, and I hope my never ending gratitude will.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Genomic studies aim to understand the role genes play in our everyday lives. This involves numerous mappings of the human genome, numerous mappings of the genomes of other animals, and studies identifying the functionality of genes. Advances in technology and methodology have made genomic studies possible, and greatly increased the efficiency of such studies. One of the earlier, and still very popular pieces of technology is the microarray. Another popular, newer, and more expensive technology is based in RNA-Seq data, such as 454. In this thesis, we will cover methods currently used in the analysis of both microarray and RNA-Seq data.

In microarray experiments, the researcher typically has a pair of conditions, control and treatment, or a number of conditions, control, treatment 1, treatment 2, and so on for some number of treatments. The manner in which data is collected allows for the analysis of pairwise comparisons (i.e. difference in gene expression from one condition to another). In many cases where scientists develop and test a hypothesis, they know exactly what they are looking for. An example of this in the genomic setting would be the belief that a specific gene was involved in a specific process (like bone growth). There are also cases

in which we don't have such a clear idea of what to expect. An example of this would be the desire to see which of a large group of genes are involved in certain processes.

Microarray experiments can be used for both of these types of hypotheses. Microarray experiments are run using a piece of technology that has a lattice of addresses. Each well is treated with a solution that is comprised of genetic material. We usually associate that genetic material with exactly one gene. Multiple addresses from any one microarray have material representing the same gene, but cannot have material from more than a single gene. It is conventional to think of each well as a probe, typically given some identifier (Probe ID). This Probe ID is usually given by the software used for collecting and analyzing data, or the scientists running the experiment. Each Probe ID is linked to some official gene.

Within each microarray, the probes are put in the same order. While the conclusions from a microarray experiment are made with respect to the genes each probe represents, the actual analysis is run per probe. This means that the probes must be linked from microarray to microarray. This could mean using the exact same probe to well layout in each microarray, or simply tracking the manner in which this is permuted. Also, this means that the exact same probes must be placed on every single microarray.

In order to prepare a microarray, the genetic material needed for each probe is treated with the appropriate condition. There is often a control case, in which no treatment is made, although experiments can be run with a number of conditions all of which involve some treatment. In order to get gene expression (by probe), a measure that quantifies how much of a gene is present in certain conditions, a visual read is made on each well. In order for such a read to be possible, it is needed to introduce some dye. The reads themselves are made by machines. In order to account for various types of dependencies

and errors within these machine reads, the scientist makes both a background read and an actual read for each probe.

This requires two dyes, the most popular choices of which are Cy3 and Cy5. In order for a biological treatment to be prepared for each well, there is a treatment and quality control step that are performed. After this, the dye is applied. When getting gene expressions for each probe, the background dye and other dye are scanned in separately. The background read represents a baseline read for each probe. The other read represents the abundance of that specific gene within each treatment. The ratio (or logged ratio) of these reads is the quantitative value associated with each probe, typically referred to as gene expression.

In order to avoid poor reads, multiple reads are taken for both dyes for each probe. One value, such as the median of all such reads, is recorded for each dye in each probe. Another way to avoid (or account) for technical error is to flag probes for which the device believes there was a bad read. Before the data is analyzed, more pre-processing is done to identify which probes have bad readings in which arrays. This leads to a smaller number of probes being analyzed than are present in the microarrays. Another step that is taken to reduce error in the experiment is dye swap. The two dyes used can have different potencies, or other differences we attribute to dye effect. In order to capture this effect, the scientist can use a technique called dye swap. Here, the scientist runs a number of replicates for each condition, with some dedicated to one dye combination, and the others swapping the dye combination. This allows for both the estimation and incorporation of the dye effect.

Once each probe for each microarray has a quantitative value, gene expression, and once all needed preprocessing has been made, the goal of the analysis is to identify genes which act differently across the conditions. We call these genes differentially expressed

(DE), the alternative to the null classification of equivalently expressed (EE) genes. This thesis focuses on microarray experiments in which a group of genes are first to be classified as EE or DE, and then to be further clustered based on the expression data.

Microarrays have thousands and thousands of addresses. This combines with the flexibility of the device to allow the scientist to test thousands of genes simultaneously. It is possible (and conventional) to have multiple probes on each device dedicated to a single gene. It is possible to include genes which should show no change in expression across the conditions of interest. These serve as control genes, allowing the scientist to validate the results of the experiment.

When choosing which genes to include in the study, it is conventional to not only include genes that are expected to be involved in the processes being tested (i.e. acting differently in the treatment condition than the control), but also to include genes that are not expected to be involved in the process. This serves as a validation for the overall conclusion of the experiment.

There have been many methodologies suggested for finding DE genes in the microarray setting, a few of which are covered in this thesis. One popular approach, a linear modeling approach incorporating a Bayesian technique for borrowing across the genes, LIMMA (Smyth (2004)). Another very popular approach, Significance Analysis of Microarrays (SAM), was introduced in Tusher et al. (2001). This is a non-parametric alternative. Both of these methods give the user the ability to rank the genes with respect to their likelihoods of being DE, as well as offering p-values. Further, these methods allow the scientist to control for a predefined false discovery rate (FDR). Should the analyst use either of these, or any of the other methods for finding a set of DE genes, the next step is to try to find groupings within this set of DE genes.

This fits into clustering, or cluster analysis. Traditionally, cluster analysis is based on a distance metric. In this setting, the goal is to identify groups of like genes. The same expression data used to call genes DE is used to classify them. Clustering algorithms can be broken down into hierarchical and non-hierarchical. Hierarchical algorithms are either agglomerative, which initially considers every observation its own cluster and joins them together, or divisive, in which a single cluster is broken into many. A criterion, such as matched distances or a function thereof, can be recorded for each of the joining or dividing steps. Then, the number of clusters maximizing that criterion is chosen to be the best. Other very popular methods include k-means MacQueen (1967), partitioning around the mediod (PAM) Kaufman and Rousseeuw (1987), and EM algorithms Hartley (1958)based on some mixture distribution assumption.

One popular non-parametric clustering algorithm was developed in Kohonen (1982). The self organizing map (SOM) uses two spaces, a map space which is a topological graph, and the data space. In most cases, the data space is an m dimensional real space with Euclidean distance. The map is comprised of a lattice of nodes, the number of connections for each node is determined by the topology. Popular topologies include rectangular, with 4 adjacencies per node, and hexagonal, with 6 adjacencies per node. There is a separate distance function for the map space and the data space, although it is common to use euclidean distance for both. Each node in the map represents a vector in the data space. This vector is similar to the location of the node in the data space. The map needs to be initialized, by either setting the initial dimensions or growing the map, and then the map is smoothed. The smoothing step involves updating the map with the data in order to make the nodes of the map fit the data. The map can be further smoothed by using a neighborhood function. A neighborhood function (NF) is based on the distance function

in the map space, and assigns weights between two nodes, where the weight is decreasing in the distance. The NF is used to control how much smoothing occurs when training the map. One popular smoothing algorithm is called a batch algorithm. It updates each node based on the entire training dataset, as a batch. It converges quickly with respect to computational time, and based on an initial map, converges to a deterministic map.

It can be noticed that a set of assumptions is used to find DE genes. Then, another method, usually with its own set of assumptions, is used to cluster the DE genes into groups. That is why Yuan and Kendziorski (2006) actually integrated the two goals with one method, which they call the Unified Method. They state the methodology broadly. It is based on any mixture distribution assumption, where one of the mixands is set to be for the equivalently expressed genes. They develop machinery for ranking genes in being DE, as well as a method for estimating and controlling FDR. The paper then focuses on a mixture of normals assumption, where all of the parameters including the true number of mixands must be estimated. An EM algorithm is used to classify the genes. The approach suggests fitting an EM algorithm for cluster number ranging from 2 to some top value, and then some criterion such as AIC or BIC can be used to determine the true number of clusters.

We address some of the issues that can come with using an EM algorithm. It can be very computationally expensive to look at the results for the EM algorithm for clusters from 2 to some top value, especially when the true number of clusters is large. Also, the convergence of the algorithm isn't ensured in cases where the model is mis-specified, including when the true number of clusters is mis-specified. This means that whichever criterion is used to determine the true number of clusters can't be relied on unless the true number of clusters is specified. Even if the criterion has a good interpretation for

the mis-specified model, there isn't any assurance the values of the criterion are based on convergent states. It is completely possible for the criterion chosen to achieve a local maximum with respect to the criterion before the number of clusters considered is even close to the true number of clusters, and even if the true number of clusters is identified, it could take an enormous amount of computing resources to identify it. We address these issues using a specific application of SOM.

We use a growing SOM, with a batch smoothing algorithm to train a map with dimension determined by the data. We use a variant of a popular hierarchical clustering algorithm to join our map into regions, or clusters, based on Euclidean distance metrics in both the data and map spaces. We establish a methodology to find the true number of clusters based on the hierarchical results. We then use a similar method to rank the true number of clusters by our criterion, which can return a limited list of possible cluster numbers to try in the case of fitting an EM algorithm. We also save the initial states for each cluster in our results, and use them to initialize the EM algorithm, which leads to faster and more reliable results. This directly addresses the issues of the EM algorithm.

We then extend this method to classify DE genes. One problem with using a mixture distribution assumption is that each mixand must be focused on and estimated. In order to get reasonable estimates, sometimes the structure of the mixture must be overly restrictive. Also, some type of EM or some equivalent is needed to arrive at a solution. While some of the concerns for the EM have already been addressed, it would be nice to develop a mixture distribution that is less restrictive than more traditional methods, and has some methodology to call genes DE. With this in mind we develop a non-parametric approach for DE gene classification based on the SOM map. A truly non-parametric approach is

considered nice since it doesn't have restrictive assumptions, but especially for something as complex as classification, it can be difficult to estimate how well your classification is.

Due to this, we use one basic assumption on the mixture distribution. We assume that the null cluster, that is all the genes which are equivalently expressed, have changes from control to treatment centered at a change vector of zero, with a normal shape. We do not put any extra assumptions on DE portion of the data. In fact, one integral part of this is estimating the multivariate pdf of the data, which we accomplish within the SOM framework. This estimate allows us to ignore the form of the rest of the mixture distribution. The assumption put on the null cluster of genes is intuitive as it fits our notion of what an EE gene does (noise randomly placed around no change). We introduce an alternative to the original clustering approach which doesn't help to find the number of clusters in the data, or provide initial estimates for the EM algorithm, but instead focuses on joining nodes to the null cluster. A rule for when to stop joining nodes to the null cluster is used, and then the probability that a gene is EE given the data is estimated. These estimates are used to rank genes as DE, estimate the false discovery rate of a group of genes called DE, and control for a false discovery rate. LIMMA, SAM, the Unified Method, and our SOM methodology are compared in a number of specific setting using simulation studies. While our SOM methodology is outperformed by the existing methods when measured from false discovery rate (FDR), false non discover rate (FNDR), empirical type 1 and type2 errors, these situations are from the mixture of normal assumption that is most appropriate for all the methods. We then go through a number of examples of data simulated from mixtures which the mixture of normals EM algorithm is not expected to do well. We look at the same measures to see how successful our methodology performs.

Since the introduction of microarray technology, many advances have been made in the area of genomic studies. We turn our focus to next generation sequencing (NGS), which is considered second gneration RNA-Seq. RNA-Seq data is fairly new. Before any data can be collected, biological specimens are treated with a condition, sampled, and the RNA is sequenced. This process includes isolating RNA from the cells, fragmenting the RNA, copying the RNA into cDNA, and amplifying the fragments. Once these biological steps are completed, the data is sequenced into reads. These reads are strands of the RNA, and raw data has a form with sequence name, corresponding read, and in some cases a quality score for the read. There are a number of new devices being developed which return this type of data, with formats varying more so than with the somewhat standard microarray. Each device used to gather RNA-Seq data has a number of lanes. Each lane can be thought of as being similar to a microarray. That is, each lane can be given biological samples that have been treated differently (i.e. each lane corresponds to its own condition), each lane can be given replicates from samples treated identically (i.e. each lane corresponds to a replicate), lanes can be given a replicate within a condition, and so on.

The main difference in the preparation and resulting expression for each biological sample is in how counts are separated for genes. Like in microarray experiments, RNA-Seq experiments begin with biological samples gathered in different conditions. Unlike microarray experiments, the specific genes to be studied are not determined by the scientist, but instead the experiment. There are no addresses that get filled with a solution comprised of genetic material for a specific gene. Instead, each lane receives genetic material corresponding to all genes represented inside of the biological sample. This specifically rules out the ability to use null genes as validation for the results of the experiment.

Further, this limits a scientist's ability to focus on specific genes within a hypothesis. Transcriptome is defined as the set of all RNA within a cell at a given time. For each living thing, this changes over time. So technically the genes found within an experiment based on cells from a single living creature will change over time, meaning that two runs of the same experiment on a single organism could yield a different set of genes. However, one would expect for the trasncriptome analysis to yield similar results from run to run.

Instead of reading expression levels as illumination reads from individual addresses, each lane of biological material returns reads. These reads are RNA Sequences of a certain length. Depending on the technology being used, these reads can be of fixed or random length, and these lengths can be long or short. Length is measured in base pairs, and counts the number of nucleotide base pairs (typically represented by letters) in a read of RNA. An RNA-Seq experiment returns what is called a sequence library. This collection of all reads from a device is considered to be a snapshot in time of the trasncriptome of the biological sample. For properly executed experiments, the differences in these libraries from transcriptome to transcriptome are attributed to the difference in condition from collection of transcriptome to transcriptome. So, much like a scientist would look to quantify changes in gene expression in the microarray setting in order to identify genes which act differently across a set of conditions, one would look for differences in these libraries for RNA-Seq experiments. Unlike the microarray where each gene is identified via the Probe IDs for each well, the RNA-Seq data simply has a bunch of reads which can be mapped to the genome. The first level of difficulty is mapping each read to the desired genome and arriving at what is considered the raw data.

The typical steps for taking data from the reads making up a sequence library into raw count data is to assemble transcripts. Transcript assembly is usually taken care of by

aligning large sets of short reads to a genome. A very popular method for this is Bowtie (Langmead et al. (2009)). The goal of Bowtie is to take a large number of these shorter reads and align them with the genome. In order to understand how this is done, it is important to understand how the reads are made. The genetic material for any lane is sequenced. This involves reading the sequencing, which is typically done one nucleotide at a time. There are possible errors here, including reading a nucleotide which isn't actually there, reading a single nucleotide multiple times, and skipping nucleotides which are there. The result is always a read of nucleotides, so there isn't always a way to know the chance that any given read has any of these possible errors in it. When matching reads to the genome, the problem is in part allowing for these types of errors to be present in our reads while still mapping them to the genome. Another difficulty comes from the possibility of multiple matches. The shorter a read, the larger the expected number of matches. One way to minimize this problem is to use longer reads, but due to monetary restrictions and technological limitations, this may not always be a possible solution. Bowtie is constructed to deal with short reads, and is ultrafast and memory efficient. Bowtie is not a final step, but its output is used by other processing tools.

One such tool is Tophat. Tophat (Trapnell et al. (2009)) was developed to build on Bowtie. Bowtie relies on known splice junctions. Tophat doesn't need these predetermined splice junctions, and instead can find novel splice junctions based on the data. When introduced, this method had great results in finding both previously known splice junctions as well as a large number of novel ones. Additionally, the steps can be run at a very high level of efficiency. When completed, tophat returns some reads that can be mapped to the genome, and others which cannot. MapSplice (Wang et al. (2010)) was introduced as

an extension to Tophat. It focuses on high accuracy (sensitivity and specificity) in the detection of splice junctions and computational efficiency.

Tophat remains quite popular for a number of reasons. One of these is that the output from tophat can be taken directly in by Cuff Links. Cuff Links is a part of a system created for the purpose of analyzing RNA-Seq data. Trapnell et al. (2010) introduced Cuff Links to build on the tophat method. It can take the alignments output by Tophat, and assemble them into transcripts. Transcripts are assembled reads. These reads are mapped over the same region of the genome. For each transcript, we can track the number of reads mapped to that transcript, and the length of the transcript in addition to the location of the transcript on the genome. It is important to notice that reads can be aligned to multiple locations on the genome. This means that reads can be mapped to multiple transcripts. Since the goal of running the experiment is to eventually analyze multiple libraries, it is important to get count data from these raw reads. This requires estimating counts for each of these transcripts. These estimated counts and lengths are returned by Cuff Links. It should be noted that these estimated counts are not actually counts, and depending on how the data is to be analyzed, some rule for converting them into counts may need to be used. Once the data has been processed by tophat, and then by Cuff Links, the current state of the data is to have mapped reads to the genome, and then to transcripts. The next step is to take the transcripts and map them to regions in the genome. This is again done by Cuff Links. When this step has been completed, the output still isn't in the form needed for analysis. One file should be used to describe each lane of the device. These files have a number of transcripts, described by estimated count and overall transcript length. The analyst must at this point combine all transcripts within each lane into a single count and length estimate for each gene. It should be noted that

bar codes, or identifiable RNA sequences, can be placed on reads from different biological samples in cases where the scientist wishes to use a complex design for the experiment. This would allow for multiple values to be tracked per gene, breaking each estimated count and length down by bar code. For this thesis, we didn't explore any data of this type, but the processing requires just one more step to get the count data needed to analyze the experiment. Of course, this more complex setting will require more complex methods of analysis. The final processed form of the data is arrived at by joining these files for each lane together. The result is a single dataset.

The union of all of the genes for each lane make up the list of all genes represented in the study. The final dataset has one row for each of these genes. Genes which are in this list need not be present in every lane. Each gene in the dataset has a value for gene length in each lane, and estimated count in each lane. For lanes in which a gene is not present, the value of 0 is recorded for both length and estimated count. This is typically referred to as the raw data for any RNA-Seq experiment. Before analysis, however, many scientists will choose to use one more processing step. This is to identify genes which are not present in many of the lanes. The idea here is that Bowtie and Tophat could well improperly assign reads to portions of the genome. This leads to genes not present in the study to be observed in the sequenced results, although not necessarily in every lane. Genes which are not present in the study while being observed in the sequenced lanes tend to be present in a smaller number of lanes than any of the genes actually present in the study. The scientist analyzing the data must create a rule for omitting genes from this final list which are likely present only due to technical error. The conventional approach is to omit genes for which there are a large number of zero counts, since that is equivalent to not being present in a large number of the lanes. It is with this description in mind

that some methods for analyzing this data use special consideration for zero counts, as does the methodology we suggest.

For RNA-Seq, the steps of aligning reads to a genome, assembling those reads into transcripts, and mapping those transcripts to a reference genome need not be accomplished using the specific methods listed here. We specify the methods used by us in preparing our read data for analysis, but others have used entirely different methods for each of the steps. Should a scientist wish to analyze RNA-Seq data, however, these steps will need to be accomplished with some method. One reason these specific steps were chosen is that they are all made available through psu.galaxy.org (Goecks et al. (2010), Blankenberg et al. (2010), Giardine et al. (2005)). In fact, for many steps needed to simply get the data in the proper format for different tools and many extensions to steps taken in this paper can be found on galaxy. It offers comprehensive documentation and a relatively friendly community to ask questions and collaborate with. Beyond being free to use and offering a large amount of storage on a cloud computing foundation, galaxy offers the user a single place and tool to perform each step and pass the data along carefully constructed work flows.

With the raw count data in hand, there are a number of methods which can be used to analyze the data. Out data falls into a control, or treatment case. The methodology we list here considers that case as well. Count data assumptions typically use a Poisson, binomial, or a negative binomial distribution to model the data. Such methods include EdgeR (Smyth and Robinson (2007b), Smyth and Robinson (2007a)) and DESeq (Anders and Huber (2010)). In (Smyth and Robinson (2007b)) Smyth covers the approaches to analysis predating his. Of all the distributional assumptions, his works is based on the negative binomial model. He uses a parameterization that allows for a mean and

overdispersion parameter. He noticed that the total count of reads per lane, which he calls library size, differs. The differences in counts from lane to lane could be due to the library size, as opposed to condition. To that end, he allowed the mean for each gene in a given lane to be proportional to the library size, and said that there was a condition specific rate. The null vs alternative hypothesis for each gene in this setting is equivalent to testing to see if the control rate is equal to the treatment rate. In this methodology, a Wald type statistic is created by estimating the true control and treatment rates, and the variance of those estimates. In cases where the sample size is too small to warrant using the Wald type statistic, an exact test is available.

Length bias in large part motivated the RNA-Seq work in this paper. It has been noted empirically that in general, there is a positive relationship between the length of a gene and the chance that gene is called DE. We explore gene length in depth. We look at the current way of calculating gene length, and suggest a different consideration, a condition-specific gene length. This consideration is motivated by an explanation for why genes with a longer length is more likely to be called DE. Our belief is that there are a group of genes which have different regions activated from control to treatment. This would mean that transcripts from the control setting are mapped to different regions of a gene than those from the treatment setting. One way to conclude that transcripts for a gene are mapped to different regions of the genome in different settings is to notice that they have significantly different gene lengths. Instead of simply trying to find genes with different condition-specific lengths, we find genes where the change in observed values can be explained by the different condition-specific lengths. In addition to modeling condition-specific gene length, we incorporate gene length into our model, while also accounting for library size. We introduce a hierarchical linear modeling approach for estimating the log

of the true mean counts for a gene in a given lane, equipped with a method for calling genes DE. Our approach also uses an intuitive approach for accounting for zero counts which is also based on the gene length. This method is flexible enough to identify 3 different types of differentially expressed genes, and the results of applying it to real data are compared with EdgeR to show a reduction in length bias. We also use a simulation study to further understand the possible effects of over/under fitting, as well as to see if there are multicollinearity concerns with fitting more than one type of DE gene in a model.

As a general note, we refer to simulation studies for both the microarray and RNA-Seq sections of this thesis. Part of the computation was done on the Beowulf cluster of the Department of Statistics, University of Connecticut, partially financed by the NSF SCREMS (Scientific Computing Research Environments for the Mathematical Sciences) grant number 0723557.

Chapter 2

Microarray

Microarrays give us the ability to observe the expression levels of tens of thousands of genes simultaneously in both treatment and control scenarios. Further, they can be used to see how gene expression levels change over time. This allows us to explore large groups of genes of interest, both for known and unknown biological properties. In order to find genes of interest within the group of genes studied, the first step is typically to identify differentially expressed (DE) genes. These genes are found using some assumptions on the numeric values representing their expression levels at each treatment level considered. One very popular method for finding DE genes is Linear Models for Microarrays (LIMMA) as presented by Smyth (2004). In this method, linear modeling is utilized along with a set of prior distributions for the regression coefficients and variance. Another popular method is Significance Analysis of Microarrays (SAM), as presented by Tusher et al. (2001). This method doesn't rely on the parametric assumptions as in the case of LIMMA. Instead, it finds a gene specific t-statistic using a gene specific standard deviation and a common standard deviation correction. Based on these values, a group of potentially differentially expresse genes are identified. SAM focuses on controlling a false discovery rate (FDR),

and uses permutations of the measurements to assess the FDR of the group of genes. However, in cases where parametric assumptions are approximately true, this method will be outperformed by some methods with appropriate assumptions.

While originally designed to simply find DE genes without considering time course data, both of the above methods have been adapted and are used in time course settings. We wish to consider methods created for the sole purpose of time course. Tai and Speed (2006) offer a Multivariate Emprical Bayes Statistic (MB-statistic) for ranking genes in a time course setting. We will outline how to deal with this later.

While the previous papers talk about methods for identifying genes which are differentially expressed, none have addressed the issue of identifying a latent structure where similar genes have similar expressions. In many, if not all microarray experiments, after identifying the DE genes, the scientists perform some clustering method. The idea being that they can find genes which have similar behavior and characteristics based on similarities in their expressions. Whereas the previous methods necessitate a separate clustering step, which will have their own set of assumptions not necessarily paralleling the method of identifying DE genes. Because of this, identifying some sense of correctness with respect to these groupings based on a clustering method is difficult, if not impossible. One method proposed by Yuan and Kendziorski (2006) actually integrates the two goals mentioned earlier. It is for this reason that we will refer to this method as the unified method, or unified approach.

Yuan and Kendziorski made an assumption that there is some latent structure for the genes, one which coincides exactly with the assumptions leading to clustering after DE genes are identified. This assumption is manifested in the form of a mixture distribution, where the parametric form of the mixands is assumed to be known, and all parameters

to be unknown. A simple example of this is a mixture of normal distributions with K mixands, where K is known, but the mixture proportions, means, and standard deviations of mixands are unknown. In order to analyze the data, the authors suggested using a very popular tool, the expectation maximization (EM) algorithm. Earlier refferences for this algorithm include Hartley (1958) and Dempster et al. (1977).

This paper explores the performance of all of the methods in a series of different settings. Since many of these methods do not accomplish identifying the underlying latent structure of the genes, we will focus on characteristics such as false discovery rate (FDR) and false non-discovery rate (FNDR). This will be accomplished via a simulation study. We will also explore different rejection rules where appropriate, trying to identify the one yielding the best results. Finally, we will continue by exploring the characteristics of the method suggested by Yuan and Kendziorski , with appropriate extensions, with respect to identifying that underlying latent structure.

This unified method has many benefits. The approach suggested for classifying the genes and identifying the latent structure is an EM algorithm based on a mixture of normal distributions. This is a common assumption for microarray data, and has been used in exploring the accuracy of a number of different methods. The EM algorithm does have some shortcomings and possible pitfalls. In order for an instance of the EM to be run, the true number of clusters within the data must be known. Typically this requires the algorithm fit the data with a varying number of clusters. Based on the results across clusters, the optimal cluster number is chosen. This can be very time consuming. Also, for incorrectly specified models (i.e. incorrect cluster number), the convergence of the EM is harder to achieve. This can make it difficult to accurately identify the number of clusters from the multiple runs of the algorithm. Finally, it is always nice to have non-parametric

extensions/alternatives to parametric approaches. We develop a methodology built around a very popular clustering method, self organizing map (SOM) Kohonen (1982), which can address these issues by estimating the number of clusters within a dataset, estimating the initial parameters for the mixture of normal assumption, and calling genes DE based on a non-parametric approach, as well as a semi-parametric method. It is worth noting that the semi-parametric method can be extended to mixtures of distributions which are not normal. It offers an unsupervised learning approach to clustering data into a topology in a non-parametric manner. We compare the mentioned conventional methods for classifying genes as DE with the unified method and our non-parametric and semi-parametric methods using a number of simulations.

We propose a new method for applying SOM to microarray data. There are different implementations of SOM, ours can be broken down into steps. We first use a growing self organizing map (GSOM) to simultaneously grow the map based on the variance in our data and smooth the map. Then, we use a batch algorithm to update the nodes of our map in a smoothing step. We then cluster the nodes of our map together. Our contributions are made first by applying this process to find the true number of clusters in a dataset and initial parameters for those clusters. This aids in the EM application and offers a non-parametric clustering alternative to that of model based approaches such as the EM. We then propose an alternative to existing clustering in the SOM framework. This alternative is motivated by the microarray setting, specifically the interpretation of EE genes. We then specify two methods, one non-parametric and one semi-parametric, for classifying genes as DE based on the output from our SOM implementation. This step requires that we use a clustering result, so technically we offer 4 approaches. These 4 approaches mimic the unified method proposed by Yuan and Kendziorski, meaning they

simultaneously cluster the data and identify DE genes. While they will be outperformed by the unified method in general for cases where the assumptions of the unified method are met, our approaches offer the advantage of taking less computational time and having less restrictive assumptions. We use simulations based on a mixture of normals to establish the performance of our method alongside other methods for calling genes DE in microarray data. T We then use data simulated outside of the assumptions of the unified method in order to establish the benefits of our methodology when compared to the unified method. While the results of these simulations shows that the unified method typically outperforms our non and semi parametric approaches when the assumptions of the unified method are met, we show that our method can be applied successfully to the data. We also establish that our methodology can be successfully applied to data where the unified method cannot.

2.1 Data

In order to understand the distribution, we must first understand the form of our data. We have a total of G genes. Across all of these genes, we are looking at some number, m, of points of interest. That is, m different conditions, such as time points for time series data, all of which take some value. In the case of time series, we could think of gene expression collected at $m+1$ time points. The levels of interest in this case would be the contrasts for each adjacent time point, representing the m jumps observed. This being the case, we have a vector, $\boldsymbol{X_i}$, which describes the i^{th} gene expressions with these m dimensions. Since we focus on time course data for this paper, we consider the original data to have the form $\boldsymbol{X_1'}, ..., \boldsymbol{X_G'}$ where $\boldsymbol{X_i'} = (X'_{i,1}, ..., X'_{i,m+1})$. We are interested in the gene expression changes over time, i.e. $X_{i,j} = X'_{i,j+1} - X'_{i,j}$, where $\boldsymbol{X}_i = (X_{i,1}, ..., X_{i,m})$. So we will consider mixture distributions for gene expressions of m dimensions. When

we have replicated data, say n_0 replicates, then we will use $\boldsymbol{X}_{i,1}, ..., \boldsymbol{X}_{i,n_0}$ describing the replicates for the i^{th} gene.

2.2 Assumption

We assume that the gene expressions considered in the microarray experiment themselves follow a mixture distribution. We assume that the mixands are themselves continuous. These mixands can be described by a mean and a variance structure. We assume there are K overall groups described by our latent structure, making the pdf for the vector of expressions of any one gene:

$$f(\boldsymbol{x}) = \sum_{j=1}^{K} p_j f_j(\boldsymbol{x}) \tag{1}$$

Without putting any additional assumptions on the form for f_j, 1 isn't a parametric assumption, it simply describes groupings for data. We argue that putting an assumption on a single mixand, such as the first mixand isn't a complete parameterization. We call the case semi-parametric where f_1 is assumed to be a normal pdf with some true mean and variance, but f_j is considered unknown and without restriction past being a bona fide density function for all $j > 1$. We can assume there is a latent variable, Y which is unobservable yet tells us to which mixand the observed gene expression belongs with $f_{\boldsymbol{x}|y}(\boldsymbol{x}) = f_y(\boldsymbol{x})$ and $g(y) = p_y$. That is, for any gene, should we know the cluster to which it belongs, we know exactly the pdf for the expression vector. Note we omitted the both the indices for gene i and its replicate for simplicity here.

2.3 Inference

We plan to cover the way in which we perform our inference, but first look at some functions of the data based on the true parameters. We have currently suggested that K true groupings of the data exist. We will fix the first group as being EE, in most cases this means that $\boldsymbol{\mu}_1 = \mathbf{0}$. Our inference will be first to identify genes very likely NOT from this null group. The second step is to find the subgroup of those which seem to belong to any of the other groups based on gene expression, suggesting strong similarities amongst these groups of genes. Typically, almost of as much interest as the classification of a gene as either EE or DE is the understanding of the pattern. We wish to perform classification, with a focus on calling genes DE. Without any other knowledge, we can say that our observation is from any given cluster with probability equal to that mixand proportion. This can be used to find the probability that any gene comes from the k^{th} cluster given its expression via

$$f_{y|\boldsymbol{x}}(k|\boldsymbol{x}) = \frac{f_{\boldsymbol{x},y}(\boldsymbol{x},k)}{f_{\boldsymbol{x}}(\boldsymbol{x})} = \frac{f_{\boldsymbol{x}|y=k}(\boldsymbol{x})p_k}{\sum_{j=1}^{K} p_j f_j(\boldsymbol{x})} = \frac{f_k(\boldsymbol{x})p_k}{\sum_{j=1}^{K} p_j f_j(\boldsymbol{x})}$$

2.3.1 Density Estimation

In some of the inference for this section, we will need to use a density estimator. We offer a brief overview of density estimation as a precursor here. At its simplest, density estimation can be thought of as creating a histogram. In order to see this, one simply needs to apply the definition of a derivative to the CDF:

$$f(x) = \lim_{h \to 0} \frac{F(x+h) - F(x)}{h} \tag{2}$$

We can think of a histogram with fixed width h, and see that as our bin number increases, the height of the bin $(F(x) - F(x-h))$ divided by the bin width will become a good approximation for the true pdf at any point within the bin. The duality of density estimation is that as our bin width decreases, the height of the bin converges down to zero. These problems are compounded when moving from univariate to multivariate data. Here, we extend the concept of binning to using a multidimensional mesh. Should each of the D dimensions of the data be split with an equal number of bins, n_b, this leaves n_b^D total bins, which will be sparsely populated by the data as D or n_b increases. This is one reason why statisticians have developed an alternative to the histogram or bin type density estimator. We expand on this by looking at a couple of examples of density estimators.

A very popular variant of a bin type density estimator is the average shifted histogram method (ASH). A very popular alternative to the bin type estimator is kernel density estimation. Kernel density estimation is often credited to Rosenblatt (1956) and Parzen (1962). Since it's proposal, there have been many papers establishing convergence properties. The form of the estimate is

$$\hat{f}_h(x) = (nh^d)^{-1} \sum_{j=1}^{N} K\left(\frac{x - x_j}{h}\right) \tag{3}$$

The typical kernel used is that of a normal random variable, $K(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$, which belongs to a class of kernel functions which is shown to yield a strongly consistent density estimate Devroye and Penrod (1984). As pointed out in Izenman (1991), the performance of this density estimator is greatly dependent on the value h. However, work has been done on identifying the optimal value for h.

Across two articles, Scott and Thompson (1983) and Scott (1985a), the foundation for ASH was made. The basic idea of this density estimator is to use the average of shifted histograms. As with the previous density estimator, we must choose a window, or bin size, in order to estimate our density. While kernel density estimation allows you to estimate the density for any given point directly, in the case of ASH you must use some interpolation technique. The approach Scott took for this was to use frequency polygon, Scott (1985b), looking at the properties of the estimator when points in between bin endpoints are estimated by connecting the estimates at the bin with a straight line.

2.3.2 Semi-Parameteric

Here we first lay out an explanation of SOM. We give an overview, examples of SOM in microarray, and detail each step for our implementation of SOM. For each step, there is a general description, psuedo code, and specific run parameters (functions, values fed in, so on). Should there be anymore questions about the specific settings for our SOM implementation, the R code written to accomplish these tasks is included in the appendix.

2.3.2.1 Overview Of SOM Algorithm

In order to give the reader a constant reference throughout, we will be using a SOM algorithm on a simulated, 2 dimensional dataset. We use the following mixture to simulate 50,000 observations:

$$\begin{aligned} &.5N\left(\begin{pmatrix}0\\0\end{pmatrix},\begin{pmatrix}1/4 & 0\\0 & 1/4\end{pmatrix}\right)+.1N\left(\begin{pmatrix}-1\\1\end{pmatrix},\begin{pmatrix}1/4 & 0\\0 & 1/4\end{pmatrix}\right)\\ &+.2N\left(\begin{pmatrix}-1\\-2\end{pmatrix},\begin{pmatrix}1/4 & 0\\0 & 1/4\end{pmatrix}\right)+.2N\left(\begin{pmatrix}1\\2\end{pmatrix},\begin{pmatrix}1/4 & 0\\0 & 1/4\end{pmatrix}\right)\end{aligned} \tag{4}$$

For the substeps of the algorithm, we offer pseudo code to help explaining the algorithm. The SOM algorithm uses two spaces, a map space and a data space. The map space is a graph connected by a specified topology. In this setting, we have a number of nodes which fall along a lattice. Each node has a set of adjacent nodes. The set is determined by the topology. If I use a rectangular topology, this means that each node has 4 possible adjacencies. If I use a hexagonal topology, this means that each node has 6 possible adjacencies. These nodes can be considered to fall along an x-y axis, and each node is equally spaced from each of its adjacencies in that x-y plane with respect to the Euclidean distance. This means that the x-y coordinate for each node can be determined, and will differ from topology to topology. While hexagonal topologies have become very popular, in large part due to how nice the graphics look, we opt to use a rectangular topology due to the ease of computation.

Once a topology is chosen, a distance metric must be chosen to determine how far apart two observations are with respect to the data. In this case, we choose to use the conventional Euclidean distance. Each observation has one quantitative value for each dimension of the data. We think of $\boldsymbol{X}_g = (X_{g1}, ..., X_{gm})$, where for any two observations $\boldsymbol{X}_{g_1}, \boldsymbol{X}_{g_2}$, the distance between them is $\sqrt{\sum_{j=1}^{m}(X_{g_1 j} - X_{g_2 j})^2}$. Just like each observation has a numerical value for each dimension of the data, each node in the map has a numerical value for each dimension of the data. We next choose to define the distance in our map space to again be Euclidean. Each node has two sets of values to consider. For the i^{th} node, we use $\boldsymbol{NM}_i = \{NM_{xi}, NM_{yi}\}$ to denote the $x - y$ coordinates of the node in the mapspace, while we use $\boldsymbol{ND}_i$ to denote the value of the node in the dataspace. We define our distance metric within our map space to be Euclidean as well. For the i^{th} and j^{th} nodes, the distance in the map space is $\sqrt{(NM_{xi} - NM_{xj})^2 + (NM_{yi} - NM_{yj})^2}$.

When the map has been trained, the result should be that nodes which are close in the map space are also close in the data space. Each observation can be mapped to exactly one node, called the best matching unit (BMU). This BMU is the node which minimizes the distance in the dataspace. One clustering to consider is that each node represents a cluster containing all of the observations mapped to it (by BMU). Data is normalized for the SOM. Normalization in this sense means to transform each variable to a scale from zero to one.

2.3.2.2 SOM In Microarray

SOM has been applied to microarray data many times in the past. Overwhelmingly, the most useful applications are to finding groupings for the DE genes found in some analysis. Sturn et al. (2002) is a nice paper which looks into applying many clustering methods to microarray data, including SOM. Toronena et al. (1999) wrote a paper looking at applying SOM, with a predetermined map dimension of 16, to a set of microarray data. The paper touched on how well the approach can summarize and explain the genes based on the expression data. A paper which focuses on the visual aspect of SOM, by integrating it with component plane presentation, Xiao et al. (2003) illustrates how powerful the SOM methodology is at highlighting trends within the data that have biological interpretation.

There are applications of SOM in microarray experiments which looks to do more than simply cluster the data. One of the first such applications of SOM can be found in Golub et al. (1999). This paper uses SOM for class discovery. The basic setup is that you have two classes of sample, in this case a healthy class and a cancer class. The method takes a training set of microarray's, each with known state. The researcher finds genes, in other research called marker genes, which are accutely different across the two classes. SOM was

applied to find which genes were close in expression to these genes. Based on these results, a new microarray representing a sample from unknown class can be classified into exactly one class. This SOM based method also returns a prediction strength, a vlue which varies from 0 to 1. This is one of the earlier papers on this topic, and led to many extensions and tests which can use a person's genetic material to classify them as having cancer or not.

In one such extension, Hsu et al. (2003) applies a growing self-organizing map (GSOM) just like ours to two classes of data. The experiment involved diseased vs. non-diseased biological samples. Using a hierarchical clustering method based on multiple GSOM outputs, the authors performed clustering on the data. The method identifies marker genes, which are genes identified by SOM which best distinguish the two classes of samples. This paper found high accuracy in classification. A similar paper focusing on class prediction is Covell et al. (2003). This method uses a slightly different SOM based approach to use microarray output to classify the underlying sample. Again, high prediction accuracy here is achieved. We notice the difference between the goals of this methodology, finding genes that distinguish classes, and our goals, findings groupings of the genes based on their expression and identifying genes which are DE accross a set of conditions.

2.3.2.3 Grow Step

Just like the EM algorithm requires an initial state, our SOM requires an initial map. This requires the specification of the number and placement of nodes, as well as initial values in the data space. Some implementations of SOM require the true number of clusters to match the number of nodes in the map. Our implementation will not be so restrictive. Instead, we wish the map to represent how spread apart the data are. Because

of this, we use a growing step to generate the initial map. We restrict the type of map to rectangular, meaning that we choose the number of rows and columns in our lattice, giving our map a rectangular shape. This differs from a non-rectangular map where not every x-y pair within our lattice has a node. Also, should we have chosen a different topology, the maps may not look perfectly rectangular, but are called rectangular when the lattice is comprised of a number of rows and columns.

The grow step shuffles the order of the data, and samples four points to create a 2X2 map. The user specifies a Spread Factor (SF) between 0 and 1. A Growth Threshold (GT) is then calculated to be $GT = -\log(SF) * m$. Growth Threshold acts as a minimum error requirement to grow the existing map within the grow step. So larger values of GT lead to less growing. This means GT decreases as SF increases, meaning that spread factor values close to 0 give smaller maps, and close to 1 creates larger maps. The BMU for any observation is the node on the map which minimizes the distance in the data space. In the grow step, observations are sampled without replacement from the data. Each observation is mapped to its BMU, and the distance is recorded. Each time a node on the map is a BMU, we add the distance from that observation. When the distance exceeds the GT, we spread the map. If the node which exceeds the GT is on the border of the map, we add a new row and/or column, depending on which border the node is on. If the node is in the interior of the map, we spread the value of the node to the adjacent nodes. Choice of SF and the natural variability in the data determine the map size. The pseudo-code for this can be found in figure 19. Also, sample output has been generated in figure 2. There is one plot for each dimension of our example data. The x-y location in the plot is determined by the map topology. The color of each dot represent the quantitative values

Figure 1: Pseudo-code: SOM Grow

procedure SOM GROW
    Initialize a 2X2 mapped
    Set node distances to 0
    Shuffle data order, set data counter $g \leftarrow 1$
top:
    Current observation $\boldsymbol{X}_g$
    Map current observation to BMU
    Update BMU and neighboring nodes with UpdateSP (See below).
    Add distance to BMU node.
    if distance> GT **then**
        if BMU is border **then**
            If right border, add column to right
            If left border, add column to left
            If upper border, add row to top
            If bottom border, add row to bottom
            Reset all node distances to 0
        else
            Spread distance and node values to adjacent nodes
Loop
    if $g < G$ **then**
        $g \leftarrow g + 1$
        goto *top*
    else
        END
procedure UPDATESP
    $\boldsymbol{ND}_{BMU}^{NEW} = \boldsymbol{ND}_{BMU}^{OLD} * (1 - L_0) + \boldsymbol{X}_g * L_0$
    Find the set of all adjacencies to the BMU, S_A
    for each $i_A \in S_A$ **do**
        Update $\boldsymbol{ND}_{i_A}^{NEW} = \boldsymbol{ND}_{i_A}^{OLD} * (1 - L_0) + \boldsymbol{X}_g * L_0$

for the variable, green corresponding to small values and red corresponding to large values.

We specify that when the map is grown, each new node has exactly one existing node which is adjacent to it. Therefore, the data vector for any new node ND_i created when using the g^{th} observation from the dataset, we have values $(\boldsymbol{X}_g + ND_{i_A})/2$ where i_A indexes the lone existing adjacent node to the i^{th} node. We use $L_0 = .1$ for the updating, or smoothing, within the grow step. We found that our method gives the best results

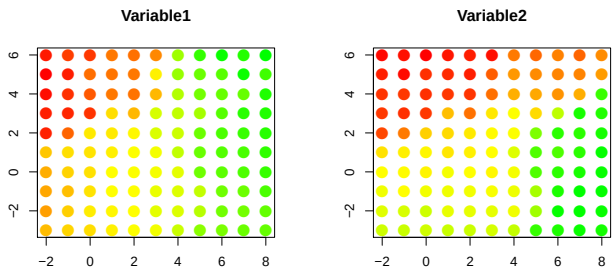


Figure 2: SOM output by variable:Initial graph after grow and before smooth steps. Each Plot represents a variable. Each dot inside of the plots corresponds to a node location in the map space. The colors represent the value for the variable at the node. Green values are smaller and red values are larger

when using a large SF. This will effect both the accuracy of our density estimator (to be covered shortly) and the clustering results. We suggest using a SF of .9, .95, or .975.

2.3.2.4 Smooth Step

Once a starting map has been grown from the data, we move to smoothing. There are different schemes for smoothing, but they all have the same foundation. Observations are mapped to their BMU, and they update the value of the BMU by some weighted average of the current value, and the observation's value. Then, a neighborhood function is applied. This function allows the observation to update all nodes within a neighborhood (in the map space) of the BMU. The weight in the weighted average assigned to the observation is smaller for nodes which are farther from the BMU in the map space. For some implementations, there is a distance in the map space past which the observation doesn't update any nodes in the map. We use such a concept in our method, and refer to this as tension. Some implementations allow the neighborhood function or overall weight assigned to the observation in these weighted averages to decrease each time an observation is mapped to a BMU, eventually converging (no longer changing the map). At this point, we say the map is smoothed or trained. A popular method used for the smoothing is called a batch algorithm. The batch process begins with assigning every observation to it's BMU in the map. Then, each node is updated one at a time. Every observation is used to update every node. The weight of an observation in the final calculation of the node value is larger for observations mapped to nodes closer to the node being updated. Once every node value is updated in this manner, the overall difference in the map is calculated, and if it is too large, the step is repeated. This algorithm converges to a single, deterministic map given an initial map, and converges quickly relative to some other conventional implementations.

Figure 3: Pseudo-code: SOM Smooth

procedure SOM SMOOTH
topSmooth:
    Find BMU for all observations from the data ($BMU_g \in \{1, 2, ..., n_{map}\}$)
    $i \leftarrow 1$
    Error Converge Track set equal to twice Error Converge Set, $ECT = 2 * ECS$
topNode:
    Set node to node i
    $g \leftarrow 1$
topGene:
    Calculate the distance from BMU_g to the current node in the map space, call $dist_g$
    Calculate $w_g = NF(dist_g)$, the weight for the BMU of the observation and the current node
LoopGene
    if $g < G$ **then**
        $g \leftarrow g + 1$
        goto *topGene*
    elseContinue to *LoopNode*
LoopNode
    if $i < n_{map}$ **then**
        Update current node with $\sum_{g=1}^{G} w_g * \boldsymbol{x_g}$
        $i \leftarrow i + 1$
        goto *topNode*
    elsegoto *LoopSmooth*
LoopSmooth
    Calculate the difference between original and updated map
    Set Current Map to Updated map
    if $ECT \geq ECS$ **then**
        goto *topSmooth*
    else *END*

For notation, we use $NF()$ to denote the neighborhood function. This function maps the distance between two nodes to a weight, in a monotonically non-increasing manner. That is, the weight assigned to nodes which are farther apart, is less. The pseudo-code can be found in figure 3. The sample output can be found in figure 4

We specify that for our implementation, we use

$$NF(d) = 1/(d+1)\mathrm{I}(d/d_{max} \leq T) \tag{5}$$

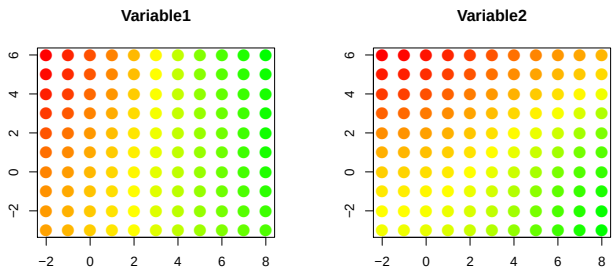


Figure 4: SOM output by Variable: Initial graph after the smooth step. Each Plot represents a variable. Each dot inside of the plots corresponds to a node location in the map space. The colors represent the value for the variable at the node. Green values are smaller and red values are larger

where $T \in (0, 1)$ is the above mentioned tension, and d_{max} is the maximum distance within the map. It is clear to see that a tension of 1 uses the entire data to smooth each node, with the weights decreasing as a function of distance in the map space, and a tension of 0 uses only those observations mapped to the current node to update ND_i. It should be noticed that using a tension too close to 0 can be problematic from a computational standpoint since nodes can be empty, meaning that it is possible for nodes to not have a single observation mapped to them (via BMU). Another consideration with tension is that it effects the clustering results. Higher tensions lead to smoother maps. We use a tension of .2 in our implementation, and suggest using tension between .2 and .5 when using our methodology.

2.3.2.5 Density Estimator Using SOM

In the following steps, we need to estimate the multivariate pdf for any of our observations $\boldsymbol{x}$, $f_x(\boldsymbol{x})$. Among the most popular multivariate, non-parametric density estimation techniques is kernel density estimation. While there are other non-parametric ways of accomplishing this, we choose to use the MAP resulting from our Grow and Smooth steps in order to estimate this function non-parametrically. In keeping with the theme of a unified, non/semi-parametric approach, this is integral to our approaches in applying SOM to both cluster and classify the DE genes in the data.

SOM most closely mimics a bin based density estimator. One difficulty of bin based multivariate density estimation is determining a mesh to apply to the data. Equally spaced bins have the problem of blowing up with the dimension of the data. If we have only 4 dimensions, and create 10 bins per dimension, we end up with 10^4 bins. Also, many of these could be empty, and an extremely large dataset is needed to perform any

useful inference in even this very simple case. The bin size also effects how accurate our estimator is. One needs only apply the definition of a derivative to the CDF to find $f(x) = \lim_{h \to 0} \frac{F(x+h)-F(x)}{h}$. Here, we use a univariate example for simplicity, and notice that our bin size determines h.

The SOM arrives at a set of nodes, exactly one to which each observation is mapped. In that sense, we can think of each node as being a bin. Estimating the size of these bins is not as simple as the case of a mesh, where there are clear upper and lower points. In the case of a mesh, we can think that each bin has a lower and upper point based on the bins adjacent to it. Extending this to the SOM, we can estimate the distances of a given node for each dimension of the data by looking at those values in the nodes adjacent to that specific node. These distances are based on the values from the trained map. This is how we estimate $||\boldsymbol{h}||$. For the i^{th} node, we have $\hat{f} = \frac{c_i}{||\boldsymbol{h}||}$ where c_i is the count of genes mapped to that node. For any observation $\boldsymbol{x}_g$, we estimate $f_x(\boldsymbol{x_g})$ using the estimate $\hat{f}$ for its BMU.

2.3.2.6 Clustering

Since each observation is mapped (via BMU) to a single node, the simplest clustering from SOM can be considered at the node level. Without augmenting the map itself, changing these clusters can be accomplished by trying to merge clusters together. Once the map has been trained, we start joining together the nodes into clusters. Since each observation has a BMU node, once nodes are grouped into clusters, it is trivial to map the observations to clusters. What isn't trivial is how to join nodes together, and determining when the proper number of clusters has been achieved. We use a variant of a popular hierarchical clustering algorithm, the Ward clustering method, and use the structure of

our map to improve upon it. The Ward method is an agglomerative hierarchical algorithm. Each observation is considered its own cluster. Then the two cluster which are most similar are joined together. Similarity is based on a distance metric, which is recorded. This is continued until there is a single cluster. The optimal number of clusters is determined by looking at the distances of joined clusters at each step and determining the optimal number of clusters. All pairwise comparisons can be computationally intensive, and minimizing a distance metric alone isn't always going to arrive at good pairings. Since this algorithm is hierarchical, any step in the process effects all further steps, meaning that initial joins can effect the final product.

2.3.2.6.1 SOM Ward

The first clustering approach we considered is refered to as SOM Ward clustering. It considers the same distance metric we used to train the map. The distances between all nodes is based on the distance between them in the data space, but also the map space. Nodes can only be joined together if they are adjacent in the map. Should the map be useful, this ensures that we only join together nodes which are similar. At each step, we track the distance between the joined nodes, and use this to find an optimal number of clusters. This can be used as a step zero for our EM, where it finds the number of clusters, and each cluster can be used to assign starting values to our EM. We use $\bar{Z}_i = \sum_{g \in Node_i} \boldsymbol{x_g}/n_i$ where n_i is the number of observations mapped to node i. For two nodes, j and i, the distance is calculated as $||\bar{Z}_i - \bar{Z}_j|| * \frac{n_i n_j}{n_i + n_j}$ so long as they are adjacent. If they are not, the distance takes value of infinity. Also, if $n_i = n_j$, we set the distance to 0. When nodes i and j are merged, the updated count becomes $n_i + n_j$, the updated

Figure 5: Pseudo-code: SOMWard

```
1: procedure SOMWard
2:     Find Z̄_i, the sample avereage for all data mapped to the i^th node.
3:     Find n_i, the count of data mapped to the i^th node.
4:     i ← 1
5: top:
6:     Find S_i = {s_{i_1}, ..., s_{i_{n_i}}}, the set of adjacent nodes to node i.
7:     Find distance between node i and all adjacencies, track minimum distance and
   pairing.
8: Loop
9:     if i < n_map then
10:        i ← i + 1
11:        goto top
12:    else if i == n_map then
13:        Merge node and adjacency with minimum distance
14:        if n_map > 3 then
15:            n_map ← n_map − 1
16:            i ← 1
17:            goto top
18:        else END
```

mean value becomes $\frac{\bar{Z}_i * n_i + \bar{Z}_j * n_j}{n_i + n_j}$, and $n_{map} = n_{map} - 1$. The pseudo-code for this can be found in figure 5.

2.3.2.6.2 Clustering Around the Null

We propose an alternative method for creating clusters in our SOM setting, Clustering Around the Null (CAN), by focusing on the null cluster. The null cluster in our setting is centered at 0. So, we first find the node to which our zero vector is mapped (BMU). Instead of checking all adjacent nodes in order to find the most natural joins as in the case of SOMWard, we focus only on joining nodes to the current zero cluster. Instead of using Euclidean distance, we focus on finding the node which maximizes an estimate of $P(Y_g = 1|\boldsymbol{X}_g)$. Here, we technically are considering joining nodes together, with expression $\bar{Z}_i$, so we rewrite $P(Y_i|\bar{Z}_i)$. We are abusing notation slightly, but the idea is to try to assign the entire node i to the null cluster, so we use $\bar{Z}_i$. This requires some parametric

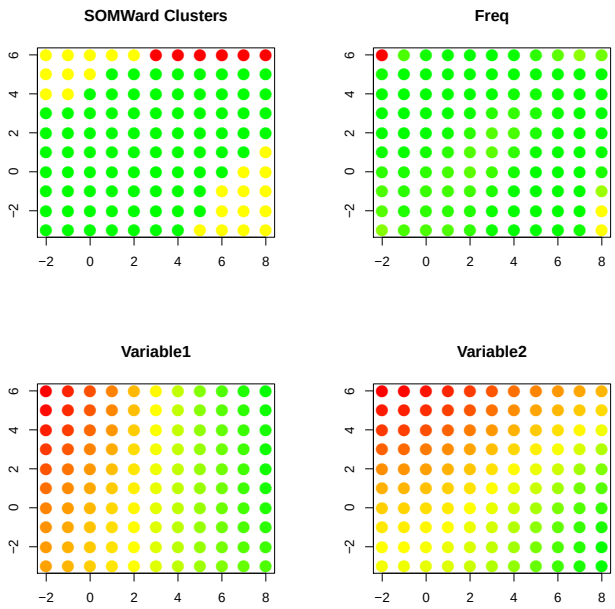


Figure 6: SOM output:Full output including clustering and frequency. SOMWard Clusters shows cluster by color, frequency shows frequency, and variable 1 and 2 are the variables from the data.

assumptions, so we consider the case where our null cluster (mixand) is centered at the origin, with a spherical shape. The easiest way to implement this, is to allow the null component to have a normal distribution. In this case, our criteria becomes equivalent to maximizing

$$P(Y_i = 1|\bar{\boldsymbol{Z}}_i) = \frac{p_1 f_1(\bar{\boldsymbol{Z}}_i)}{f(\bar{\boldsymbol{z}}_i)} \tag{6}$$

Here, the denominator $f(\boldsymbol{x})$ is typically estimated using $f(\boldsymbol{x}) = \sum_{y=1}^{k} p_y f_y(\boldsymbol{x})$. This would require we know the form of the DE clusters, and the parameters governing them. We can avoid needing to burden ourselves with such strict assumptions by estimating $f(\boldsymbol{x})$ in a different manner. We realize that SOM delivers a set of bins for our data in the multidimensional space, and from that arrive at estimates for $f(\boldsymbol{x}_g)$ by using the pdf estimate assigned to the BMU for the g^{th} gene. This means that we only require our null

```
1: procedure CAN
2:     Find the zero node, call zero cluster
3:     Set the number of clusters, n_Clust = n_map
4: top:
5:     Find the set of nodes adjacent to the zero cluster, S
6:     For each s ∈ S, calculate the distances between s and the zero cluster, track
       minimum distance and pairing
7: Loop
8:     if n_Clust > 2 then
9:         Merge node with minimum distance to zero cluster
10:        n_Clust ⇐ n_Clust − 1
11:        goto top
12:    else
13:        END
```

Figure 7: Pseudo-code: CAN

mixture to have a parametric form. We need not know the true number of clusters, the parametric form of the non-DE clusters, or the parameters governing those clusters.

2.3.2.7 Optimal Cluster Identification

In order to estimate our criteria, we still need to have estimates for p_1, $\boldsymbol{\mu}_1$, and $\boldsymbol{\Sigma}_1$. We estimate these from the current EE cluster. We opt to use $\boldsymbol{\mu}_1$ as the zero vector. If the analyst believes that the true mean should be close to the origin, but not necessarily exactly the origin, they can instead use a sample average here. We also notice that the estimates for p_1 should change quite a bit as we join more clusters together. In terms of maximizing our criteria, all of our terms will share p_1, so the fact that this value changes over time is of no concern here. The pseudo-code for this clustering can be found in figure 7:

In order to find an optimal number of clusters, the SOMWard method looks at the distance of the merged cluster at each step. Specifically, we define c as the number of clusters after the merge (i.e. we go from 50 to 49 clusters for $c = 49$). We then have $d(c)$

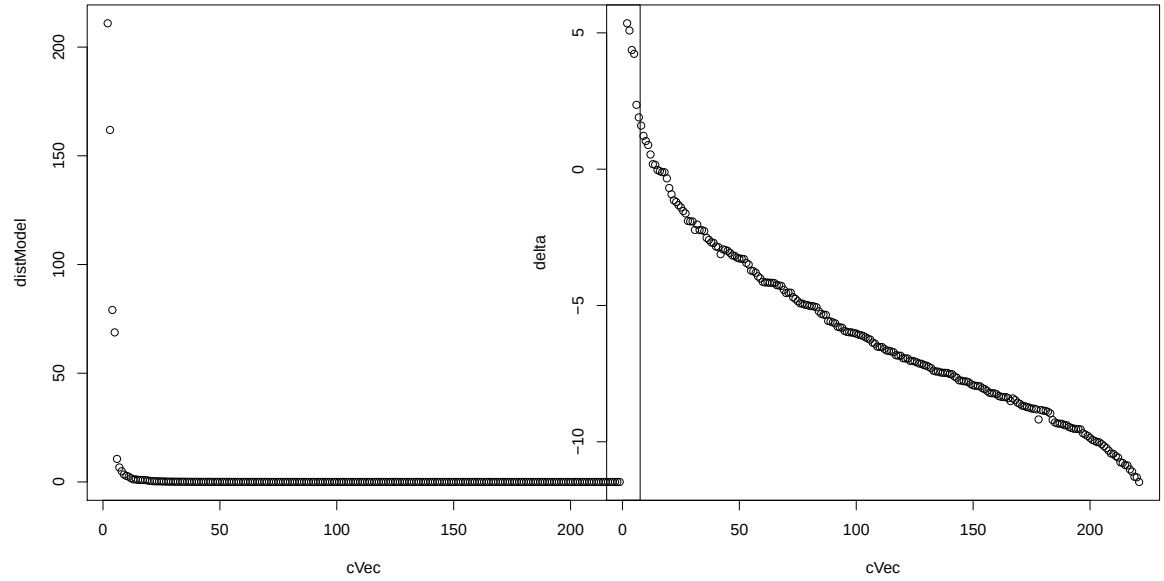


Figure 8: Distances (log distances) for the joined clusters versus the number of clusters

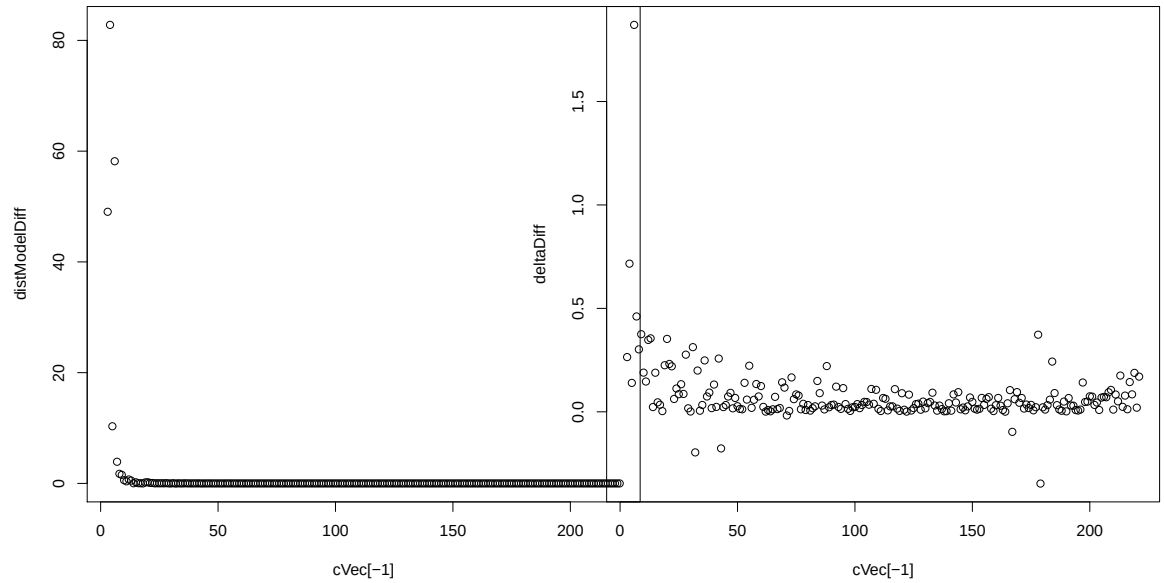


Figure 9: Differences of the distances (log distances) for the joined clusters versus the number of clusters

as the distance of the merged clusters at point c, and wish to use these values to identify when we go from the correct number of clusters to too few clusters. Figure 8 shows the distances and logged distances plotted vs. c. We notice that each step tends to yield a larger distance, although unlike the Wald algorithm, these values are not monotonic. We wish to identify the value of c for which we have a large jump in these distances, or log distances, and after which the jumps seem to be smaller. Sometimes this is referred to as a knee in the graph. As this should be an unsupervised method, we can't expect the user to know how to do this, and use the simple rule of identifying the c for which we have the largest difference in the distances and logged distances.

Regardless of clustering method, we have a number of clusterings for a number of clusters, c, and the distance of the nodes joined, $d(c)$. In order to identify the true number of clusters based on these two values, we noticed that there was consistently a large jump into the correct number of clusters in the $\log(d(c))$ values, followed by relatively small jumps. We iterated through a number of different mixtures, with number of mixands ranging from 2 to 10, in which we found the true number of clusters with a very high chance. Using a top number of clusters for both logged distance and distance returned the correct number of clusters within the top five nearly everytime. We did notice that increasing the variance in the mixands, while leaving the mean parameters fixed, lead to difficulty identifying the true number of clusters. These mixands will have the same location, but with variance decreasing in sample size. That being said, for any experiment satisfying the mixture of normal assumption, as well as others, the SOM method for identifying the true number of clusters should work as we increase our number of replicates.

For our alternative to the SOMWard method, we are not trying to find the correct number of clusters. We wish to join nodes to our null cluster until we start to add nodes

which seem different from our null group. To measure this, we actually consider our distance metric $\hat{P}(Y_i = 1|\bar{\boldsymbol{Z}}_i)^{(-1)}$. Similar to above, we wish to find the case where this distance goes from a set of small values to large ones. To this end, we look at the jumps from c to $c-1$ as $d(c) - d(c-1)$. We find the standard deviation for these distances across c, and then try to identify the value c for which $d(c) - d(c-1)$ is large with respect to that standard deviation. This is the point at which we feel our joins change from joining null nodes to DE nodes. Our goal is to use this final clustering number to estimate $P(Y_i = 1|\bar{\boldsymbol{Z}}_i)$ for each of our genes in hopes of classifying them as DE and EE. If we join too many nodes to our null cluster, we will overestimate p_1. This will result in a small group of genes being called DE. If we join too few nodes, we will underestimate p_1. This will result in a large group of genes being called DE. So we can't simply be conservative here, and must try to find the correct place to break. We search for the first jump which exceeds the standard deviation, scaled by a number larger than 1. It is important to notice that while the SOMWard method arrives at starting points for our EM, estimates the true number of clusters, and can be used to find DE genes, this alternative focuses on the machinery which is used to call genes DE, and cannot accomplish the other two tasks.

2.3.2.8 SOM: Non-Parametric DE Classification

We can extend this to actually call genes DE based on which cluster they are assigned to. The first step is to identify which cluster is our EE cluster. For our specific setup, this can be done by mapping a zero vector to its BMU, and whichever cluster that node belongs to we call our EE cluster. In general, if such a relationship cannot be assumed, we can still use the fact that most genes are considered to be EE. We can choose our

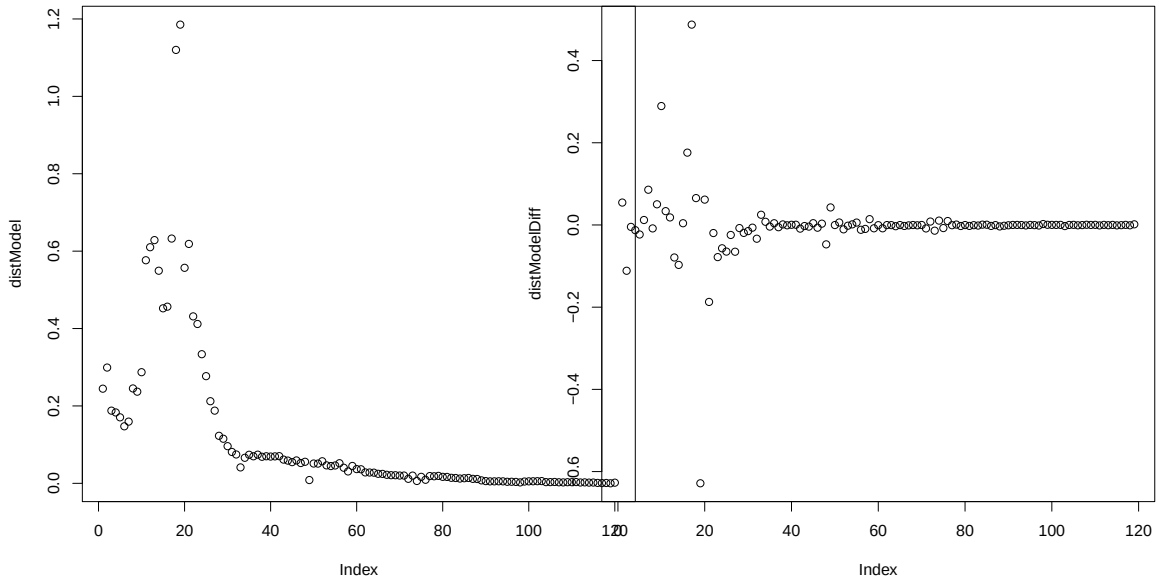


Figure 10: Distances and Distance Jumps for Ward Alternative

largest cluster (by frequency) to be our EE cluster and move forward that way. Once our EE cluster has been identified, we can say all genes mapped to it are not called DE, and all genes mapped to different clusters are considered DE. Since we use no distributional assumptions here, we can arrive at our clustering in a purely non-parametric manner.

2.3.2.9 SOM: Semi-Parametric DE Classification

We can extend this to a semi-parametric approach depending on the distributional assumptions we make. Since we are using a mixture of normals assumption, we can consider $P(Y_g = 1|\boldsymbol{X_g})$, where $Y_g = 1$ is equivalent to the gene being EE. In this setting, $P(Y_g = 1|\boldsymbol{x_g}) = \frac{f_1(\boldsymbol{x_g})*p_1}{f_x(\boldsymbol{x_g})}$. Technically, we can think of each node in our map as a bin, and the relative frequency as a non-parametric estimate of the multivariate density. So, we can use this to estimate $f_x(\boldsymbol{x}_g)$, we can use the relative frequency of genes mapped to

the null cluster to estimate p_1, and we can use either the zero vector OR the sample mean of all vectors mapped to the null cluster to find the mean of that cluster, $\boldsymbol{\mu}_1$. We choose to use the sample mean of the genes mapped to the null cluster, this is nice because it allows the true mean to differ slightly from zero, but by construction this should be very close to zero. Then, we can use the data to estimate the standard deviation, $\boldsymbol{\Sigma}_1$. For simplicity, we limit ourselves to the case where the covariance matrix for our observations is diagonal. Based on these values, for any observation, we can estimate $f_1(\boldsymbol{x}_g)$ with $\frac{1}{\sqrt{2\pi|\boldsymbol{\Sigma}_1|}}e^{-1/2(\boldsymbol{x}_g-\boldsymbol{\mu}_1)'\boldsymbol{\Sigma}_1^{-1}(\boldsymbol{x}_g-\boldsymbol{\mu}_1)}$. Instead of just calling each gene DE based on the cluster its BMU is mapped to, we can order the genes by our estimates for $\tau_{g1} = P(Y_g = 1|\boldsymbol{x_g})$. In fact, we will later introduce the rejection rule for our EM algorithm, and this very rejection rule can be applied using these values as well. Notice this allows us to control for an estimated FDR, but this estimate isn't as accurate as the EM when the assumptions are met. The relatively loose assumption of a normal null cluster gives us the ability to estimate FDR, which we cannot do in the non-parametric case.

2.3.3 Parametric

A common set of assumptions when finding DE genes is normality of the data. Should we choose to cluster after finding DE genes, we typically use some mixture distribution assumption. So, when we analyze our data and find clusters, depending on our methodology, we can be assuming a mixture of normals. Further, this mixture of normals is a very common assumption for the clustering algorithms themselves. For this reason, we focus on a mixture of normals for our parametric assumption. This falls into the framework of the Unified Method. We will both state how to use our SOM methodology to simplify

the computations needed for this method, and use this Unified Method to interpret the results of our simulation study.

2.3.3.1 Initializing EM With SOM

The SOM can be used as described above. However, we can also use it as a first step, in conjunction with our EM algorithm. We can use SOM to first identify how many clusters there are within the data. Then, we can use it to estimate the locations, standard deviations, and proportions of the mixands. If accomplishable, this can be invaluable to anyone using an EM. Typically, the EM must be run on a set of feasible cluster numbers. After this is finished, some measure such as BIC is compared across the number of clusters in order to find an optimal one. Historically, this has met criticism both over the possible arbitrary nature of the value being optimized (many times two such values do not agree on the optimal number of clusters). Also, these values rely on convergence of the EM. However, when the incorrect number of clusters is specified, the EM need not necessarily converge. It can be difficult to set an upper bound on the number of iterations to run the EM. Also, the convergence rate is greatly affected by the starting points of the EM. While there are tricks to try and best set up the EM without any analysis of the data, having some initial step can greatly reduce the number of iterations needed for convergence.

In order to use our SOM as a step 0 to our process, we run the grow step, the smoothing step via batch algorithm, and run SOMWard clustering. Once accomplished, we choose the number of clusters based on the SOMWard output. This gives us the number of clusters. We can use the proportion of genes mapped to each cluster to identify its mixand percentage, and the sample means and standard deviations will serve as the initial mean and standard deviation estimates for the EM algorithm. At this point, we switch to the

EM algorithm, and need to only run the one iteration. We expect this to run much quicker than had we not estimated the starting values with some method. Should the data analyst be worried about the effectiveness of using our SOM method for identifying the number of clusters, it would still be advantageous to look at the SOMWard clusterings ranked by our cluster identification criteria. Considering the top 10 (or whatever top number the analyst desires) clusterings, using the SOMWard output to estimate the initial points of the EM for those cluster numbers would allow the analyst to prioritize likely numbers of clusters, as well as getting more quickly converging results due to the data driven initial points.

2.3.3.2 Expectation Maximization

We will focus on a mixture of normals for the remainder of this paper, making it simpler to summarize the parameters of interest, but the technique can be generalized for other distributions. We will need to have already identified estimates of the mixture proportions, $\boldsymbol{p} = (p_1, ..., p_K)$, the means $(\boldsymbol{\mu_1}, ...\boldsymbol{\mu_K})$, and covariance structures $(\boldsymbol{\Sigma}_1, ..., \boldsymbol{\Sigma}_K)$. These estimates are useful, but mainly in application. In practice, we are most concerned with estimating $P(Y_i = k|\boldsymbol{X}_{i,1} = \boldsymbol{x}_{i,1}, ..., \boldsymbol{X}_{i,n_0} = \boldsymbol{x}_{i,n_0})$. We assume that there is a parametric form to this, and simply use our estimates for the above listed parameters to find

$$\widehat{f}_{y|\boldsymbol{x}}(k|\boldsymbol{x}) = \frac{f(\boldsymbol{x}|\widehat{\boldsymbol{\mu}}_k, \widehat{\boldsymbol{\Sigma}}_k)\widehat{p_k}}{\sum_{j=1}^{K} \widehat{p_j} f(\boldsymbol{x})|\widehat{\boldsymbol{\mu}}_j, \widehat{\boldsymbol{\Sigma}}_j)\widehat{p_j}}$$

As mentioned earlier, some EM algorithm is typically employed for the estimates of the parameters. Once we have them, we have this estimate of the probability that any gene belongs to cluster k given the expression data.

2.3.3.3 Rejection Rule

We have currently suggested that K true groupings of the data exist. We will fix the first group as being EE, in most cases this means that $\boldsymbol{\mu}_1 = \mathbf{0}$. In order to perform any classification, we will be considering the following estimate

$$\hat{\tau_k} = \widehat{f}_{y|\boldsymbol{x}}(k|\boldsymbol{x}) = \frac{\widehat{p_k} f(\boldsymbol{x}|\widehat{\boldsymbol{\mu}}_k, \widehat{\boldsymbol{\Sigma}}_k)}{\sum_{j=0}^{K-1} \widehat{p_j} f(\boldsymbol{x})|\widehat{\boldsymbol{\mu}}_j, \widehat{\boldsymbol{\Sigma}}_j)} \tag{7}$$

In practice, we use $\hat{\tau}_{ik} = \widehat{P}(Y_i = k|\boldsymbol{X})$. Since we wish to find genes likely not from the null group, we consider $\hat{\tau}_{i0}$ as our test statistic in a sense. We can first consider some value we wish to control our FDR for, $q \in (0, 1)$. First, we order our $\hat{\tau}_{i1}$ from smallest to largest, denoted by $\hat{\tau}_{(i)1}$. The next step is to find some value, $s^{(*)}$ satisfying $\sum_{i=1}^{s^{(*)}} \hat{\tau}_{(i)1}/s^{(*)} \leq q$ and $\sum_{i=1}^{s^{(*)}+1} \hat{\tau}_{(i)1}/(s^{(*)} + 1) > q$. We call all genes DE so long as $\hat{\tau}_{i1} < s^{(*)}$. This same ranking and Call DE rule can be applied to the semi-parametric approach we developed within the SOM setting.

2.4 Interpretation

This paper wishes to explore the performance of all of the methods in a series of different settings. Since many of these methods do not accomplish identifying the underlying latent structure of the genes, we will focus on characteristics such as FDR. This will be accomplished via a simulation study. We will also explore different rejection rules where appropriate, trying to identify the one yielding the best results. Finally, we will continue by exploring the characteristics of the method suggested by Yuan and Kendziorksi, with appropriate extensions, with respect to identifying that underlying latent structure. This

extension will be anchored in confidence intervals which will allow the biologists to better assess which comparisons genes tend to be expressed in per cluster, allowing them to better understand the biological reasoning for them to be clustered together.

We notice that for the j^{th} dimension of the k^{th} location parameter, μ_{kj}, our estimate is $\sum_{i=1}^{G} \hat{\tau_{ik}} X_{ij}/(\hat{p}_k G) \to \sum_{i=1}^{G} \tau_{ik} X_{ij}/(p_k G) \to \mu_{kj}$. Also, $Var(\tau_{ik} X_{ij}) \leq Var(X_{ij}) < \infty$. So, let us call $\omega_{ikj}^2 = Var(\tau_{ik} X_{ij}/p_k)$. and $A_{ikj} = \hat{\tau}_{ik} X_{ij}/\hat{p}_k - \hat{\mu_{kj}} \to \tau_{ik} X_{ij}/p_k - \mu_{kj}$, so for a set k, $E[A_{ikj}] = 0$ and $Var(A_{ikj}) \to \omega_{ik}^2$. So, $E[A_{ikj}^2] - \omega_{ikj}^2 = 0$, and $E[(\hat{\tau}_{ik} X_{ij})^4] \leq E[X_{ij}^4] < \infty$, so $E[A_{ikj}^4] < \infty$, meaning that $Var(A_{ikj}^2) < \infty$. So, by the law of large numbers, $\frac{\sum_{i=1}^{G} A_{ikj}^2 - \sum_{i=1}^{G} \omega_{ikj}^2}{G} \to 0$. This is equivalent to $\frac{\sum_{i=1}^{G} A_{ikj}^2}{G} \to \frac{\sum_{i=1^G} \omega_{ikj}^2}{G}$. Also, as $Var(\hat{\mu}_{kj}) \to Var(\sum_{i=1}^{G} \tau_{ik} X_{ij}/(p_k G)) = \frac{\sum_{i=1^G} \omega_{ikj}^2}{G}$. So, we say $\widehat{Var}(\hat{\mu}_{ikj}) = \frac{\sum_{i=1}^{G} A_{ikj}^2}{G}$, and create our typical CI since for sufficiently large G, using $\hat{\mu}_{ikj} \overset{Appox}{\sim} N(\mu_{ikj}, \sigma^2(\hat{\mu}_{kj}))$

2.4.1 Results

In order to compare these methods, we have run a simulation study. Here, we assume that the genes truly follow some mixture normal distribution. We stick to time series experiments of 3 time points, giving us a total of 2 contrasts, in order to make computation times more reasonable. First, we compare the ability of these different methods to identify differentially expressed genes. Then, we continue by looking at the ability of the confidence interval approach to identify which of the contrasts are themselves differentially expressed. The following table summarizes the three scenarios we have considered in our study.

2.4.2 Mixture of Normals Distribution

For each of these scenarios, we simulate 3 replicates of 50,000 observations from the appropriate mixture distribution. We run a simulation study over 1000 iterations for all

scenarios considered, and report values of interest which we will look over and summarize. The true data we simulate is summarized in 1. For each of the three scenarios, we

Table 1: Sampling Distributions For Mixture of Normal Distriubtions

Scenario	Distribution
1	$.8N\left(\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}2 & 0\\0 & 2\end{pmatrix}\right) + .2N\left(\begin{pmatrix}1\\2\end{pmatrix}, \begin{pmatrix}1 & 0\\0 & 5/6\end{pmatrix}\right)$
2	$.6N\left(\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}2 & 0\\0 & 2\end{pmatrix}\right) + .1N\left(\begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}1 & 0\\0 & 2\end{pmatrix}\right) + .1N\left(\begin{pmatrix}-1\\2\end{pmatrix}, \begin{pmatrix}1 & 0\\0 & 2\end{pmatrix}\right) + .2N\left(\begin{pmatrix}0\\-2\end{pmatrix}, \begin{pmatrix}1 & 0\\0 & 5/6\end{pmatrix}\right)$
3	$.6N\left(\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}2 & 0\\0 & 2\end{pmatrix}\right) + .2N\left(\begin{pmatrix}1\\3\end{pmatrix}, \begin{pmatrix}2 & 0\\0 & 2\end{pmatrix}\right) + .2N\left(\begin{pmatrix}-1\\2\end{pmatrix}, \begin{pmatrix}2 & 0\\0 & 2\end{pmatrix}\right)$

first look at a table summarizing the FDR, FNDR, Type 1, and Type 2 errors observed over the 1000 iterations. The results are summarized in Table 2. Since we are dealing with values necessitating rejection rules, we try to control the FDR for each method at the level of .05. Focusing on the existing methods, Limma, SAM, and Unified do a good job of controlling for FDR. Limma and SAM seem to perform slightly better than Unified in the simplest, first scenario with just 1 DE cluster, while they are outperformed in nearly all aspects by the Unified in the other two, more complicated scenarios with multiple DE clusters. All the established methods outperform the non-paramteric and semi-parametric methods we propose in these three scenarios. We focus on the Unified method. We see that the non-parametric methods cannot control FDR nearly as well as they need to, and that in general our proposed CAN clustering outperforms the SOMWard results in classification as DE. Further, we see that the semi-parametric approach applied to the CAN clustering results yields the best results. The FDR isn't controlled at the .05 level as desired, but only in one of the three cases is the FDR much larger than the desired value of .05. Further exploration into the approach could find a better manner for estimating the FDR.

Table 2: Error rate comparisons among seven methods from simulation study 1

Scenario	Method	FDR	FNDR	Type 1	Type 2
1	WardSP	0.1307	0.1174	0.0385	0.5294
	WardNP	0.3622	0.0186	0.2113	0.0568
	CANSP	0.0726	0.1845	0.0014	0.9057
	CANNP	0.2109	0.1065	0.0468	0.4678
	Unified	0.0499	0.0594	0.0098	0.2503
	Sam	.0369	.0323	.0008	.1327
	Limma	.0211	.0000	.0005	.0001
2	WardSP	0.2556	0.312	0.0871	0.6456
	WardNP	0.5025	0.216	0.5369	0.2068
	CANSP	0.1121	0.380	0.0080	0.9203
	CANNP	0.2635	0.260	0.1407	0.4892
	Unified	0.0500	0.267	0.0161	0.5399
	Sam	0.1312	0.3366	0.0260	0.7412
	Limma	0.0101	0.3935	0.0001	0.9731
3	WardSP	0.03601	0.21108	0.019098	0.40404
	WardNP	0.18584	0.05549	0.164796	0.07768
	CANSP	0.00636	0.36303	0.000826	0.86033
	CANNP	0.03221	0.24867	0.015627	0.51379
	Unified	0.04999	0.08071	0.030610	0.12766
	Sam	0.0133	0.2664	0.0041	0.5423
	Limma	0.0057	0.2486	0.0019	0.4952

2.4.3 Non-Traditional Mixtures

We now explore two examples favorable to our nonparametric approach. The results can be seen in Table 3. The first example is motivated from the common assumption of a null group, and up regulated group, and a down regulated group. This is often applied to univariate data, but is not trivial to extend to the multivariate case. Refer to figure 12 to see a scatter plot of the two dimensional data we generate. The null cluster is centered at the origin with a normal distribution, and there is a ring of data around the null cluster, clearly defined from the null group. This is a departure from the mixture of normals setting. In fact, when given the number of 2 clusters, the EM should never converge, but instead just keep moving in the ring around the null group. We see that the SOM based methods can analyze the data, and again the SOMWard Alternative semi-parametric approach arrives at the best results, which controls for an FDR well below the desired .05 while having a FNDR of nearly zero.

When applying the EM based mixture of normals approach, we use a lower bound of 2 and an upper bound of 20 clusters. We use BIC as the criterion for choosing which is the correct number of clusters. We then use the rejection rule outlined earlier in the paper. That rejection rule aims to control the expected FDR by cutting based on the average of the selected genes estimated chance of being EE, $\hat{\tau}_g$. This rule tries to grab the largest group of genes called DE while controling that expected FDR. A more conservative alternative would be to select all genes with $\hat{\tau}_g$. We show the performance for both rejection rules applied in our simulation study to give the best representation of the EM based approach's performance. It wasn't obvious to us how the EM would try to fit to the data, which number of clusters would be selected, or even how the algorithm would

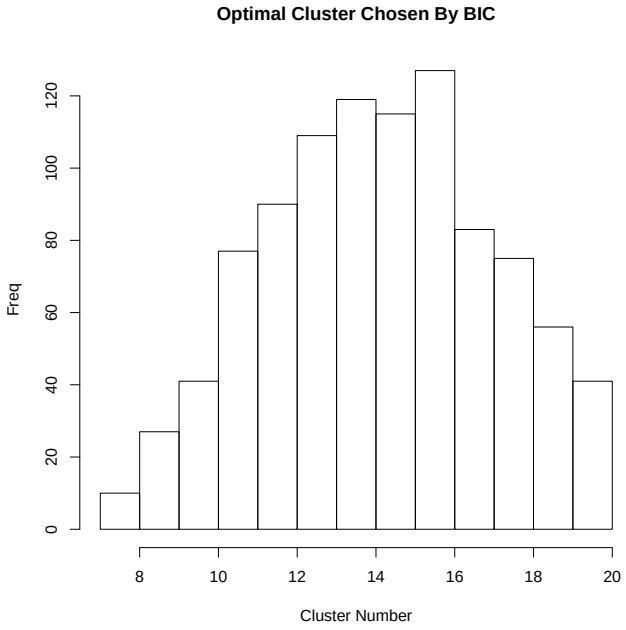


Figure 11: Histogram of Optimal Cluster Numbers Chosen By BIC For Example 1

perform for each cluster chosen. We walk the reader through the EM applied to one such simulated dataset, and then explain how the method did in the simulation study.

It turns out that for $k = 2$ clusters, the mixture of normals assigned a location very close to the origin for both clusters. The DE cluster was further from the origin, and had a much larger variance estimate. These heavy tails actually split the data fairly well into a null and DE grouping. We show the results for the conservative rejection rule, which performed best for this example. The algorithm correctly identifies all DE genes, and in doing so has a fairly low FDR. This has the worst BIC, however. The best BIC is for 12 clusters for this example, but we can see from 11 that in general clusters from 8 to 20, mainly close to 14 were found as the optimal cluster by BIC. As more and more mixands are added, the fdr gets worse and worse. It seems like each time a new DE cluster is added to the EM, it splits the difference from the DE ring and the EE center. So every

time all DE genes are identified as DE, but more and more genes called DE are actually EE, illustrated by the raising fdr. It was unexpected that for low cluster numbers the fdr seems pretty good, but it was absolutely expected that BIC would select a number of clusters which would have poor results. The full results show that for the conservative

Our second example is to have the null mixture again centered at the origin with a standard deviation, and then to have the DE genes come from a normal centered at the origin with a larger standard deviation. We chose to give each DE gene its own standard deviation, where $\sigma_g = \sigma_0$ when gene g is null, and $\sigma_g = \sigma_0 + \gamma_g$ where $\gamma_g \sim \Gamma(\alpha, \beta)$. Also, here we let $\Sigma_g = \mathrm{I}\sigma_g$, and for our simulation $\sigma_0 = 1$ and $\alpha = \beta = 1$. So the expected value of the standard deviation of DE genes is twice that of the EE genes. We notice that this is not something which the EM can directly estimate, as it requires the standard deviation to be the same. Also, the EM breaks down on situations where the location paramters can be the same for multiple clusters. This could be easily extended to giving each DE gene it's own location parameter as well. As we see from the results, the SOMWard Alternative with the semi-parametric classification not only does the best, but manages to control for a very small FDR. If we look at the method with the best FNDR, using the Ward method with a semi-parametric rejection rule, we see a value of .152. The method controlling best for FDR, the alternative to the Ward clustering with a semi-parametric rejection rule, has a FNDR of .172. The tradeoff in FDR associated with the increase of .02 in FNDR is more than .2. In this example, many of the genes which are DE will be close to the origin, making it impossible to identify which ones are actually DE. Our best method controls for FDR very close to the desired level of .05, and identifies nearly all of the DE genes possible, without making the FDR explode.

Table 3: Error rate comparisons among seven methods from simulation study 2

Scenario	Method	FDR	FNDR	Type 1	Type 2
Example 1	WardSP	0.3089	0.0103	0.1514	0.0952
	WardNP	0.8777	0.0547	0.6137	0.2431
	CANSP	0.0276	0.0194	0.0032	0.1865
	CANNP	0.2270	0.0330	0.0253	0.3070
	EM1	0.272	0.000	0.048	0.000
	EM2	0.674	0.000	0.250	0.000
Example 2	WardSP	0.2722	0.1529	0.1400	0.6613
	WardNP	0.7801	0.1619	0.5728	0.3620
	CANSP	0.0400	0.1726	0.0038	0.8326
	CANNP	0.4403	0.1605	0.08273	0.7126
	EM1	0.569	0.208	0.241	0.486
	EM2	0.704	0.218	0.467	0.283

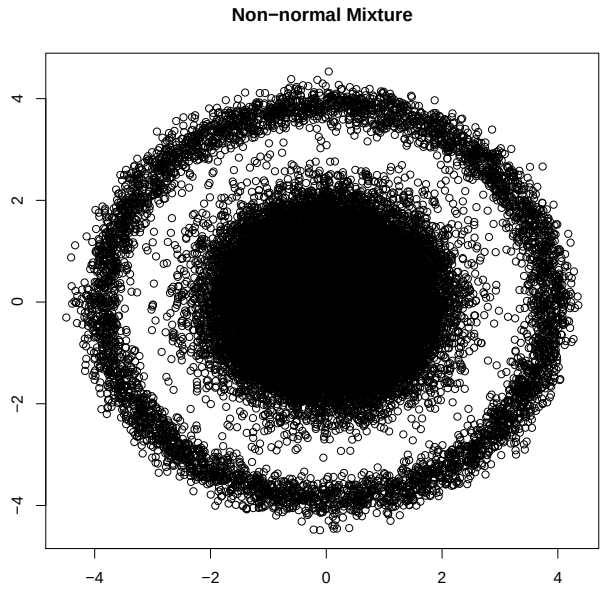


Figure 12: Plot of data from example 1

2.5 Conclusions

We see that for the three scenarios we consider the unified methodology either performs as well as or better than the others. It is due to our rejection rule that we can still control FDR in a sense, in all cases having less bias, while giving ourselves more power. Further, we notice that in terms of ranking genes, the unified method again performs as well and better than the other methods. It must be stated that one reason why our method will outperform the others is due to the more restrictive assumptions, namely the necessity of a mixture distribution.

We aid in the run time and true cluster number assignment by using SOM as a step zero for our EM algorithm. If we wish to be more careful than simply using the clustering returned by optimizing our SOMWard criteria, we can consider a top number of these clusterings. This allows us to compare the BIC (or the criteria of the users choice) across these numbers of clusters to see if our EM results agree with our SOM results. This allows for faster and more reliable results from each run of our EM algorithm, as well as allowing for less runs of the EM to be required. Not only does this save time, but limits the chance to have an improper result from the EM lead to a mistake in the analysis.

Finally, we explore the usefulness of applying the optimal clustering returned by SOM to classify the genes are either EE or DE. We can see that using the alternative clustering proposed in this paper over a more traditional SOMWard algorithm isn't able to identify the true cluster number, and isn't useful in finding final groupings, but performs superiorly to the SOMWard algorithm in our classification scheme. SOMWard can then be used to explain the groupings the DE genes fall into. This accomplishes the unification of classification and calling genes DE simultaneously, using the same MAP for both steps.

The non-parametric DE classification performs worse than the semi-parametric across the board, and the semi-parametric method allows for estimation of and control of FDR. While it didn't perform as well as the EM counterpart, it uses less parametric assumptions and can be used to find likely cluster numbers and initial points for much quicker and more accurate results from an EM algorithm. We show examples showing two intuitive mixtures, the data for which the EM cannot analyze since they depart from the parametric assumptions. In one case the EM algorithm need not even converge. For both examples, we applied the EM algorithm using a mixture of normals assumption where BIC was used to determine the true number of clusters. The EM algorithm had very poor performance in these instances. Our algorithm is capable of classifying the genes as EE and DE with great success since they fit the assumption of having a normal null mixand centered at the origin. It is noteworthy that using the SOM to estimate the density $f_x(\boldsymbol{x})$ allows us to relax our assumption about the non-null mixands in our mixture, and is vital to estimating the FDR rate. This opens up our semi-parametric approach to a number of different mixture assumptions, and seems likely to fit microarray type data. Other gene study data, such as RNA-Seq data can have the form to fit this type of an experiment too. As the sets of genes can be quite large in this setting as well, this method seems a good fit.

Chapter 3

RNA-Seq

Understanding which strands, and their abundances, of RNA are present in differing conditions for an organism offers a great deal of insight as which genes are involved in processes. In order to get this understanding, tissue is treated in these differing conditions. Cells from this tissue then have their RNA isolated, fragmented, copied into complimentary DNA (cDNA), and finally amplified using some polymerase chain reaction (PCR). This allows for the sequencing of RNA reads. These reads are put into a raw data format with a name, and in some cases a quality score. There are numerous technologies, along with specific methods for using the above steps to get the RNA into a readable form. These reads are RNA sequences, and typically these are gathered in large numbers. Computational methods are then applied to these reads. The first computational step is typically to allign these reads to a genome. The next step involves getting these alligned reads into transcripts, equivalent to getting the counts of reads mapped to specific areas of the genome, such as genes. Finally, this count data is analyzed via some statistical method.

We focus on a subset of RNA-Seq experiments, those belonging to the second generation of technology, refered to as next generation sequencing (NGS). Popular and widely used RNA-Seq platforms include Illumina, Solid, Sage, and Roche 454. For both Illumina and Solid platforms, a device called a flow cell is used to gather data. Each flow cell is made up of a number of lanes. These lanes allow for the use of replicates. It is the case that a single biological sample can be sequenced in more than one lane, exactly one lane, or even that a number of samples can be sequenced on a single lane. The last case requires that each biological sample is treated with an index or barcode. This is a sequence of RNA which either starts or ends the strands of RNA being read. One file of reads is returned per lane, and in cases where multiple samples are sequenced on each lane, the barcodes are used to seperate the reads for each sample. The Roche 454 platform is considered a third generation platform. The distinction here is that Roche 454 returns smaller overall libraries of much longer reads. For our paper, we deal with data gathered in the Illumina platform, and from here on use the word lane in this context. None of our methods currently explore data gathered using barcoding or indexing techniques, meaning that each lane represents a single replicate.

RNA-Seq consists of counting the number of tags which fall into certain regions of the genome. Depending on the type of data collected, these regions can describe various characteristics of a person's makeup, such as genes. The typical progression of parametric assumptions to describe data is to begin with a Poisson distribution, and should some overdispersion exist within the data, move to the more general negative binomial distribution. Also, in some cases either the raw data, or some transformation there of follows an approximate normal distribution. This paper focuses on the existing methodologies utilizing a negative binomial assumption. Such methods must account for differing number of

reads per library, called library size and typically denoted with m. These methods allow for estimating an overdispersion parameter. We summarize these methods, and extend the approaches to incorporating a gene length variable, and consider models for different types of differentiation (i.e. specific ways a true mean can differ from control to treatment). Popular existing methods under this setting include EdgeR (Smyth and Robinson, 2007a), DESeq (Anders and Huber, 2010), and baySeq (Hardcastle and Kelly, 2010).

In RNA-Seq data, length bias refers to the fact that methods tend to favor calling genes of a greater length DE. As will be covered in more detail later, when we look at gene length, we really mean the length of transcripts mapped to a gene. So it can be the case that a gene has a larger region on the genome, but is given a shorter value for the gene length. One explanation for length bias is that longer genes have a larger area for the reads to be mapped to. Genes which have larger areas tend to have either more transcripts mapped to them, or more reads mapped to larger transcripts. This makes these gene counts more reliable. Another explanation, which we have come accross as a result of the work on this paper, could be that we count a gene as being large because we are counting multiple regions of a gene, a region activated under the control, and a different region activated under a treatment condition. This will be clearer when we cover gene length in more detail, but we introduce a model which aims to identify a group of genes which have different regions activated under the control and the treatment. Another explaation we come upon through the work for this paper is that genes with shorter length will have more zero counts, an occurence which doesn't simply mean a count of zero was observed, but can also occur due to errors in gathering the raw reads and processing them. By treating these zeroes as zero counts against the sampling distribution directly,

variability is added into the counts for those genes which is not present in the longer lengthed counterparts.

Our work has led us to consider different ways in which genes can be DE. The most basic way to identify genes that are DE is to find genes for whom there is a difference in counts between a control and treatment condition. At it's simplest, this can be done by taking the total reads for control and treatment, breaking these reads into bins for each gene, and performing a Fisher Exact Test (Fisher, 1922). This approach requires minimal assumptions, and doesn't take into account gene length or library size. Gene length is typically calculated as length of the union of the transcripts assigned to a gene. Normally, we assign a single value to a gene, summarizing its length. A somewhat unstated assumption in this approach is that roughly the same region is being considered for all of the conditions. From our dataset, we noticed that there tended to be large differences in the length of the transcripts for the control and treatment, and that for certain genes the counts seemed to be proportional to the gene length within each condition. Along these lines, we wish to count gene length and model it slightly differently than what we have seen so far. We will measure each gene based on the reads in the control, and separately in the treatment condition. It is clear to see how this would extend to multiple groups. A difference in gene lengths between conditions would suggest that the regions activated under each condition differ. We notice that it is also possible for genes to have the same length in different conditions while activating different regions. Our model isn't equiped to identify these genes, although we hope to extend our methodology so it does.

We propose a model using the negative binomial distribution to model the data. We do so following the work of Smyth, who detailed the need for the negative binomial to account for the overdispersion within RNA-Seq type data. We also account for a library

size effect, and incorporate gene length as well. We use a hierarchical log linear model to estimate the means for the count data in each lane of the experiment. We incorporate an estimate for the probability of observing zero counts, as this involves a level of technical error due to the way the data is collected. Zero counts in an RNA-Seq experiment do not coincide with observing a count of zero directly, but have to do with the processing steps that take the raw read data into the form of count data. This zero inflation is estimated as a function of gene length. Treating the zero counts in this manner is very important in the analysis, as this will greatly effect the mean estimates. Our initial results lead to the consideration of different ways in which the genes can be DE. That is to say, biological explanations for why the gene counts for certain genes seem to vary from condition to condition.

This biological explanation is grounded in gene length. We calculate the gene trasncript length for each lane individually. We then get a overall summary length X_g by averaging the lengths for each lane, as well as condition-specific gene lengths X_{g0} and X_{g1} as the average of the control and treatment lanes respectively. This can be expanded to more than just a control and treatment set of conditions. This will allow us to identify genes for which using the condition-specific gene lengths eliminates a large amount of variance relative to using a single length value for both conditions. We also allow for an overall shift between the mean counts for the two conditions, as well as a shift in the effect per length between control and treatment, for either a single gene length or condition-specific gene length.

We use real data to illustrate the benefits of using a condition-specific gene length as well as using gene length to estimate the chance of zero counts. We use model assessment criteria and a plot illustrating the length bias within statistical methods to quantify the

benefits our method adds. Our results include a drastic improvement the fit of our model when our zero inflation technique is incorporated, as well as models including a condition-specific gene length ranking the highest by Deviance Information Criteria (DIC) and Logged Psuedo-Marginal Likelihood (LPML). We establish that our methodology ranks genes in a manner that doesn't seem to have any real bias for longer genes. In order to better understand how certain assumptions may effect our classification of DE versus EE genes, we consider a traditional receiver operating curve (ROC). We use a simulation study to assess the effects of changing the true distribution of the data and/or the modelling assumptions. We use the Area Under Curve (AUC) to measure these effects.

3.1 Data

We use the terminology of NGS devices such as the Illumina and Solid platform. That is, we consider RNA-Seq data which are collected from a device which gathers lanes of data. Should data be collected without the ability to spread sequenced data over a number of physical partitions, the same data can be arrived at after incorporating a barcoding or indexing technique. Each lane can be thought of as a replicate, the biological component of which determines if it is a technical or biological replicate (harvested from the same/different animals). Due to cost, some experiments use less replicates than would be prefered, making it difficult to fit complex models to the data. When cost becomes less of an issue, many replicates can be gathered, allowing for models with more parameters to be used.

We took the raw data provided by Gene Yeo, (Li et al., 2008). The experiment run was using data collected from Illumina's 1G genome analyzer. This data returns reads of length 35 bp. The typical experiment will return 8 lanes of data, however for our dataset,

one lane was bad and not read in. 4 of the lanes correspond to the control condition, and 3 of the lanes correspond to the treatment. Each of these reads was mapped to the human genome HG19 using tophat (Trapnell et al., 2009). Once we had the positions for our reads, we used cufflinks, (Trapnell et al., 2010), to find transcripts, and then finally to allign our transcripts to refGenes, a dataset made available by the UCSC genome browser, (Fujita et al., 2010), for HG19. The tools needed to accomplish all of these tasks are free for public use in galaxy, (Goecks et al., 2010), (Blankenberg et al., 2010), (Giardine et al., 2005). We call the total number of reads in any lane the library size, denoted by m_{ij}, that is the total number of reads mapped in a given lane for the j^{th} replicate of the i^{th} condition. Also, as we refer to gene length, it isn't the actual length of the region corresponding to a gene, but instead the length of the transcript(s) mapped into a single gene. Also, each replicate is a technical replicate. Figure (17a) illustrates these steps for a single lane, the count data for which is combined with those of the other lanes to get what we call the full data.

After getting count data for our experiment, we need to process the data by identifying genes which we don't have good data for. In this type of data, zero counts can represent an actual observation, or simply a case in which transcripts got incorrectly mapped. Along these lines, genes which have a large proportion of zeroes are considered more likely to not be involved in the study, but observed due to technical error. We use the conventional method of omitting all genes which are observed in less than half of the total lanes. We started with 6,467 genes, and had 3,693 genes pass our criterion.

3.2 Negative Binomial Model

There are two main parameterizations for a negative binomial random variable, Y

$$Y \sim NB(r,p) \;\; f(y|r,p) = \binom{y+r-1}{r} p^r (1-p)^y \tag{8}$$

$$Y \sim NB(\mu,\phi) \;,\; g(y|\mu,\phi) = \frac{\Gamma(y+\phi^{-1})}{\Gamma(\phi^{-1})\Gamma(y+1)} \left(\frac{\mu}{\phi^{-1}+\mu}\right)^y \left(\frac{1}{1+\mu\phi}\right)^{\phi^{-1}} \tag{9}$$

We notice that

$$\mu = \frac{1-p}{p} r \;,\; \phi = \frac{1}{r} \;,\; \sigma^2 = \mu + \mu^2\phi \tag{10}$$

Here, ϕ is reffered to as the overdispersion parameter, and μ and σ are reserved for the true mean and standard deviation, respectively. There are three very popular methods, Smyth and Robinson (2007a) suggested EdgeR, Anders and Huber (2010) introduced DESeq, and Hardcastle and Kelly (2010) introduced baySeq, for finding DE genes under the negative binomial setting. They start with the same approach. Let Y_{gij} denote the count of the g^{th} gene in the j^{th} replicate and the i^{th} condition. We assume $Y_{gij} \sim NB(\mu_{gij}, \phi_g)$ where $\mu_{gij} = \lambda_{gi} m_{ij}$. Basically, the form of normalization comes on allowing the expected count in replicate j of condition i to be proportional to the library size. The interpretation of λ_{gi} is the rate per library size for the g^{th} gene under the i^{th} condition. As such, the test for differentially expressed genes is described by:

$$H_{g0} : \lambda_{g0} = \lambda_{g_1} \; vs. \; H_{g1} : \lambda_{g0} \neq \lambda_{g_1} \tag{11}$$

The three methods differ in both the manner in which they estimate the gene condition specific rates, λ_{gi}, and the overdispersion parameters. Also, EdgeR and DESeq use p-values to correct for a false discovery rate (FDR) while baySeq uses posterior probability estimates directly. EdgeR is a method first introduced for SAGE type data. It uses a MLE approach for estimating the gene condition rates. Originally, EdgeR used a common

overdispersion parameter assumption. Since, they have introduced methods for estimating gene specific overdispersion parameters. They introduced an empirical Bayes (EB) method for borrowing information about the overdispersion parameters from genes. DESeq similarly allows for a gene specific overdispersion parameter. Specifically, it utilizes a relationship between the mean and the variance, which can be written as a function of the overdispersion parameter. P-values are returned, and a method for controlling for FDR can be implemented. BaySeq uses an EB approach to estimate the parameters and the posterior probability of a gene being DE. These methods are very popular and have been used to analyze many datasets. As our method for identifying DE genes differs mainly in the use of gene length, or more appropriately transcript length, we first outline the dataset considered, and the steps of converting it from raw to count data, and then introduce the transcript length to our methodology. In order to describe our new model, we need to discuss how gene length and zero counts are obtained in the context of RNA-Seq.

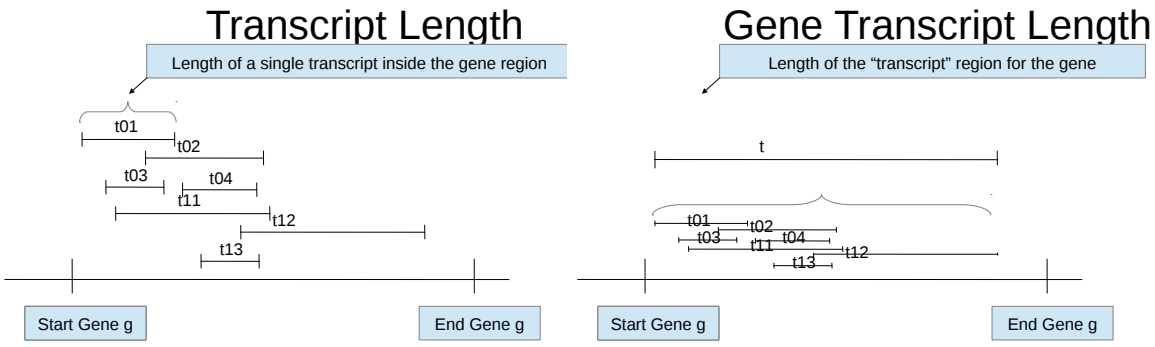
3.3 Transcript Length

The gene length is calculated for each lane. This is done by looking at the transcripts mapped to each gene within the lane. We call the entire portion of the genome covered by the transcripts mapped to a gene as that gene length for the lane. From here, transcript length is brought in as follows. For each count, Y_{gij}, we have a length associated with it, X_{gij}. As mentioned earlier, we will assign a single value X_g to each gene, where this is the average of the lengths accross all lanes (count). We call this value a summary gene length. We are introducing a new way to consider the gene length differently accross conditions. X_{g0} and X_{g1} denote the condition-specific gene length, which is the average of the gene lengths accross all lanes within each condition. In figures 13 and 14, t_{ij} refers

to the transcript from the i^{th} condition and the j^{th} replicate. There is exactly one lane corresponding to this i , j index. In general, the transcript can be considered a collection of smaller transcripts mapped to that gene within the same lane, and can have gaps. For simplicity, we have represented each transcript as a region with a single lower and upper point on the genome within the gene.

Refer to figure 14b to see how condition specific gene length is calculated. We perform the same steps as before, but we do so once for all the control transcripts, and then again for the treatment transcripts. This allows us to get a control gene length and a treatment gene length.

Figure 13: Transcript Lengths



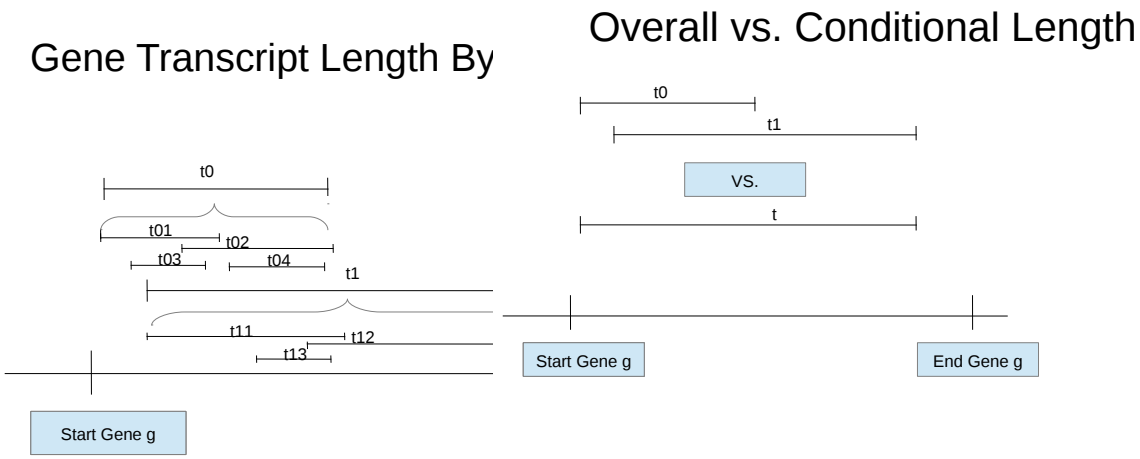
(a) Condition-Specific transcript length

(b) Gene Length

3.3.1 Zero Counts

Count data takes on only integer values. A constant consideration with it is how to deal with zero counts. Most distributions selected for count data allow for the value of zero

Figure 14: Transcript Lengths



(a) Condition Specific Gene Length

(b) One Length vs. Conition-specific Length

to be observed. In cases where there is an over or under abundance of zeroes in the data for a given distribution, either zero inflated or truncated distributions can be used to help the model fit the data. The way in which zero counts are observed lends itself to a zero inflated distribution. In fact, if we ignore the manner in which zero counts are 'observed' in this data, we will ultimately bias our estimates of the mean parameters, and result in a flawed analysis. In an effort to control for length bias, we introduce a zero inflation based on gene length. Finally, one suggestion for modeling DE genes in this setting will call for a condition-specific gene length to estimate the probability of observing a zero count within each condition for each gene.

For our data, we have zero counts only when a transcript gets mapped to a gene in at least one lane, and not in another. Getting mapped to a gene in any lane includes that gene in our study. Lanes where no transcripts are mapped to that gene lead to a zero count. This differs from our usual method of observing zero counts since the transcript assembly

and the mapping of transcripts to regions of the genome corresponding to a gene are done by computer science and statistical methods. The reason this is included in the gene length section is that there is no transcript length to calculate either. With our alternative way of considering gene length, this issue becomes somewhat unique to a typical case where zero counts must be dealt with. Past that, to simply model zero counts in a traditional manner ignores the fact that there is an event, namely that a gene must have at least one transcript mapped to it, necessary for a non-zero count to be observed. Instead of trying to fit a zero count against the modelled distribution, we include an indicator for each gene, for each lane to aid here. We define T_{gij} as the (observable) indicator that the g^{th} gene has no transcripts mapped to it in the lane corresponding to condition i and replicate j. Eventually we will consider the actual distribution for our observed gene expression, Y_{gij} to be :

$$f(y_{gij}) = \begin{cases} \omega_{gi} \ , \ y_{gij} = 0 \\ (1-\omega_{gi})g(y_{gij}|\mu_{gij},\phi_g)/(1-g(0|\mu_{gij},\phi_g)) \ , \ y_{gij} > 0 \end{cases} \tag{12}$$

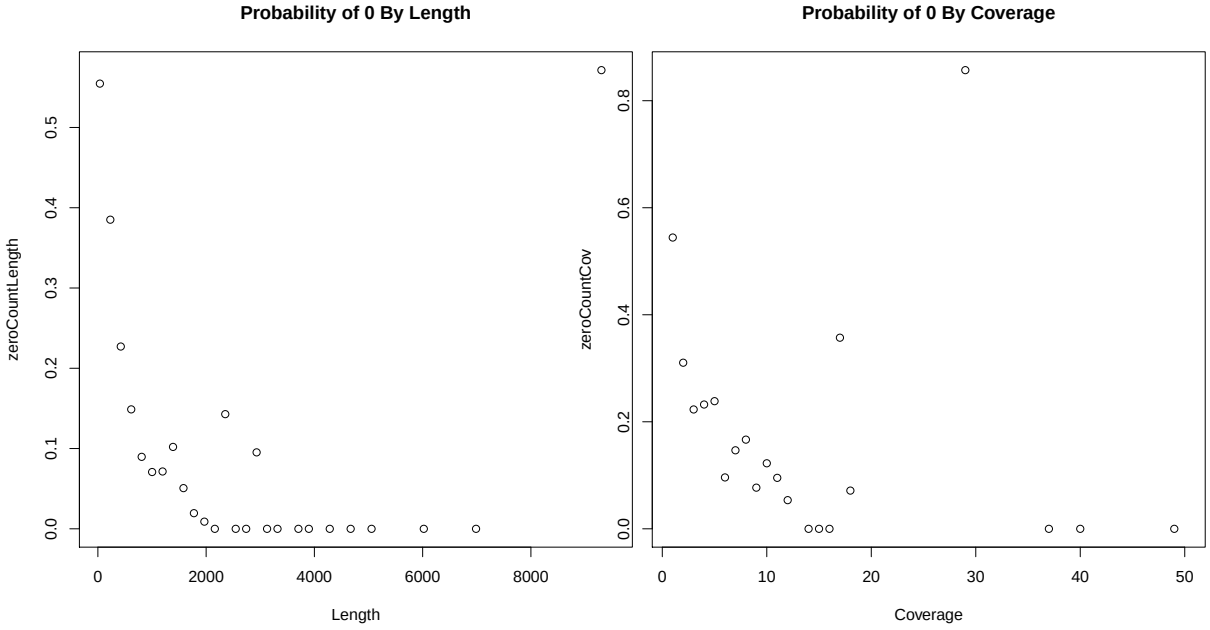
For the final likelihood, it will be convenient to use an alternate form to (12) based on T_{gij},

$$f(y_{gij}) = (\omega_{gi} * \delta_0(y_{gij}))^{T_{gij}} \left((1-\omega_{gi})g(y_{gij}|\mu_{gij},\phi_g)/(1-g(0|\mu_{gij},\phi_g))\right)^{1-T_{gij}} \tag{13}$$

Here, δ_0 gives point mass to 0. The value $\omega_{gi} = \omega(X_{gi}) \in (0,1)$ is a probability estimated with a logistic regression model given the gene length. X_{gi} denotes conditional gene length ($i \in \{0,1\}$). We can use an overall probability of a zero count based on how many of the total observations are zero. However, genes that have any or multiple zero counts are considered to be less reliable. If there are too many zero counts for a gene across the experiment, that gene is omitted and not included in the analysis. There are various values included with our experiment that can be used to help us predict which

genes we expect to see any or multiple zero counts with. For example, genes which have overall lower transcript lengths can be more likely to have zero counts. Also, genes which have low coverage in cases where reads are mapped to them are more likely to have zero counts. The figures (15a) and (15b) use observations binned by gene length coverage. The proportion of genes with zero counts for each bin is calculated, and plotted versus the average value of length or coverage for each bin. The plots show a clear pattern, with larger proportions of zero counts for shorter length genes.

Figure 15: Zero Counts by length and coverage



(a) Zero count proportions are plotted versus their average gene length

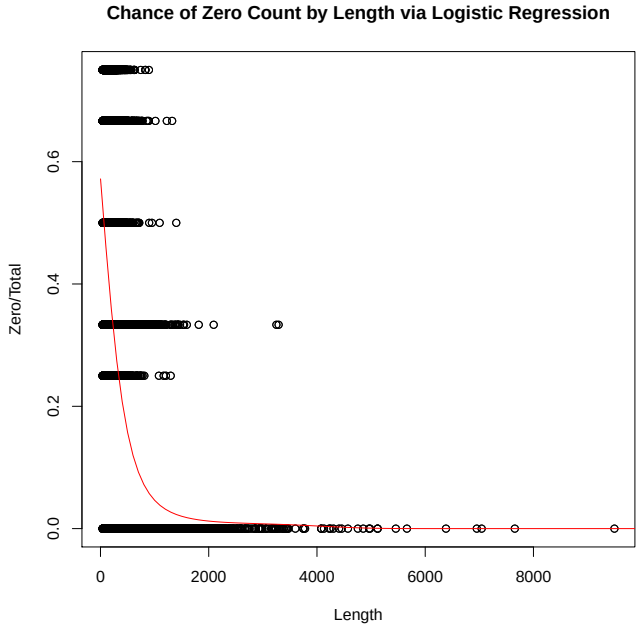
(b) Zero count proportions are plotted versus their average coverage

For each gene, we break the data into control and treatment, for which we have 4 and 3 lanes respectively. We use the earlier mentioned condition-specific gene length X_{gi} as a predictor, and fit a third order logistic regression polynomial model to the zero counts of the data. Figure (16) shows the proportion of zero counts plotted versus conditional gene length, with the estimate for the probability of a zero count as a function of gene

length (over the same range) overlayed. The form of the regression solution is $\text{logit}(p) = .2941 - 4.648 * 10^{-3} X_g + 1.505 * 10^{-6} X_g^2 - 1.754 * 10^{-10} X_g^3$

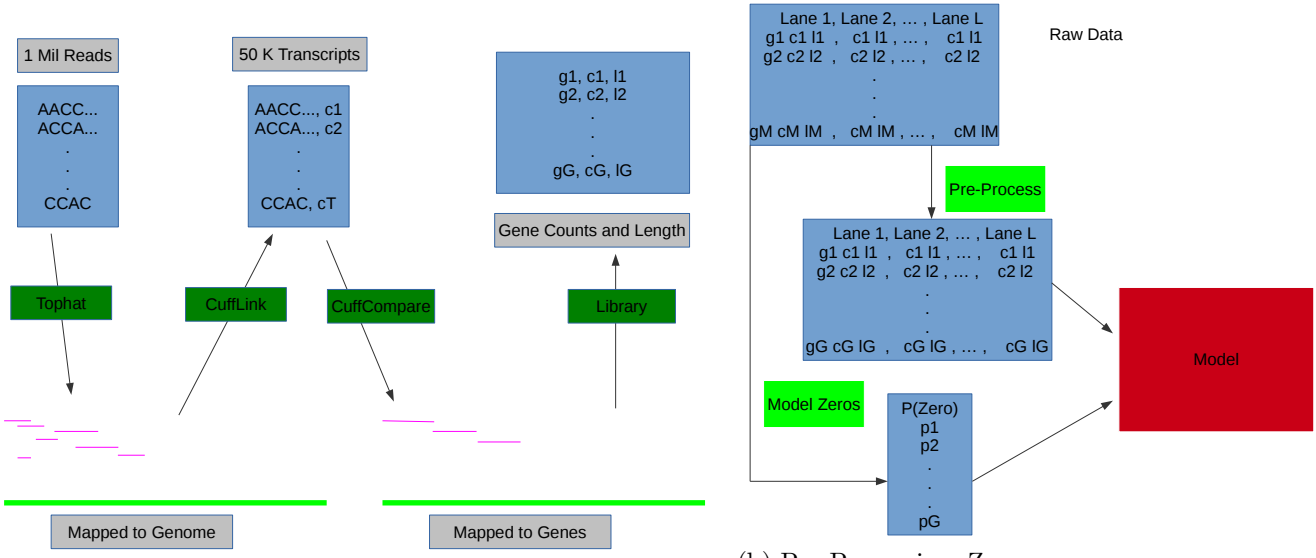
When we wish to estimate ω_g, the probability of a zero count given its gene length, it

Figure 16: Zero Count By Length: Proportion of zero counts by gene length. Overlay estimated probability of zero count in red.



comes from the solid line. Figure (17a) outlines the steps of taking the raw data to count data. This process begins with a text file full of reads (in this case each read has the same length). The notation cg stands for the count observed for gene g, lg the length for that gene, and gg refers to the identifier for the g^{th} gene (i.e. gene name). The final form of this is the count data for a single lane. Figure (17b) begins with the full data, which is the collection of the count data for each lane. The modeling the probability of zeroes based on each gene's length is calculated from the full data. Pre-processing is run by removing genes which have zero counts in more than half the lanes then occurs to arrive at the processed. These are then fed into our models from which we get our results.

Figure 17: Transcript Assembly and Pre-Processing



(a) Transcript Assembly

(b) Pre-Processing, Zero Inflate, Model

3.4 Hierarchical log-linear Model

We will use a variation of models to see what is the most appropriate and effective way to model this data. We use similar notation as before, where $Y_{gij} \sim NB(\mu_{gij}, \phi_g)$. However, we will let $\mu_{gij} = \lambda_{gi} m_{ij} X_g$ where we now consider λ_{gi} to be the rate per base pair (length). We can notice that the above case, where length is not considered, considers the rate of $\lambda_{gi} X_g$, that is to say that the the above rate is the per length rate multiplied by length. However, were we to simply use a Wald type statistic, both methods would yield the same values. We can actually extend this methodology by considering the case where our expected values μ_{gij} are actually random. We utilize the following assumption, $Y_{gij}|\mu_{gij}, \phi_g \sim NB(\mu_{gij}, \phi_g)$ where $\mu_{gij} = \lambda_{gi} m_{ij}^{\beta_2} X_g^{\beta_{g3}}$

We can consider the following models:

The base model, without any manner of DE, is

$$\log(\mu_{gij}) = \beta_{g0} + \beta_2 * \log(m_{ij}) + \beta_{g3} * \log(X_g) \tag{14}$$

The Intercept Shift model uses V_g to introduce a treatment specific shift in intercept.

$$\log(\mu_{gij}) = \beta_{g0} + \beta_{g1} I_i V_g + \beta_2 * \log(m_{ij}) + \beta_{g3} * \log(X_g) \tag{15}$$

The Condition-Specific Gene Length model uses U_g to introduce a condition-specific gene length.

$$\log(\mu_{gij}) = \beta_{g0} + \beta_2 * \log(m_{ij}) + \beta_{g3} * \log(X_g) * (1 - U_g) + \beta_{g4} * U_g(\log(X_{g0})(1 - I_i) + \log(X_{g1}) I_i) \tag{16}$$

The Slope Shift model uses Z_g to introduce a treatment specific shift in slope, or coefficient to the log gene length.

$$\log(\mu_{gij}) = \beta_{g0} + \beta_2 * \log(m_{ij}) + \beta_{g3} * \log(X_g) + \beta_{g5} * Z_g \log(X_g) I_i \tag{17}$$

The Intercept/Condition Shift Model uses both V_g and U_g, this model allows for a treatment specific shift and condition-specific gene length.

$$\log(\mu_{gij}) = \beta_{g0} + \beta_{g1} I_i V_g + \beta_2 * \log(m_{ij}) \beta_{g3} * \log(X_g) * (1 - U_g) + \beta_{g4} * U_g(\log(X_{g0})(1 - I_i) + \log(X_{g1}) I_i) \tag{18}$$

The Intercept/Slope Shift Model uses V_g and Z_g, this models allows for both a treatment specific shift in intercept and slope.

$$\log(\mu_{gij}) = \beta_{g0} + \beta_{g1} I_i V_g + \beta_2 * \log(m_{ij}) + \beta_{g3} * \log(X_g) + \beta_{g5} * Z_g * \log(X_g) I_i \tag{19}$$

When utilizing Bayesian hierarchical models, it is typical to include graphical models to help with the nature of the model. Since we actually have a number of models, we use a single example for the graph, but feel this will be easily extendable should the reader wish to see such a graph for any of the other models. We consider the Intercept Shift

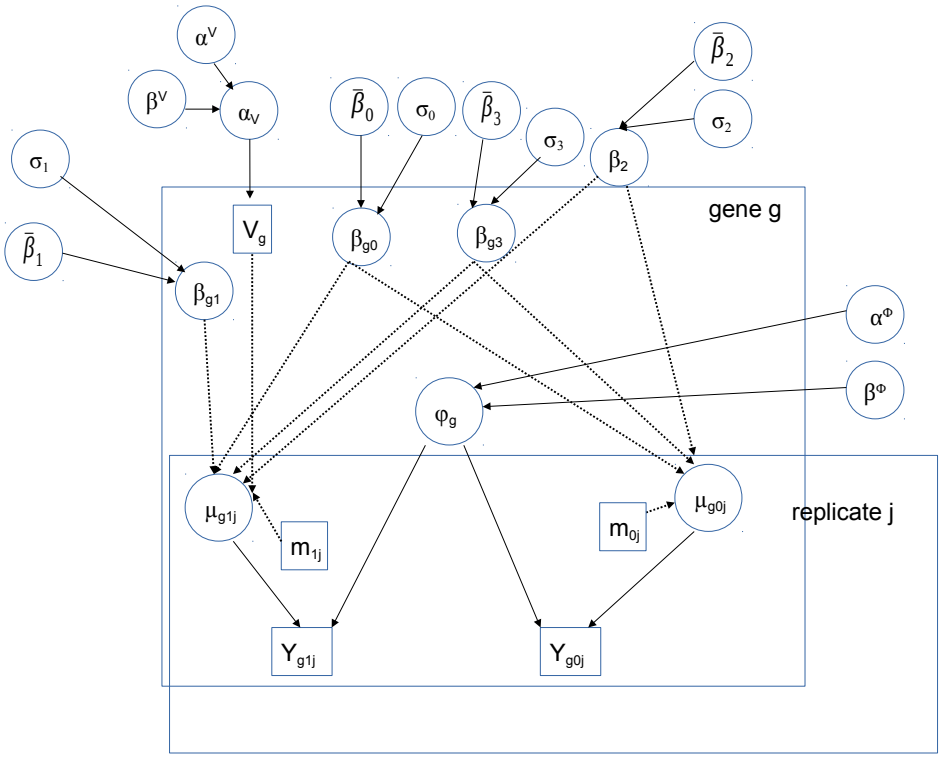


Figure 18: Graphical Model Intercept Shift Model

model (15) as in figure 18:

While we only include the graphical model for one of our models, it is easily extended to any of the others. The portion of the graphical model that is V specific stems from the boxed V, and includes the hyperparameters. One need only replace those with the proper variable from the model. The following should clear up any of the notation which is unclear:

1. An α with a subscript refers to a mixing proportion for the latent variable in the subscript ($V_g \sim \text{Bern}(\alpha_V)$).

2. A superscript on α or β refers to the hyperparameters of a beta distribution ($\alpha_V \sim \text{Beta}(\alpha^V, \beta^V)$).

3. A bar over a beta parameter refers to the mean for a coefficient ($\beta_2 \sim N(\bar{\beta}_2, \sigma_2)$, $\beta_{g3} \sim N(\bar{\beta}_3, \sigma_3)$).

3.4.1 Likelihoods

$$L_1 = \prod_{g=1}^{G} \prod_{i=0}^{1} \prod_{j=1}^{J} \left[\frac{\Gamma(Y_{gij} + \phi_g^{-1})}{\Gamma(\phi_g^{-1})\Gamma(Y_{gij}+1)} \left(\frac{\mu_{gij}}{\phi_g^{-1} + \mu_{gij}} \right)^{Y_{gij}} \left(\frac{1}{1+\mu_{gij}\phi_g} \right)^{\phi_g^{-1}} \right] \tag{20}$$

The likelihood (20) fits zero counts against the distribution directly.

$$L_2 = \prod_{g=1}^{G} \prod_{i=0}^{1} \prod_{j=1}^{J} \left(\left[\frac{\Gamma(Y_{gij} + \phi_g^{-1})}{\Gamma(\phi_g^{-1})\Gamma(Y_{gij}+1)} \left(\frac{\mu_{gij}}{\phi_g^{-1} + \mu_{gij}} \right)^{Y_{gij}} \left(\frac{1}{1+\mu_{gij}\phi_g} \right)^{\phi_g^{-1}} \Big/ \left(1 - \left(\frac{1}{1+\mu_{gij}\phi_g} \right)^{\phi_g^{-1}} \right) \right] * (1-\omega_{gij}) \right)^{1-T_{gij}} \times \omega_{gij}^{T_{gij}} \tag{21}$$

The likelihood (21) fits zero counts against the estimated probabilities based on gene length. We consider the prior distributions for our different types of DE.

3.4.2 Prior Choices

$$\pi_g^V = \pi(\beta_{g1}, V_g | \alpha_V) = \left(\frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-(\beta_{g1})^2/2\sigma_1^2} \right)^{V_g} \alpha_V^{V_g} \left((1-\alpha_V) \right)^{1-V_g} \tag{22}$$

$$\pi(\alpha_V | \alpha^V, \beta^V) = \frac{\alpha_V^{\alpha^V - 1}(1-\alpha_V)^{\beta^V - 1}}{B(\alpha^V, \beta^V)} \tag{23}$$

$$\pi_g^U = \pi(\beta_{g3}, \beta_{g4}, U_g | \alpha_U) = \left[\left(\frac{1}{\sqrt{2\pi\sigma_3^2}} e^{-(\beta_{g3} - \bar{\beta}_3)^2/2\sigma_3^2} \right)^{1-U_g} \left(\frac{1}{\sqrt{2\pi\sigma_3^2}} e^{-(\beta_{g4} - \bar{\beta}_4)^2/2\sigma_4^2} \right)^{U_g} \right] \times \alpha_U^{U_g} (1-\alpha_U)^{1-U_g} \tag{24}$$

$$\pi(\alpha_U|\alpha^U,\beta^U) = \frac{\alpha_U^{\alpha^U-1}(1-\alpha_U)^{\beta^U-1}}{B(\alpha^U,\beta^U)} \tag{25}$$

$$\pi_g^Z = \pi(\beta_{g5}, Z_g|\alpha_Z) = \left(\frac{1}{\sqrt{2\pi\sigma_5^2}}e^{-(\beta_{g5}-\bar{\beta}_5)^2/2\sigma_5^2}\right)^{1-Z_g} \alpha_Z^{Z_g}\left((1-\alpha_Z)\right)^{1-Z_g} \tag{26}$$

$$\pi(\alpha_Z|\alpha^Z,\beta^Z) = \frac{\alpha_Z^{\alpha^Z-1}(1-\alpha_Z)^{\beta^Z-1}}{B(\alpha^Z,\beta^Z)} \tag{27}$$

$$\pi_g^{(l)} = \pi(\beta_{gl}|\bar{\beta}_{gl},\sigma_l) = \frac{1}{\sqrt{2\pi\sigma_l^2}}e^{-(\beta_{gl}-\bar{\beta}_l)/2\sigma_l^2},\ l \in \{0,3\} \tag{28}$$

$$\pi^{(2)} = \pi(\beta_2|\bar{\beta}_2,\sigma_2) = \frac{1}{\sqrt{2\pi\sigma_2^2}}e^{-(\beta_2-\bar{\beta}_2)/2\sigma_2^2} \tag{29}$$

$$\pi_g^\phi = \pi(\phi_g) = (\phi_g)(1-\phi_g) \tag{30}$$

3.4.3 Posterior Inference

At this point, we would like to use the typical posterior expectation to estimate our parameters, and we will use $\hat{\tau}_g^V$ where $\tau_g^V = P(V_g = 0|\boldsymbol{Y})$ to perform multiple hypothesis testing controlling for some FDR value. Should we be considering a model with either U_g or Z_g in it, we would then be considering $\tau_g^U = P(U_g = 0|\boldsymbol{Y})$ and $\tau_g^Z = P(Z_g = 0|\boldsymbol{Y})$. In order to do so, we will utilize Metropolis Hastings. We will now run through each of our models. For each model, we will specify the kernel of the posterior, the portion of the kernel needed to update each variable, and an outline of the steps for the Metropolis Hastings algorthm used for posterior inference.

Metropolis-Hastings: Within the Metropolis-Hastings algorithm, we use a normal distribution with small variance as our proposal distribution. In the case of sampling the overdispersion parameter, ϕ_g, we use a uniform distribution since we need ϕ to be between 0 and 1. We initialize our parameters by simulating values from the prior distribution. For

Figure 19: Metropolis-Hastings:Updating θ

1: **procedure** M-H
2: Simulate θ' from your proposal distribution
3: Calculate $ar = \frac{h(\theta')}{h(\theta)}$, h being the kernel of the posterior
4: $ar' = \min(1, ar)$
5: Update $\theta = \theta'$ with probability ar'

each parameter we update, we generate an alternative value via our proposal distribution, and find the acceptance ratio. Here, the acceptance ratio is the ratio of the kernel of the posterior evaluated for the candidate value (drawn from the proposal) over the kernel of the posterior evaluated for the original value. This value of the acceptance ratio is used to determine the update rule. If we call the function $h(\theta)$ the posterior kernel evaluated at θ, than our acceptance ratio $ar = \frac{h(\theta)}{h(\theta')}$ where θ is the current value of our parameter, and $\theta' = \delta + \theta$ where δ comes from our proposal distribution (i.e. is symmetric about 0). Here, our update rule is $I_U = I(\text{Update}) \sim \text{Bern}(ar')$ where $ar' = min(1, ar)$.

3.4.3.1 Base 1

We refer to Base1 as the model that uses (14) as the form for the mean. There is no consideration of DE gene here. Also, this base model doesn't use any zero inflation. So the kernel of the posterior has the form:

$$L_1 \prod_{g=1}^{G} \left(\pi_g^{\phi} \pi_g^{(3)} \pi_g^{(0)} \right) \pi^{(2)} \tag{31}$$

$$\Pi(\beta_{g0}|\boldsymbol{Y}, \beta_{g1}, \beta_{g3}, \beta_2, \phi_g) \propto \prod_{i=0}^{1} \prod_{j=1}^{J} \left(\frac{\mu_{gij}}{\phi_g^{-1} + \mu_{gij}} \right)^{Y_{gij}} \left(\frac{1}{1 + \mu_{gij}\phi_g} \right)^{\phi_g^{-1}} \frac{1}{\sqrt{2\pi\sigma_0^2}} e^{-(\beta_{g0} - \bar{\beta}_0)^2 / 2\sigma_0^2} \tag{32}$$

$$\Pi(\beta_{g3}|\boldsymbol{Y}, \beta_{g1}, \beta_{g0}, \beta_2, \phi_g) \propto \prod_{i=0}^{1} \prod_{j=1}^{J} \left(\frac{\mu_{gij}}{\phi_g^{-1} + \mu_{gij}} \right)^{Y_{gij}} \left(\frac{1}{1 + \mu_{gij}\phi_g} \right)^{\phi_g^{-1}} \frac{1}{\sqrt{2\pi\sigma_3^2}} e^{-(\beta_{g3} - \bar{\beta}_3)^2 / 2\sigma_3^2} \tag{33}$$

$$\Pi(\beta_2|\boldsymbol{Y},\beta_{g0},\beta_{g1},\beta_{g3},\phi_g) \propto \prod_{g=1}^{G}\prod_{i=0}^{1}\prod_{j=1}^{J}\left(\frac{\mu_{gij}}{\phi_g^{-1}+\mu_{gij}}\right)^{Y_{gij}}\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\frac{1}{\sqrt{2\pi\sigma_2^2}}e^{-(\beta_2-\bar{\beta}_2)^2/2\sigma_2^2} \tag{34}$$

$$\Pi_g(\phi_g|\boldsymbol{Y},\alpha,\beta_{g0},\beta_2,\beta_{g3},\beta_{g1}) \propto \prod_{i=0}^{1}\prod_{j=1}^{J}\frac{\Gamma(Y_{gij}+\phi_g^{-1})}{\Gamma(\phi_g^{-1})\Gamma(Y_{gij}+1)}\left(\frac{\mu_{gij}}{\phi_g^{-1}+\mu_{gij}}\right)^{Y_{gij}}\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\phi_g(1-\phi_g) \tag{35}$$

Our Gibbs sampler:

1. $t = t+1$,
2. Update $\beta_{g0}^{(t)}$ using (32) (M-H),
3. Update $\beta_{g3}^{(t)}$ using (33) (M-H),
4. Update $\beta_2^{(t)}$ using (34) (M-H),
5. Update $\phi_g^{(t)}$ using (35) (M-H).

3.4.3.2 Base 2

The only distinction between Base 2 and Base 1 is the likelihood, which is now L_2 (21).

3.4.3.3 Intercept-Shift

The kernel of the posterior for the Intercept-Shift (15) model is:

$$L_2\prod_{g=1}^{G}\left(\pi_g^V\pi_g^{\phi}\pi_g^{(3)}\pi_g^{(0)}\right)\pi^{(2)}\pi(\alpha_V|\alpha^V,\beta^V) \tag{36}$$

Looking at each of variables to be updated, we consider the portion of the posterior kernel used to update:

$$P(V_g = 1|\beta_{g11}, \boldsymbol{Y}, \alpha_V, \beta_{g0}, \beta_2, \beta_{g3}, \phi_g) \propto$$
$$\prod_{i=0}^{1}\prod_{j=1}^{J}\left(\frac{\mu_{gij}}{\phi_g^{-1}+\mu_{gij}}\right)^{Y_{gij}}\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}} \Bigg/ \left(1-\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\right) \alpha_V \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-(\beta_{g11})^2/2\sigma_l^2}$$
$$\stackrel{set}{=} \boldsymbol{I}$$

$$P(V_g = 0|\beta_{g11}, \boldsymbol{Y}, \alpha_V, \beta_{g0}, \beta_2, \beta_{g3}, \phi_g) \propto$$
$$\prod_{i=0}^{1}\prod_{j=1}^{J}\left(\frac{\mu_{gij}}{\phi_g^{-1}+\mu_{gij}}\right)^{Y_{gij}}\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}} \Bigg/ \left(1-\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\right) (1-\alpha_V)$$
$$\stackrel{set}{=} \boldsymbol{II}$$

(37)

$$\Pi_(\beta_{g1}|\boldsymbol{Y}, \boldsymbol{\beta}_{g(-1)}, \phi_g, \alpha, V_g = 1) \propto$$
$$\prod_{i=0}^{1}\prod_{j=1}^{J}\left(\frac{\mu_{gij}}{\phi_g^{-1}+\mu_{gij}}\right)^{Y_{gij}}\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}} \Bigg/ \left(1-\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\right) \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-(\beta_{g1})^2/2\sigma_1^2}$$
$$\Pi(\beta_{g1}|\boldsymbol{Y}, \boldsymbol{\beta}_{g(-1)}, \phi_g, \alpha, V_g = 0) \propto \frac{1}{\sqrt{2\pi}\sigma_1} e^{-(\beta_{g1})^2/2\sigma_1^2}$$

(38)

$$\Pi(\beta_{g0}|\boldsymbol{Y}, \beta_{g1}, \beta_{g3}, \beta_2, \phi_g, \alpha_V, V_g) \propto$$
$$\prod_{i=0}^{1}\prod_{j=1}^{J}\left(\frac{\mu_{gij}}{\phi_g^{-1}+\mu_{gij}}\right)^{Y_{gij}}\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}} \Bigg/ \left(1-\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\right) \frac{1}{\sqrt{2\pi\sigma_0^2}} e^{-(\beta_{g0}-\bar{\beta}_0)^2/2\sigma_0^2}$$

(39)

$$\Pi(\beta_{g3}|\boldsymbol{Y}, \beta_{g1}, \beta_{g0}, \beta_2, \phi_g, \alpha_V, V_g) \propto$$
$$\prod_{i=0}^{1}\prod_{j=1}^{J}\left(\frac{\mu_{gij}}{\phi_g^{-1}+\mu_{gij}}\right)^{Y_{gij}}\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}} \Bigg/ \left(1-\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\right) \frac{1}{\sqrt{2\pi\sigma_3^2}} e^{-(\beta_{g3}-\bar{\beta}_3)^2/2\sigma_3^2}$$

(40)

$$\Pi(\beta_2|\boldsymbol{Y},\beta_{g0},\beta_{g1},\beta_{g3},\phi_g,\alpha_V,V_g)\propto$$
$$\prod_{g=1}^{G}\prod_{i=0}^{1}\prod_{j=1}^{J}\left(\frac{\mu_{gij}}{\phi_g^{-1}+\mu_{gij}}\right)^{Y_{gij}}\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\Bigg/\left(1-\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\right)\frac{1}{\sqrt{2\pi\sigma_2^2}}e^{-(\beta_2-\bar{\beta}_2)^2/2\sigma_2^2} \tag{41}$$

$$\Pi_g(\phi_g|\boldsymbol{Y},\alpha,\beta_{g0},\beta_2,\beta_{g3},\alpha_V,V_g,\beta_{g1})\propto$$
$$\prod_{i=0}^{1}\prod_{j=1}^{J}\frac{\Gamma(Y_{gij}+\phi_g^{-1})}{\Gamma(\phi_g^{-1})\Gamma(Y_{gij}+1)}\left(\frac{\mu_{gij}}{\phi_g^{-1}+\mu_{gij}}\right)^{Y_{gij}}\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\Bigg/\left(1-\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\right)\phi_g(1-\phi_g) \tag{42}$$

$$\alpha_V\sim\text{Beta}(\alpha^V+\sum_{g=1}^{G}V_g,\beta^V+G-\sum_{g=1}^{G}V_g) \tag{43}$$

So, we proceed with our Gibbs sampler. After getting our initial estimates, $\alpha_V^{(0)},\ V_g^{(0)},\beta_{g0}^{(0)},\ \beta_{g1}^{(0)},\ \beta_2^{(0)},\ \beta_{g3}^{(0)},\ \phi_g^{(0)}$ for $g=1,...,G$, set $t=0$, and then follow these steps:

1. $t=t+1$,
2. Update $\alpha_V^{(t)}$ using (43),
3. Update V_g using $V_g^{(t)}\sim Bern(p_{Vg}^{(t)})$ where $p_{Vg}^{(t)}=\frac{I}{I+II}$ from (37),
4. Update $\beta_{g1}^{(t)}$ using (38) (M-H),
5. Update $\beta_{g0}^{(t)}$ using (39) (M-H),
6. Update $\beta_{g3}^{(t)}$ using (40) (M-H),
7. Update $\beta_2^{(t)}$ using (41) (M-H),
8. Update $\phi_g^{(t)}$ using (42) (M-H).

3.4.3.4 Condition-Specific

The kernel of the posterior for the Condition-Specific (16) model is:

$$L_2 \prod_{g=1}^{G} \left(\pi_g^U \pi_g^{\phi} \pi_g^{(0)} \right) \pi^{(2)} \pi(\alpha_U | \alpha^U, \beta^U) \tag{44}$$

Looking at each of variables to be updated, we consider the portion of the posterior kernel used to update:

$$\begin{aligned}
&P(U_g = 1 | \boldsymbol{Y}, \alpha_U, \beta_{g0}, \beta_2, \beta_{g3}, \beta_{g4}, \phi_g) \propto \\
&\prod_{i=0}^{1} \prod_{j=1}^{J} \left(\frac{\mu_{gij}}{\phi_g^{-1} + \mu_{gij}} \right)^{Y_{gij}} \left(\frac{1}{1 + \mu_{gij}\phi_g} \right)^{\phi_g^{-1}} \Bigg/ \left(1 - \left(\frac{1}{1 + \mu_{gij}\phi_g} \right)^{\phi_g^{-1}} \right) \alpha_U \frac{1}{\sqrt{2\pi\sigma_4^2}} e^{-(\beta_{g4} - \bar{\beta}_4)^2 / 2\sigma_4^2} \\
&\stackrel{set}{=} \boldsymbol{I} \\
&P(U_g = 0 | \boldsymbol{Y}, \alpha_U, \beta_{g0}, \beta_2, \beta_{g3}, \beta_{g4}, \phi_g) \propto \\
&\prod_{i=0}^{1} \prod_{j=1}^{J} \left(\frac{\mu_{gij}}{\phi_g^{-1} + \mu_{gij}} \right)^{Y_{gij}} \left(\frac{1}{1 + \mu_{gij}\phi_g} \right)^{\phi_g^{-1}} \Bigg/ \left(1 - \left(\frac{1}{1 + \mu_{gij}\phi_g} \right)^{\phi_g^{-1}} \right) (1 - \alpha_U) \frac{1}{\sqrt{2\pi\sigma_3^2}} e^{-(\beta_{g3} - \bar{\beta}_3)^2 / 2\sigma_3^2} \\
&\stackrel{set}{=} \boldsymbol{II}
\end{aligned} \tag{45}$$

$$\begin{aligned}
&\Pi(\beta_{g0} | \boldsymbol{Y}, \beta_{g1}, \beta_{g3}, \beta_2, \phi_g, \alpha_U, U_g) \propto \\
&\prod_{i=0}^{1} \prod_{j=1}^{J} \left(\frac{\mu_{gij}}{\phi_g^{-1} + \mu_{gij}} \right)^{Y_{gij}} \left(\frac{1}{1 + \mu_{gij}\phi_g} \right)^{\phi_g^{-1}} \Bigg/ \left(1 - \left(\frac{1}{1 + \mu_{gij}\phi_g} \right)^{\phi_g^{-1}} \right) \frac{1}{\sqrt{2\pi\sigma_0^2}} e^{-(\beta_{g0} - \bar{\beta}_0)^2 / 2\sigma_0^2}
\end{aligned} \tag{46}$$

$$\begin{aligned}
&\Pi(\beta_{g3} | \boldsymbol{Y}, \beta_{g1}, \beta_{g0}, \beta_{g4}, \beta_2, \phi_g, \alpha_U, U_g = 0) \propto \\
&\prod_{i=0}^{1} \prod_{j=1}^{J} \left(\frac{\mu_{gij}}{\phi_g^{-1} + \mu_{gij}} \right)^{Y_{gij}} \left(\frac{1}{1 + \mu_{gij}\phi_g} \right)^{\phi_g^{-1}} \Bigg/ \left(1 - \left(\frac{1}{1 + \mu_{gij}\phi_g} \right)^{\phi_g^{-1}} \right) \frac{1}{\sqrt{2\pi\sigma_3^2}} e^{-(\beta_{g3} - \bar{\beta}_3)^2 / 2\sigma_3^2}
\end{aligned} \tag{47}$$

$$\Pi(\beta_{g3} | \boldsymbol{Y}, \beta_{g1}, \beta_{g0}, \beta_{g4}, \beta_2, \phi_g, \alpha_U, U_g = 1) \propto \frac{1}{\sqrt{2\pi\sigma_3^2}} e^{-(\beta_{g3} - \bar{\beta}_3)^2 / 2\sigma_3^2} \tag{48}$$

$$\Pi(\beta_{g4}|\boldsymbol{Y},\beta_{g1},\beta_{g0},\beta_{g3},\beta_2,\phi_g,\alpha_U,U_g=0)\propto\frac{1}{\sqrt{2\pi\sigma_4^2}}e^{-(\beta_{g4}-\bar{\beta}_4)^2/2\sigma_4^2} \tag{49}$$

$$\Pi(\beta_{g3}|\boldsymbol{Y},\beta_{g1},\beta_{g0},\beta_{g3},\beta_2,\phi_g,\alpha_U,U_g=1)\propto$$
$$\prod_{i=0}^{1}\prod_{j=1}^{J}\left(\frac{\mu_{gij}}{\phi_g^{-1}+\mu_{gij}}\right)^{Y_{gij}}\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\Bigg/\left(1-\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\right)\frac{1}{\sqrt{2\pi\sigma_4^2}}e^{-(\beta_{g4}-\bar{\beta}_4)^2/2\sigma_4^2} \tag{50}$$

$$\Pi(\beta_2|\boldsymbol{Y},\beta_{g0},\beta_{g1},\beta_{g3},\beta_{g4},\phi_g,\alpha_U,U_g)\propto$$
$$\prod_{g=1}^{G}\prod_{i=0}^{1}\prod_{j=1}^{J}\left(\frac{\mu_{gij}}{\phi_g^{-1}+\mu_{gij}}\right)^{Y_{gij}}\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\Bigg/\left(1-\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\right)\frac{1}{\sqrt{2\pi\sigma_2^2}}e^{-(\beta_2-\bar{\beta}_2)^2/2\sigma_2^2} \tag{51}$$

$$\Pi_g(\phi_g|\boldsymbol{Y},\alpha,\beta_{g0},\beta_2,\beta_{g3},\beta_{g4},\alpha_U,U_g,\beta_{g1})\propto$$
$$\prod_{i=0}^{1}\prod_{j=1}^{J}\frac{\Gamma(Y_{gij}+\phi_g^{-1})}{\Gamma(\phi_g^{-1})\Gamma(Y_{gij}+1)}\left(\frac{\mu_{gij}}{\phi_g^{-1}+\mu_{gij}}\right)^{Y_{gij}}\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\Bigg/\left(1-\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\right)$$
$$\times\,\phi_g(1-\phi_g) \tag{52}$$

$$\alpha_U\sim\text{Beta}(\alpha^U+\sum_{g=1}^{G}U_g,\beta^U+G-\sum_{g=1}^{G}U_g) \tag{53}$$

So, we proceed with our Gibbs sampler as follows. After getting our initial estimates, $\alpha_U^{(0)},\ U_g^{(0)},\beta_{g0}^{(0)},\ \beta_{g1}^{(0)},\ \beta_2^{(0)},\ \beta_{g3}^{(0)},\beta_{g4}^{(0)},\ \phi_g^{(0)}$ for $g=1,...,G$, set $t=0$, and then follow these steps:

1. $t=t+1$,
2. Update $\alpha_U^{(t)}$ using (53),
3. Update U_g using $U_g^{(t)}\sim Bern(p_{Ug}^{(t)})$ where $p_{Ug}^{(t)}=\frac{I}{I+II}$ from (45),
4. Update $\beta_{g0}^{(t)}$ using (46) (M-H),

5. Update $\beta_{g3}^{(t)}$ using (48) if $U_g^{(t)} = 1$, or (47) if $U_g^{(t)} = 0$,

6. Update $\beta_{g4}^{(t)}$ using (50) if $U_g^{(t)} = 1$, or (49) if $U_g^{(t)} = 0$,

7. Update $\beta_2^{(t)}$ using (51) (M-H),

8. Update $\phi_g^{(t)}$ using (52) (M-H).

3.4.3.5 Slope-Shift

The kernel of the posterior for the Intercept-Shift (17) model is:

$$L_2 \prod_{g=1}^{G} \left(\pi_g^Z \pi_g^\phi \pi_g^{(3)} \pi_g^{(0)} \right) \pi^{(2)} \pi(\alpha_Z | \alpha^Z, \beta^Z) \tag{54}$$

Looking at each of variables to be updated, we consider the portion of the posterior kernel used to update:

$$\begin{aligned}
&P(Z_g = 1 | \boldsymbol{Y}, \alpha_Z, \beta_{g0}, \beta_2, \beta_{g3}, \beta_{g5}, \phi_g) \propto \\
&\prod_{i=0}^{1} \prod_{j=1}^{J} \left(\frac{\mu_{gij}}{\phi_g^{-1} + \mu_{gij}} \right)^{Y_{gij}} \left(\frac{1}{1 + \mu_{gij}\phi_g} \right)^{\phi_g^{-1}} \Bigg/ \left(1 - \left(\frac{1}{1 + \mu_{gij}\phi_g} \right)^{\phi_g^{-1}} \right) \alpha_Z \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-(\beta_{g5})^2 / 2\sigma_5^2} \\
&\stackrel{set}{=} \boldsymbol{I} \\
&P(Z_g = 0 | \boldsymbol{Y}, \alpha_Z, \beta_{g0}, \beta_2, \beta_{g3}, \beta_{g5}, \phi_g) \propto \\
&\prod_{i=0}^{1} \prod_{j=1}^{J} \left(\frac{\mu_{gij}}{\phi_g^{-1} + \mu_{gij}} \right)^{Y_{gij}} \left(\frac{1}{1 + \mu_{gij}\phi_g} \right)^{\phi_g^{-1}} \Bigg/ \left(1 - \left(\frac{1}{1 + \mu_{gij}\phi_g} \right)^{\phi_g^{-1}} \right) (1 - \alpha_Z) \\
&\stackrel{set}{=} \boldsymbol{II}
\end{aligned} \tag{55}$$

$$\Pi_(\beta_{g5}|\boldsymbol{Y},\phi_g,\beta_{g1},\beta_{g3},\beta_2,\alpha_Z,Z_g=1)\propto$$
$$\prod_{i=0}^{1}\prod_{j=1}^{J}\left(\frac{\mu_{gij}}{\phi_g^{-1}+\mu_{gij}}\right)^{Y_{gij}}\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\Bigg/\left(1-\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\right)\frac{1}{\sqrt{2\pi\sigma_5^2}}e^{-(\beta_{g5})^2/2\sigma_5^2}$$
$$\Pi_(\beta_{g5}|\boldsymbol{Y},\phi_g,\beta_{g1},\beta_{g3},\beta_2,\alpha_Z,Z_g=0)\propto\frac{1}{\sqrt{2\pi}\sigma_5}e^{-(\beta_{g5})^2/2\sigma_5^2}$$

(56)

$$\Pi(\beta_{g0}|\boldsymbol{Y},\beta_{g5},\beta_{g3},\beta_2,\phi_g,\alpha_Z,Z_g)\propto$$
$$\prod_{i=0}^{1}\prod_{j=1}^{J}\left(\frac{\mu_{gij}}{\phi_g^{-1}+\mu_{gij}}\right)^{Y_{gij}}\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\Bigg/\left(1-\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\right)\frac{1}{\sqrt{2\pi\sigma_0^2}}e^{-(\beta_{g0}-\bar{\beta}_0)^2/2\sigma_0^2}$$

(57)

$$\Pi(\beta_{g3}|\boldsymbol{Y},\beta_{g5},\beta_{g0},\beta_2,\phi_g,\alpha_Z,Z_g)\propto$$
$$\prod_{i=0}^{1}\prod_{j=1}^{J}\left(\frac{\mu_{gij}}{\phi_g^{-1}+\mu_{gij}}\right)^{Y_{gij}}\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\Bigg/\left(1-\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\right)\frac{1}{\sqrt{2\pi\sigma_3^2}}e^{-(\beta_{g3}-\bar{\beta}_3)^2/2\sigma_3^2}$$

(58)

$$\Pi(\beta_2|\boldsymbol{Y},\beta_{g0},\beta_{g5},\beta_{g3},\phi_g,\alpha_Z,Z_g)\propto$$
$$\prod_{g=1}^{G}\prod_{i=0}^{1}\prod_{j=1}^{J}\left(\frac{\mu_{gij}}{\phi_g^{-1}+\mu_{gij}}\right)^{Y_{gij}}\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\Bigg/\left(1-\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\right)\frac{1}{\sqrt{2\pi\sigma_2^2}}e^{-(\beta_2-\bar{\beta}_2)^2/2\sigma_2^2}$$

(59)

$$\Pi_g(\phi_g|\boldsymbol{Y},\alpha,\beta_{g0},\beta_2,\beta_{g3},\alpha_Z,Z_g,\beta_{g5})\propto$$
$$\prod_{i=0}^{1}\prod_{j=1}^{J}\frac{\Gamma(Y_{gij}+\phi_g^{-1})}{\Gamma(\phi_g^{-1})\Gamma(Y_{gij}+1)}\left(\frac{\mu_{gij}}{\phi_g^{-1}+\mu_{gij}}\right)^{Y_{gij}}\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\Bigg/\left(1-\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\right)\phi_g(1-\phi_g)$$

(60)

$$\alpha_Z\sim\text{Beta}(\alpha^Z+sum_{g=1}^{G}Z_g,\beta^Z+G-sum_{g=1}^{G}Z_g)\tag{61}$$

So, we proceed with our Gibbs sampler as follows. After getting our initial estimates, $\alpha_Z^{(0)},\ Z_g^{(0)}, \beta_{g0}^{(0)},\ \beta_{g1}^{(0)},\ \beta_2^{(0)},\ \beta_{g3}^{(0)},\ \phi_g^{(0)}$ for $g = 1, ..., G$, set $t = 0$, and then follow these steps:

1. $t = t + 1$,
2. Update $\alpha_Z^{(t)}$ using (61),
3. Update Z_g using $Z_g^{(t)} \sim Bern(p_{Zg}^{(t)})$ where $p_{Zg}^{(t)} = \frac{I}{I+II}$ from (55),
4. Update $\beta_{g5}^{(t)}$ using (56) (M-H),
5. Update $\beta_{g0}^{(t)}$ using (57) (M-H),
6. Update $\beta_{g3}^{(t)}$ using (58) (M-H),
7. Update $\beta_2^{(t)}$ using (59) (M-H),
8. Update $\phi_g^{(t)}$ using (60) (M-H).

3.4.3.6 Intercept-Condition

The kernel of the posterior for the Condition-Specific (18) model is:

$$L_2 \prod_{g=1}^{G} \left(\pi_g^U \pi_g^V \pi_g^{\phi} \pi_g^{(0)} \right) \pi^{(2)} \pi(\alpha_U | \alpha^U, \beta^U) \pi(\alpha_V | \alpha^V, \beta^V) \tag{62}$$

Looking at each of variables to be updated, we consider the portion of the posterior kernel used to update:

$$P(V_g = 1|\beta_{g11}, \boldsymbol{Y}, \alpha_V, \beta_{g0}, \beta_2, \beta_{g3}, \phi_g, \alpha_U, U_g) \propto$$
$$\prod_{i=0}^{1}\prod_{j=1}^{J}\left(\frac{\mu_{gij}}{\phi_g^{-1}+\mu_{gij}}\right)^{Y_{gij}}\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}} \Bigg/ \left(1-\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\right)\alpha_V\frac{1}{\sqrt{2\pi\sigma_1^2}}e^{-(\beta_{g11})^2/2\sigma_l^2}$$
$$\stackrel{set}{=} \boldsymbol{I}$$

$$P(V_g = 0|\beta_{g11}, \boldsymbol{Y}, \alpha_V, \beta_{g0}, \beta_2, \beta_{g3}, \phi_g, \alpha_U, U_g) \propto$$
$$\prod_{i=0}^{1}\prod_{j=1}^{J}\left(\frac{\mu_{gij}}{\phi_g^{-1}+\mu_{gij}}\right)^{Y_{gij}}\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}} \Bigg/ \left(1-\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\right)(1-\alpha_V)$$
$$\stackrel{set}{=} \boldsymbol{II}$$

(63)

$$P(U_g = 1|\boldsymbol{Y}, \alpha_U, \beta_{g0}, \beta_2, \beta_{g3}, \beta_{g4}, \phi_g, \alpha_V, V_g) \propto$$
$$\prod_{i=0}^{1}\prod_{j=1}^{J}\left(\frac{\mu_{gij}}{\phi_g^{-1}+\mu_{gij}}\right)^{Y_{gij}}\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}} \Bigg/ \left(1-\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\right)\alpha_U\frac{1}{\sqrt{2\pi\sigma_4^2}}e^{-(\beta_{g4}-\bar{\beta}_4)^2/2\sigma_4^2}$$
$$\stackrel{set}{=} \boldsymbol{I}$$

$$P(U_g = 0|\boldsymbol{Y}, \alpha_U, \beta_{g0}, \beta_2, \beta_{g3}, \beta_{g4}, \phi_g, \alpha_V, V_g) \propto$$
$$\prod_{i=0}^{1}\prod_{j=1}^{J}\left(\frac{\mu_{gij}}{\phi_g^{-1}+\mu_{gij}}\right)^{Y_{gij}}\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}} \Bigg/ \left(1-\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\right)(1-\alpha_U)\frac{1}{\sqrt{2\pi\sigma_3^2}}e^{-(\beta_{g3}-\bar{\beta}_3)^2/2\sigma_3^2}$$
$$\stackrel{set}{=} \boldsymbol{II}$$

(64)

$$\Pi_(\beta_{g1}|\boldsymbol{Y}, \boldsymbol{\beta}_{g(-1)}, \phi_g, \alpha, V_g = 1, \alpha_U, U_g) \propto$$
$$\prod_{i=0}^{1}\prod_{j=1}^{J}\left(\frac{\mu_{gij}}{\phi_g^{-1}+\mu_{gij}}\right)^{Y_{gij}}\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}} \Bigg/ \left(1-\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\right) \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-(\beta_{g1})^2/2\sigma_1^2}$$
$$\Pi(\beta_{g1}|\boldsymbol{Y}, \boldsymbol{\beta}_{g(-1)}, \phi_g, \alpha, V_g = 0, \alpha_U, U_g) \propto \frac{1}{\sqrt{2\pi}\sigma_1} e^{-(\beta_{g1})^2/2\sigma_1^2} \tag{65}$$

$$\Pi(\beta_{g0}|\boldsymbol{Y}, \beta_{g1}, \beta_{g3}, \beta_2, \phi_g, \alpha_V, V_g, \alpha_U, U_g) \propto$$
$$\prod_{i=0}^{1}\prod_{j=1}^{J}\left(\frac{\mu_{gij}}{\phi_g^{-1}+\mu_{gij}}\right)^{Y_{gij}}\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}} \Bigg/ \left(1-\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\right) \frac{1}{\sqrt{2\pi\sigma_0^2}} e^{-(\beta_{g0}-\bar{\beta}_0)^2/2\sigma_0^2} \tag{66}$$

$$\Pi(\beta_{g3}|\boldsymbol{Y}, \beta_{g1}, \beta_{g0}, \beta_{g4}, \beta_2, \phi_g, \alpha_U, U_g = 0, \alpha_V, V_g) \propto$$
$$\prod_{i=0}^{1}\prod_{j=1}^{J}\left(\frac{\mu_{gij}}{\phi_g^{-1}+\mu_{gij}}\right)^{Y_{gij}}\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}} \Bigg/ \left(1-\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\right) \frac{1}{\sqrt{2\pi\sigma_3^2}} e^{-(\beta_{g3}-\bar{\beta}_3)^2/2\sigma_3^2} \tag{67}$$

$$\Pi(\beta_{g3}|\boldsymbol{Y}, \beta_{g1}, \beta_{g0}, \beta_{g4}, \beta_2, \phi_g, \alpha_U, U_g = 1, \alpha_V, V_g) \propto$$
$$\frac{1}{\sqrt{2\pi\sigma_3^2}} e^{-(\beta_{g3}-\bar{\beta}_3)^2/2\sigma_3^2} \tag{68}$$

$$\Pi(\beta_{g4}|\boldsymbol{Y}, \beta_{g1}, \beta_{g0}, \beta_{g3}, \beta_2, \phi_g, \alpha_U, U_g = 0, \alpha_V, V_g) \propto$$
$$\frac{1}{\sqrt{2\pi\sigma_4^2}} e^{-(\beta_{g4}-\bar{\beta}_4)^2/2\sigma_4^2} \tag{69}$$

$$\Pi(\beta_{g3}|\boldsymbol{Y}, \beta_{g1}, \beta_{g0}, \beta_{g3}, \beta_2, \phi_g, \alpha_U, U_g = 1, \alpha_V, V_g) \propto$$
$$\prod_{i=0}^{1}\prod_{j=1}^{J}\left(\frac{\mu_{gij}}{\phi_g^{-1}+\mu_{gij}}\right)^{Y_{gij}}\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}} \Bigg/ \left(1-\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\right) \frac{1}{\sqrt{2\pi\sigma_4^2}} e^{-(\beta_{g4}-\bar{\beta}_4)^2/2\sigma_4^2} \tag{70}$$

$$\Pi(\beta_2|\boldsymbol{Y},\beta_{g0},\beta_{g1},\beta_{g3},\beta_{g4},\phi_g,\alpha_U,U_g,\alpha_V,V_g)\propto$$
$$\prod_{g=1}^{G}\prod_{i=0}^{1}\prod_{j=1}^{J}\left(\frac{\mu_{gij}}{\phi_g^{-1}+\mu_{gij}}\right)^{Y_{gij}}\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\Bigg/\left(1-\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\right)\frac{1}{\sqrt{2\pi\sigma_2^2}}e^{-(\beta_2-\bar{\beta}_2)^2/2\sigma_2^2} \tag{71}$$

$$\Pi_g(\phi_g|\boldsymbol{Y},\alpha,\beta_{g0},\beta_2,\beta_{g3},\beta_{g4},\alpha_U,U_g,\beta_{g1},\alpha_V,V_g)\propto$$
$$\prod_{i=0}^{1}\prod_{j=1}^{J}\frac{\Gamma(Y_{gij}+\phi_g^{-1})}{\Gamma(\phi_g^{-1})\Gamma(Y_{gij}+1)}\left(\frac{\mu_{gij}}{\phi_g^{-1}+\mu_{gij}}\right)^{Y_{gij}}\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\Bigg/\left(1-\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\right)\phi_g(1-\phi_g) \tag{72}$$

$$\alpha_U\sim\text{Beta}(\alpha^U+\sum_{g=1}^{G}U_g,\beta^U+G-\sum_{g=1}^{G}U_g) \tag{73}$$

$$\alpha_V\sim\text{Beta}(\alpha^V+\sum_{g=1}^{G}V_g,\beta^V+G-\sum_{g=1}^{G}V_g) \tag{74}$$

So, we proceed with our Gibbs sampler as follows. After getting our initial estimates, $\alpha_U^{(0)},\ U_g^{(0)},\beta_{g0}^{(0)},\ \beta_{g1}^{(0)},\ \beta_2^{(0)},\ \beta_{g3}^{(0)},\beta_{g4}^{(0)},\ \phi_g^{(0)}$ for $g=1,...,G$, set $t=0$, and then follow these steps:

1. $t=t+1$,
2. Update $\alpha_V^{(t)}$ using (74) ,
3. Update $\alpha_U^{(t)}$ using (73),
4. Update V_g using $V_g^{(t)}\sim Bern(p_{Vg}^{(t)})$ where $p_{Vg}^{(t)}=\frac{I}{I+II}$ from (63),
5. Update $\beta_{g1}^{(t)}$ using (65) (M-H),
6. Update U_g using $U_g^{(t)}\sim Bern(p_{Ug}^{(t)})$ where $p_{Ug}^{(t)}=\frac{I}{I+II}$ from (64),
7. Update $\beta_{g3}^{(t)}$ using (68) if $U_g^{(t)}=1$, or (67) if $U_g^{(t)}=0$ (M-H),

8. Update $\beta_{g4}^{(t)}$ using (70) if $U_g^{(t)} = 1$, or (69) if $U_g^{(t)} = 0$ (M-H),

9. Update $\beta_{g0}^{(t)}$ using (66) (M-H),

10. Update $\beta_2^{(t)}$ using (51) (M-H),

11. Update $\phi_g^{(t)}$ using (72) (M-H).

3.4.3.7 Intercept-Slope

The kernel of the posterior for the Condition-Specific (19) model is:

$$L_2 \prod_{g=1}^{G} \left(\pi_g^V \pi_g^Z \pi_g^\phi \pi_g^{(3)} \pi_g^{(0)} \right) \pi^{(2)} \pi(\alpha_V | \alpha^V, \beta^V) \pi(\alpha_Z | \alpha^Z, \beta^Z) \tag{75}$$

Looking at each of variables to be updated, we consider the portion of the posterior kernel used to update:

$$\begin{aligned}
&P(V_g = 1 | \beta_{g11}, \boldsymbol{Y}, \alpha_V, \beta_{g0}, \beta_2, \beta_{g3}, \phi_g, \alpha_Z, Z_g) \propto \\
&\prod_{i=0}^{1} \prod_{j=1}^{J} \left(\frac{\mu_{gij}}{\phi_g^{-1} + \mu_{gij}} \right)^{Y_{gij}} \left(\frac{1}{1 + \mu_{gij}\phi_g} \right)^{\phi_g^{-1}} \Bigg/ \left(1 - \left(\frac{1}{1 + \mu_{gij}\phi_g} \right)^{\phi_g^{-1}} \right) \alpha_V \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-(\beta_{g11})^2 / 2\sigma_l^2} \\
&\stackrel{set}{=} \boldsymbol{I} \\
&P(V_g = 0 | \beta_{g11}, \boldsymbol{Y}, \alpha_V, \beta_{g0}, \beta_2, \beta_{g3}, \phi_g, \alpha_Z, Z_g) \propto \\
&\prod_{i=0}^{1} \prod_{j=1}^{J} \left(\frac{\mu_{gij}}{\phi_g^{-1} + \mu_{gij}} \right)^{Y_{gij}} \left(\frac{1}{1 + \mu_{gij}\phi_g} \right)^{\phi_g^{-1}} \Bigg/ \left(1 - \left(\frac{1}{1 + \mu_{gij}\phi_g} \right)^{\phi_g^{-1}} \right) (1 - \alpha_V) \\
&\stackrel{set}{=} \boldsymbol{II}
\end{aligned} \tag{76}$$

$$P(Z_g = 1|\boldsymbol{Y}, \alpha_Z, \beta_{g0}, \beta_2, \beta_{g3}, \beta_{g5}, \phi_g, \alpha_V, V_g) \propto$$
$$\prod_{i=0}^{1}\prod_{j=1}^{J}\left(\frac{\mu_{gij}}{\phi_g^{-1}+\mu_{gij}}\right)^{Y_{gij}}\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}} \Bigg/ \left(1-\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\right)\alpha_Z\frac{1}{\sqrt{2\pi\sigma_1^2}}e^{-(\beta_{g5})^2/2\sigma_5^2}$$
$$\stackrel{set}{=} \boldsymbol{I}$$

$$P(Z_g = 0|\boldsymbol{Y}, \alpha_Z, \beta_{g0}, \beta_2, \beta_{g3}, \beta_{g5}, \phi_g, \alpha_V, V_g) \propto$$
$$\prod_{i=0}^{1}\prod_{j=1}^{J}\left(\frac{\mu_{gij}}{\phi_g^{-1}+\mu_{gij}}\right)^{Y_{gij}}\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}} \Bigg/ \left(1-\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\right)(1-\alpha_Z)$$
$$\stackrel{set}{=} \boldsymbol{II}$$

(77)

$$\Pi_(\beta_{g1}|\boldsymbol{Y}, \boldsymbol{\beta}_{g(-1)}, \phi_g, \alpha, V_g = 1, \alpha_Z, Z_g) \propto$$
$$\prod_{i=0}^{1}\prod_{j=1}^{J}\left(\frac{\mu_{gij}}{\phi_g^{-1}+\mu_{gij}}\right)^{Y_{gij}}\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}} \Bigg/ \left(1-\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\right)\frac{1}{\sqrt{2\pi\sigma_1^2}}e^{-(\beta_{g1})^2/2\sigma_1^2}$$
$$\Pi(\beta_{g1}|\boldsymbol{Y}, \boldsymbol{\beta}_{g(-1)}, \phi_g, \alpha, V_g = 0, \alpha_Z, Z_g) \propto \frac{1}{\sqrt{2\pi}\sigma_1}e^{-(\beta_{g1})^2/2\sigma_1^2}$$

(78)

$$\Pi_(\beta_{g5}|\boldsymbol{Y}, \phi_g, \beta_{g1}, \beta_{g3}, \beta_2, \alpha_Z, Z_g = 1, \alpha_V, V_g) \propto$$
$$\prod_{i=0}^{1}\prod_{j=1}^{J}\left(\frac{\mu_{gij}}{\phi_g^{-1}+\mu_{gij}}\right)^{Y_{gij}}\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}} \Bigg/ \left(1-\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\right)\frac{1}{\sqrt{2\pi\sigma_5^2}}e^{-(\beta_{g5})^2/2\sigma_5^2}$$
$$\Pi_(\beta_{g5}|\boldsymbol{Y}, \phi_g, \beta_{g1}, \beta_{g3}, \beta_2, \alpha_Z, Z_g = 0, \alpha_V, V_g) \propto \frac{1}{\sqrt{2\pi}\sigma_5}e^{-(\beta_{g5})^2/2\sigma_5^2}$$

(79)

$$\Pi(\beta_{g3}|\boldsymbol{Y}, \beta_{g1}, \beta_{g0}, \beta_2, \phi_g, \alpha_V, V_g, \alpha_Z, Z_g) \propto$$
$$\prod_{i=0}^{1}\prod_{j=1}^{J}\left(\frac{\mu_{gij}}{\phi_g^{-1}+\mu_{gij}}\right)^{Y_{gij}}\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}} \Bigg/ \left(1-\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\right)\frac{1}{\sqrt{2\pi\sigma_3^2}}e^{-(\beta_{g3}-\bar{\beta}_3)^2/2\sigma_3^2}$$

(80)

$$\Pi(\beta_{g0}|\boldsymbol{Y},\beta_{g1},\beta_{g3},\beta_2,\phi_g,\alpha_V,V_g,alpha_Z,Z_g)\propto$$
$$\prod_{i=0}^{1}\prod_{j=1}^{J}\left(\frac{\mu_{gij}}{\phi_g^{-1}+\mu_{gij}}\right)^{Y_{gij}}\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\Bigg/\left(1-\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\right)\frac{1}{\sqrt{2\pi\sigma_0^2}}e^{-(\beta_{g0}-\bar{\beta}_0)^2/2\sigma_0^2} \tag{81}$$

$$\Pi(\beta_2|\boldsymbol{Y},\beta_{g0},\beta_{g1},\beta_{g3},\beta_{g4},\phi_g,\alpha_U,U_g,\alpha_V,V_g,\alpha_Z,Z_g)\propto$$
$$\prod_{g=1}^{G}\prod_{i=0}^{1}\prod_{j=1}^{J}\left(\frac{\mu_{gij}}{\phi_g^{-1}+\mu_{gij}}\right)^{Y_{gij}}\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\Bigg/\left(1-\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\right)\frac{1}{\sqrt{2\pi\sigma_2^2}}e^{-(\beta_2-\bar{\beta}_2)^2/2\sigma_2^2} \tag{82}$$

$$\Pi_g(\phi_g|\boldsymbol{Y},\alpha,\beta_{g0},\beta_2,\beta_{g3},\beta_{g4},\alpha_Z,Z_g,\beta_{g1},\alpha_V,V_g)\propto$$
$$\prod_{i=0}^{1}\prod_{j=1}^{J}\frac{\Gamma(Y_{gij}+\phi_g^{-1})}{\Gamma(\phi_g^{-1})\Gamma(Y_{gij}+1)}\left(\frac{\mu_{gij}}{\phi_g^{-1}+\mu_{gij}}\right)^{Y_{gij}}\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\Bigg/\left(1-\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\right)\phi_g(1-\phi_g) \tag{83}$$

$$\alpha_Z\sim\text{Beta}(\alpha^Z+\sum_{g=1}^{G}Z_g,\beta^Z+G-\sum_{g=1}^{G}Z_g) \tag{84}$$

$$\alpha_V\sim\text{Beta}(\alpha^V+\sum_{g=1}^{G}V_g,\beta^V+G-\sum_{g=1}^{G}V_g) \tag{85}$$

So, we proceed with our Gibbs sampler as follows. After getting our initial estimates, $\alpha_V^{(0)},\ V_g^{(0)},\alpha_Z^{(0)},\ Z_g^{(0)},\beta_{g0}^{(0)},\ \beta_{g1}^{(0)},\ \beta_2^{(0)},\ \beta_{g3}^{(0)},\beta_{g4}^{(0)},\ \phi_g^{(0)}$ for $g=1,...,G$, set $t=0$, and then follow these steps:

1. $t=t+1$,
2. Update $\alpha_V^{(t)}$ using (85),
3. Update $\alpha_Z^{(t)}$ using (84),
4. Update V_g using $V_g^{(t)}\sim Bern(p_{Vg}^{(t)})$ where $p_{Vg}^{(t)}=\frac{I}{I+II}$ from (76),

5. Update $\beta_{g1}^{(t)}$ using (78) (M-H),

6. Update Z_g using $Z_g^{(t)} \sim Bern(p_{Zg}^{(t)})$ where $p_{Zg}^{(t)} = \frac{I}{I+II}$ from (77),

7. Update $\beta_{g5}^{(t)}$ using (79) (M-H),

8. Update $\beta_{g0}^{(t)}$ using (81) (M-H),

9. Update $\beta_{g3}^{(t)}$ using (80) (M-H),

10. Update $\beta_2^{(t)}$ using (82) (M-H),

11. Update $\phi_g^{(t)}$ using (72) (M-H).

3.4.4 Hyper Parameters

So, when we consider genes which are not DE, we let $E[\mu_{gij}] = \lambda m_{ij} X_g$ where λ is the typical rate per base (length) for tags to be assigned to a gene. As such, we set $\lambda = 1/\sum_{g=1}^{G} X_g$. Here, we notice that m_{ij} is a changing value, and for reasons which will be clear soon, we use $\bar{m} = \sum_{i=0}^{1}\sum_{j=1}^{J_i} m_{ij}/(J_0 + J_1)$ and try to get $E[\lambda_{gi}] = \lambda$, $E[m_{ij}^{\beta_2}] = m_{ij}$, and $E[X_g^{\beta_{g3}}] = S_g$. However, in the case of library size, since we have unique values per replicate, we will have to instead make it so for the typical library size we have observed, $\bar{m}$ and desire to have $E[\bar{m}^{\beta_2}]$. We will use three indicator variables, each with its own interpretation, to describe DE genes. The first is a shift in the mean, represented by V_g, second is a shift in the rate per gene length, represented by Z_g, and the final value indicates whether the summary or conditional gene length is appropriate, U_g. One last indicator that will be used throughout the following is I_i = I(i=1), which will indicate when we are using treatment terms.

Here, for $l = 0, 1, 2, 3, 4, 5$ $\beta_{gl} \sim N(\bar{\beta}_l, \sigma_l^2)$, where $\bar{\beta}_1 = \mathbf{0}$. However, we give β_{g1} a mixture distribution. Historically, this type of mixture has been called a spike and slab prior. When we do the updating for β_{g1}, we will use the method outlined in Kuo and Mallick (1998). Similarly, we use exactly one of the two methods for calculating gene length per gene. This means that exactly one of the values, β_{g3} or β_{g4} must be zero. Meaning that $U_g = 0$ uses a model with one gene length, and $U_g = 1$ uses a model with one gene length per condition. In order to get our hyperparameters, we use the fact that for some constant c and some W where c^W has a lognormal distribution and $E[W] = \mu$ and $Var(W) = \sigma^2$

$$\begin{aligned} W\log(c) \sim N(\mu\log(c), \sigma^2\log(c)^2)\ ,\ E[c^W] = e^{\mu\log(c)+\sigma^2\log(c)^2/2} \overset{set}{=} c \\ \Rightarrow \mu\log(c) + \sigma^2\log(c)^2/2 = \log(c) \Rightarrow \mu = 1 - \log(c)\sigma^2/2 \end{aligned} \tag{86}$$

That is to say, if $E[X] = 1 - \log(x)\sigma^2/2$, then $E[c^X] = c$. In the above model, we see that the role of c is played by $\bar{m}$ and X_g while the role of W is played by β_2 and β_{g3} respectively. As we choose our variance hyperparameters, we should notice that if β_2 or β_{g3} are negative, this leads to a decrease in μ_{gi} as either the length or the library size increase. This is counter to what we wish, so we choose the variances to satisfy

$$0 = \mu - 3\sigma = 1 - \log(c)\sigma^2/2 - 3\sigma \Rightarrow -(\log(c)/2)\,\sigma^2 - 3\sigma + 1 = 0 \Rightarrow \sigma = \frac{\sqrt{9 + 2log(c)} - 3}{\log(c)} \tag{87}$$

We see that (87) is always positive. In order to get the hyperparameters for λ_{gi}, we must notice that our approach has $\log(\lambda_{gi}) = \beta_{g0} + \beta_{g1}I(i = 1)$, where we have a mixture distribution $\beta_{g1} = \beta_{g10}I(V_g = 1) + 0I(V_g = 0)$ where $\beta_{g10} \sim N(0, \sigma_1^2)$. Also, we say that $V_g \sim Bern(\alpha_V)$ where α represents the chance that any given gene is DE, and we let $\alpha \sim Beta(\alpha^V, \beta^V)$ where we choose $\alpha_1,\ \alpha_2$ to satisfy $E[\alpha] = .05$ or some other desired

value. Should it be the case that $V_g = 0$, i.e. out gene is not differentially expressed,

$$E[\lambda_{g1}] = e^{\mu_0+\sigma_0^2} \overset{set}{=} \lambda \;\Rightarrow \mu_0 = \log(\lambda) - \sigma_0^2/2 \tag{88}$$

We also need to consider the distribution for ϕ_g. We say that $\phi_g \sim \text{Beta}(2,2)$.

3.5 Model Assessment

We will use a common criterion, deviance information criterion (DIC), to judge which models fit well. This is possible in our modeling scheme since we use hierarchical models, a reduced model and a full model which also includes the transcript length of the gene. In general, the deviance is defined as

$$D(\boldsymbol{\theta}) = -2 * \ln(p(\boldsymbol{X}|\boldsymbol{\theta})) =$$
$$-2 * \sum_{g=1}^{G}\sum_{i=0}^{1}\sum_{j=1}^{J} \ln\left(\left[\left(\frac{\Gamma(Y_{gij}+\phi_g^{-1})}{\Gamma(\phi_g^{-1})\Gamma(Y_{gij}+1)}\left(\frac{\mu_{gij}}{\phi_g^{-1}+\mu_{gij}}\right)^{Y_{gij}}\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\right/\left(1-\left(\frac{1}{1+\mu_{gij}\phi_g}\right)^{\phi_g^{-1}}\right) \times (1- \tag{89}$$

Here $\boldsymbol{\theta} = \{\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_G, \phi_1, ..., \phi_2\}$ where $\boldsymbol{\mu}_g$ is the vector of mean values for the g^{th} gene. We also have

$$\bar{D} = E^{\theta|\boldsymbol{Y}}[D(\theta)] \tag{90}$$

Also, we have

$$p_D = \bar{D} - D(\bar{\theta}) \tag{91}$$

We arrive at

$$\text{DIC} = D(\bar{\theta}) + 2p_D \tag{92}$$

In cases where we have missing data, which is what we consider of latent variables indicating differential expression (V_g, U_g, Z_g), calculating the DIC can be a bit more tricky. Celeux et al. (2006) summarizes applying a number of variations to the calculation of DIC

in such cases. One such variation, denoted DIC_4, involves tracking the parameter θ given the possible values of the missing data, and then integrating them out. This is supposed to more accurately capture the dimensional penalty. In our results, we include DIC as the traditional calculation, and then use DIC_4 to track this missing data specific formulation. We notice that in the case of the base models, which do not track differentially expressed data (and therefore are free of missing data), there is no such calculation to make.

Another popular criterion for model assessment is the Logged Psuedo Marginal Likelihood (LPML). In order to calculate this value, we will use the Conditional Predictive Ordinates (CPO). CPOs are popular in part because they offer a method for cross validation without needing to run a model multiple times on subcollections of the data. CPOs have been established as a posterior predictive approach in (Geisser (1980), Gelfand et al. (1992),Chen et al. (2000)).

$$\text{CPO}_{gij} = f(y_{gij}|\boldsymbol{y}_{-gij}) \tag{93}$$

We say that $\boldsymbol{y}_{-gij}$ is the full data with the expression count of the g^{th} gene in the j^{th} replicate of the i^{th} condition removed. As was shown in (Chen et al. (2000)), (93) can be rewritten as

$$\text{CPO}_{gij} = \left(\int \frac{1}{f(y_{gij}|\boldsymbol{\theta})} f(\boldsymbol{\theta}|\boldsymbol{y}) d\boldsymbol{\theta} \right)^{-1} \tag{94}$$

We can use (94) this to come up with an MCMC estimate (Chen et al. (2000)) for CPO of

$$\widehat{\text{CPO}}_{gij} = \left(\frac{1}{L} \sum_{l=1}^{L} \frac{1}{f(y_{gij}|\boldsymbol{\theta}_l)} \right)^{-1} \tag{95}$$

We use L as the total number (after burn in) of iterations for our MCMC, θ_l being the parameters simulated in the l^{th} step of the MCMC algorithm. We now tie this back into

the LPML as

$$\text{LPML} = \sum_{g=1}^{G} \sum_{j=1}^{J} \sum_{i=0}^{1} \log(\widehat{\text{CPO}}_{gij}) \tag{96}$$

3.6 Results

3.6.1 Simulation Study

In order to assess how well the models perform when it is appropriate, we use a simulation study. To start with, we use the actual library sizes and gene lengths from our collected dataset. We use a sampling with replacement, where each gene from our experiment is given the same sampling probability. We use the gene lengths from the selected genes, and from here generate $\beta_{g0}, \beta_{g1}, V_g, \beta_2, \beta_{g3},$ and ϕ_g. Then we calculate the appropriate mean values and generate data based on μ_{gij}, ϕ_g.

In order to move forward, we let $\boldsymbol{D}$ be our dataset, where we have G' actual observed genes. We will track the gene summary length (X_g) and condition-specific gene length (X_{gi}). We outline the steps for generating the data for our simulation. We detail generating a small dataset which is DE from only V_g in two conditions. The first is a gene specific overdispersion parameter (ϕ_g), and the second is a shared overdispersion parameter (ϕ). This can be easily extended into generating datasets with genes that are DE from only U_g, as well as changing the conditions to a gene specific intercept term β_{g0} versus a shared intercept term β_0. This encompasses all types of simulation used in our study. The following summarizes the simulation with a gene specific overdispersion parameter:

1. Choose $G' = 1000$

2. For each $g = 1, 2, ..., G'$, draw $X_g,\ X_{g0},\ X_{g1}$

3. Simulate $\alpha_V, \beta_{g0}, \beta_{g1}, V_g, \beta_2, \beta_{g3}$, and ϕ_g from (23), (28), (23), (22), (29) and (30)

Table 4: AUC comparison for over fitting, under fitting, and properly fitting the overdispersion parameter

(a) Model V, Small Sample

Model\TRUE	ϕ_g	ϕ
ϕ_g	0.7171	0.7161
ϕ	0.7130	0.7209

(b) Model U, Small Sample

Model\TRUE	ϕ_g	ϕ
ϕ_g	0.6242	0.6274
ϕ	0.6220	0.6202

(c) Model V, Large Sample

Model\TRUE	ϕ_g	ϕ
ϕ_g	0.8522	0.8553
ϕ	0.8540	0.8555

(d) Model U, Large Sample

Model\TRUE	ϕ_g	ϕ
ϕ_g	0.6564	0.6489
ϕ	0.6470	0.6463

4. Calculate μ_{gij}

5. Generate $Y_{gij} \sim NB(\mu_{gij}, \phi_g)$

The following summarizes the simulation with a shared overdispersion parameter:

1. Choose $G' = 1000$

2. For each $g = 1, 2, ..., G'$, draw $X_g,\ X_{g0},\ X_{g1}$

3. Simulate $\alpha_V, \beta_{g0}, \beta_{g1}, V_g, \beta_2,$ and β_{g3} from (23), (28), (23), (22), and (29). We generate a single , $\phi \sim B(2, 2)$

4. Calculate μ_{gij}

5. Generate $Y_{gij} \sim NB(\mu_{gij}, \phi_g)$

Once the data is simulated, apply our models to it, and see how well the model performs in classification. To do this, we will look at the AUC for the ROC curve. The tables have a truth (columns) and how they are modeled (rows).

We can see from table 4 that there is no real effect in our simulation by over or under fitting with respect to the ϕ parameter. In fact, within each of our tables, the amount of

Table 5: AUC comparison for over fitting, under fitting, and properly fitting the intercept term

(a) Model V, Small Sample

Model\TRUE	β_{g0}	β_0
β_{g0}	0.6293	0.7204
β_0	0.6857	0.7108

(b) Model U, Small Sample

Model\TRUE	β_{g0}	β_0
β_{g0}	0.6227	0.6277
β_0	0.6061	0.6191

(c) Model V, Large Sample

Model\TRUE	β_{g0}	β_0
β_{g0}	0.8559	0.8607
β_0	0.7503	0.7948

(d) Model U, Large Sample

Model\TRUE	β_{g0}	β_0
β_{g0}	0.6564	0.6489
β_0	0.6470	0.6463

variation for our AUC values is smaller than our standard deviations for the AUC. Along these lines, we feel less worried about our choice of how to model ϕ with our real data.

We can see from 5 that there is a substantial effect for over and under fitting of our intercept term. Also, it seems like the effect changes by setting. That is to say that the model U seems to be a bit less effected by over or underfitting then it's counterpart, V. Further, we notice that for model V, small data is modeled with the highest AUC when the intercept parameter is study wide, while the large data has the highest AUC when the parameter is gene specific. Also, for both small and large data, the effect of overfitting seems gives a higher AUC than underfitting. For the model U, there is no real change to the AUC values accross conditions of over/under fitting. For the small data, we see that the only real change in AUC seems come from the case of underfitting.

From the simulation, we can see: First, the simulation study makes us feel as though there is no real multicollinearity issues within our simulation scheme. Second, we feel confident that the way in which we choose to model ϕ, gene specific or study wide, shouldn't have a significant impact on our classification. Finally, over/underfitting the intercept term in our model can have drastic results. Especially for the V model. For both models

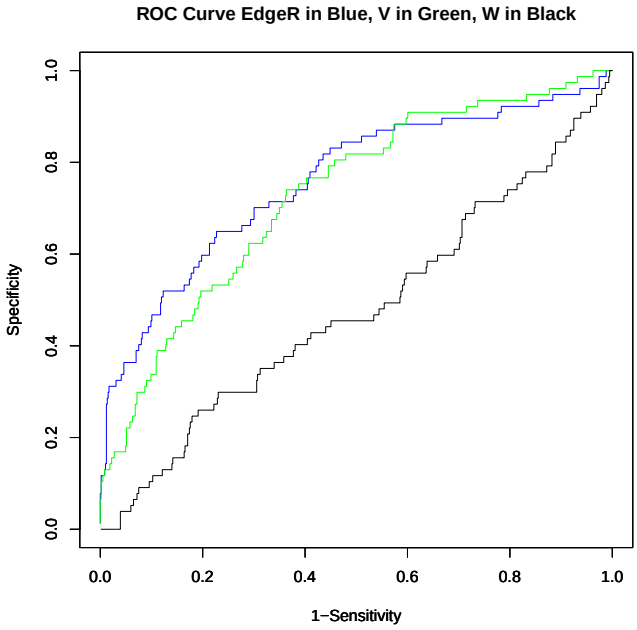


Figure 20: Comparison of ROC curves

considered, using larger sample sizes seems to lower the effect of over/under fitting, and when using small sample sizes.

3.6.2 Real Data

We use the table (6) to summarize the results for our models:

Table 6: Results: DIC, dimensional penalty and LPML for Models: For β_2, library size coefficient is shared across all genes

Model	DIC	pD	DIC_4	pD_4	LPML
Base1	328538.4	3919.004	NA	NA	-200976.1
Base2	274348.6	3519.642	NA	NA	-176603.9
V	273532.8	3576.893	273513	3535.35	-176541.8
U	288168.8	18757.12	277529.2	8117.51	-176402.6
Z	286447.1	8588.704	284185.4	7368.577	-294363.3
VU	271802	2608.148	271697.9	2555.93	-176306.6
VZ	287091.4	9205.695	285761.7	6383.039	-296668.5

Here, we note that Base1 refers to the model where we use the negative binomial distribution to fit the zero counts, while Base2 refers to instead using a zero inflated approach. We notice that the increase in DIC is due to the change in the estimates of the mean parameter in cases where zero values are present. We notice that using DIC directly leads to the conclusion that the best model is using VU together, and that the next best model uses just V. According to DIC, using summary gene length is prefered, while according LPML using the condition-specific gene length to fit the probability of a zero count is prefered. For both criterion, the best models are of VU, V and U, although the order changes from DIC to LPML. We give our formal recommendations based on these values in the conclusion section.

3.6.2.1 Comparing New Methods With Existing Methods

We wish to see how well our approach agrees or disagrees with other conventional methods. We wish to compare the genes considered DE from method to method. The best apples to apples way to do so is to look at the top number of genes as ranked by the different approaches. We use the top 500 to get the following results.

First, we focus on similarities. 7 summarizes the intersection of the top 500 genes for the 3 conventional methods considered. We see that the largest overlap exists between DESeq and edgeR at just over 70%, while the least overlap is just under 40% between baySeq and edgeR. We then look at the intersection of the genes for all 3 conventional methods (188) and the union of all methods (880). These represent the genes present in every method and present in at least one method respectively. We compare the top 500 genes as output by our 3 top models with these two sets, and notice that the smallest

overlap exists with the Intercept-Shift model, while both models including the condition-specific gene length have roughly the same overlap. The former includes 15% of the of the 188 intersect set and has roughly 30% of their genes in the 880 union set, while the later has in the 75% range for the intersect set and 65% range of their genes in the union set. Using a higher granularity, and comparing our output to each method individually, the same basic trend exists. The Intercept-Shift model has less overlap with each individual method than the other two models with condition-specific gene length. Overall, we can see from these tables that the two models with condition-specific gene length actually have top ranked genes coinciding with the conventional methods more than the Intercept-Shift model. This is counter intuitive, since these two models incorporate a different way of modeling the genes than any of the conventional methods. In a sense, this is very encouraging. The most overlap between any method with a conventional method is edgeR with DESeq (.702). The next largest overlap is .568 between the Condition-Specific model and DESeq. The overlap between edgeR and DESeq with either model with the condition-specific gene length in it is in the range of 50% This shows us that our methods with condition specific gene length find a very similar set of genes called DE as the conventional methodologies.

Table 7: Intersection of top 500 genes existing methods

	edgeR	DESEQ	baySeq
edgeR		351	199
DESeq	(.702)		266
baySeq	(.382)	(.532)	

Figure 21 allows us to look at the lengths and trend of lengths for the genes as ranked by our methods and existing methods. We choose to exclude all models involving the Z indicator function as they have such poor DIC values. We look at the top 500 genes as

Table 8: Intersection of top 500 genes our models with summary of existing methods

	V	U	VU
Intersect Exist (188)	29(.150)	141(.750)	146(.770)
Union Exist (880)	157(.314)	339(.678)	321(.642)

Table 9: Intersection of top 500 genes our models with existing methods

	V	U	VU
edgeR	60(.120)	272(.544)	253(.506)
DESeq	101(.202)	284(.568)	275(.550)
baySeq	101(.202)	184(.368)	192(.384)

ranked by each method. We look at the top gene, top 2 genes, and so on, each time taking the average length of the top genes selected. We then plot this by the number of top genes considered. We look to see if there is a decreasing trend in the number of top genes, as that indicates a method that favors long genes. We also look to see how the methods compare, if any of them find clearly longer or shorter genes DE, or if they are somewhat similar.

We see that all of our models seem to have no real trend, while edgeR has a downward trend. Also, all of our models are below edgeR. This suggests there is less length bias for the results of our models. Overall, we notice that models with U tend to select longer genes. This summary length is an averae of the control and treatment lanes. In order for U to select a gene as DE, it has to be DE in such a manner that the gene lengths differ from one condition to another. Along those lines, this method will be more effective for longer genes. The indicator V has no such characteristic. The lengths of the plot suggest that while U selects longer genes than V, the UV model is somewhere in between. They also have different interpretations, as V finds a difference in rate per length regadless of consistency of gene length, while U finds genes which have the same rate per length, but strongly different gene lengths by condition. We feel including V in our final model as VU,

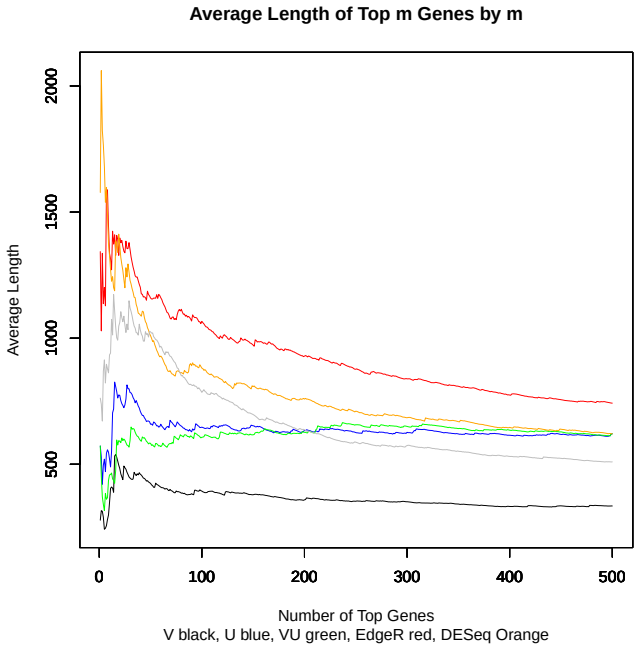


Figure 21: Plot of average lengths comparison of top selected genes among our three models and EdgeR

the number 2 candidate as ranked by DIC, is best for this reason, even over the model with the best DIC, U.

3.7 Conclusion

We have derived a model which allows for multiple considerations of types of DE genes. We used a simulation study to assess the effects of over/under fitting on the overdispersion parameter and intercept term. While the overdispersion parameter fitting didn't seem to effect the AUC, we found a large effect in the intercept term. For large samples, there isn't a negative change in AUC due to overfitting, while there is for underfitting. We suggest fitting a gene specific intercept term. The simulation study allows us to assess the effects of over fitting and under fitting, but do not offer insight as to which one is correct. In terms of interpretation, our intercept term is very similar to the gene rates

λ_{g0} and λ_{g1} found in EdgeR. If the intercept term was shared, this would be equivalent to reducing these gene rates to a single rate. For that reason, we suggest fitting a gene specific intercept term.

We applied our model to a dataset with a small sample size. From the DIC and LPML, we can assess the advantage of using a zero inflated model versus allowing zero counts to be modelled by the negative binomial distribution in the difference of Base1 and Base2. Based sole on the DIC, we would choose the model for U alone, while LPML favors the model VU. Also, V and VU are the top two models for both criteria. We can also notice that the DIC suggests a large improvement in using this zero inflated model present in all but the Base1 model. Part of the benefit of using the model with U present is that we allow for the zero inflated process to model zeroes based on conditional-specific length, as opposed to the summary gene length used in all models (other than Base1) excluding U. As U alone requires genes to have regions activated from control to treatment with differing lengths, it is not a great model by itself. Also using V allows us to find genes that are differentially expressed in a different sense. Also, the model for VU has the best DIC and best LPML, and in terms of gene length, the only model that seems to favor shorter genes than VU is V, which isn't the best model by any other criterion. We also noticed that using a condition-specific length to estimate the probability of a zero count is a benefit to using a condition specific gene length. It is for this reason that we consider the best choice for a model to be VU, and suggest that future models consider the possibility of differing condition-specific lengths in their implementation. Using the condition-specific gene length measure seemed to choose genes with a typically larger length than the Intercept Shift model, but both seem to have less bias towards larger genes than the conventional methods. Combining this trait with the overlap similarity of

the set of genes called DE, our results suggest that the ordering of genes called DE has less of a relationship with the length of the genes. By accounting for gene length in estimating the chance of zero counts, as well as in the estimation of the mean counts, we return a similar set of genes as the methods considered while seeming to have results more robust to the length of the genes.

3.8 Future Work

The methodology we present here is used when only either technical or biological replicates are available (of course biological replicates are prefered, but due limitations experiments have been run without any biological replicates). Great work has been done outlining ways in which this new technology can very cost effectively run more meaningful experiments by desiging a blocked model in order to utilize both technical and biological replicates. One issue that could come up moreso with these types of approaches is coverage depth. Clustering in gene study data is typically performed based on gene expression data. We notice that with our model, one can consider the U_g term, and the τ_g^U values to identify genes which seem to have differing gene lengths by condition. This could identify a subgroup of genes which have different regions activated in different conditions. Also, one could use the gene length effect (β_{g3}), overdispersion parameters ϕ_g, and intercept terms β_{g0} to cluster genes. This would offer different insight than simply clustering based on the count expression.

We also notice that our work suggests that different regions of a gene may be active during different conditions. It is possible that the same gene length from condition to condition could be over different regions of the gene. Along these lines, it would be useful to find an overall location estimate for the transcripts mapped to a given gene in addition

to gene length. This would allow one to find additional ways to find DE genes based on differing gene regions. It is important to note that simply using a condition-specific length and location wouldn't necessarily be sufficient for all possible conditional regions of a gene. Using a scan statistics approach, or additional measures could be useful here. Since our transcripts are assembled from single reads, comparing the reads associated with a single gene over differing conditions could be another method for finding DE genes in this sense.

Chapter 4

Conclusion

We built onto the foundation from Yuan and Kendziorski in the microarray setting. We add SOM methodologies which can identify the true number of clusters, and arrive at a set of possible cluster numbers, much like constructing a feasible set. For each cluster number in this set, initial points for an EM algorithm can be used in the popular mixture of normals assumption for the data. We see this not just reduces large computation times, but adds another, more robust way of seeing which clusterings fit the data than using some AIC or BIC or other criteria based on the convergence of an EM with a specified number of cluster, k. Here, k isn't actually known, so some upper bound K is chosen, and the true k is chosen by the criteria.

Beyond simply aiding the EM machinery from the Unified Method, we also develop our own machinery based on the SOM algorithm and less restrictive assumptions to find DE genes. From our simulations, we notice that in general the semi-parametric classifier outperforms the non-parametric classifier. Also, the classifier based on the alternative to the SOM Ward method seems to outperform the SOMWard method. The existing methods, the assumptions for which the simulated data falls into, outperform even our

best SOM based approach within the simulations. Focusing on the comparison to the Unified Method, the SOM based method performs well in classification for our examples, the form of which cannot be estimated using the classical EM approach. Finally, we notice that the clustering results found via the SOM Ward is best suited for the clustering analysis performed on our DE genes, while the results from the alternative to the SOM Ward are best used to classify the genes as DE and EE. We offer a more robust alternative to finding the true number of clusters and fitting an EM in the mixture of normal case, and offer methodology based on less restrictive assumptions which, by construction, are not as effected by mis-specification of cluster numbers.

In the case of RNA-Seq data, we offer a new conceptualization of gene length. Based on this concept, we introduced a different manner in which genes can be DE. We then developed a model which incorporates gene length, both in estimation of mean and the chance of a zero count, and allows for genes to be differentiated in a shift in intercept, shift in slope, and condition-specific gene length manner. We use real data to show compare our length bias to that of EdgeR. We also use a simulation study to see the effect of over and under fitting, as well as to assess any effect on classification due to using models allowing multiple types of DE. We see that adding types of DE to the model does not effect the AUC negatively. We also see that the condition-specific gene length model does not typically call genes DE when they come from the intercept-shift model, but the intercept-shift model does call genes DE when they come from the condition-specific gene length model. This coincides with the belief that some genes found DE using popular models could be DE in the sense that different regions of the genes are activated in the control and treatment conditions.

Bibliography

Anders, S., Huber, W., 2010. Differential expression analysis for sequence count data. Genome Biology 11.

Blankenberg, D., Von Kuster, G., Coraor, N., Ananda, G., Lazarus, R., Mangan, M., Nekrutenko, A., Taylor, J., 2010. Galaxy: a web-based genome analysis tool for experimentalists. Current Protocols in Molecular Biology .

Celeux, G., Forbes, F., Robert, C., Titterington, D., 2006. Deviance information criteria for missing data models. Bayesian Analysis 1 (4), 651–674.

Chen, M. H., Shao, Q. M., Ibrahim, J., 2000. Monte carlo methods in bayesian computation. New York; Springer .

Covell, D., Wallqvist, A., Rabow, A., Thanki, N., 2003. Molecular classification of cancer: Unsupervised self-organizing map analysis of gene expression microarray data. Mol Cancer Ther .

Dempster, A., Laird, N., Rubin, D., 1977. Maximum likelihood from incomplete data via the em algorithm. Journal of the Royal Statistical Society, Series B 39 (1), 1–38.

Devroye, L., Penrod, C. S., 1984. The consistency of automatic kernel density estimate. The Annals of Statistics 12, 1231–1249.

Fisher, R. A., 1922. On the interpretation of chi-squared from contingency tables, and calculation of p. Journal of the Royal Statistical Society 85 ((1)), 87–94.

Fujita, P., Rhead, B., Zweig, A., Hinrichs, A., Karolchik, D., Cline, M., Goldman, M., Barber, G., Clawson, H., Coelho, A., Diekhans, M., Dreszer, T., Giardine, B., Harte, R., Hillman-Jackson, J., Hsu, F., Kirkup, V., Kuhn, R., Learned, K., Li, C., Meyer, L., Pohl, A., Raney, B., Rosenbloom, K., Smith, K., Haussler, D., Kent, W., 2010. The ucsc genome browser database: update 2011. Nucleic Acids Res. .

Geisser, S., 1980. In discussion of g.e.p box paper entitled: Sampling and bayes' inference in scientific modemodel and robustness. Stat. Soc A (143), 416–417.

Gelfand, A., Dey, D., Chang, H., 1992. Model determination using predictive distributions with implementation via sampling-based methods (with discussion). Tech Report Stanford University, Stanford, California , 462.

Giardine, B., Riemer, C., Hardison, R., Burhans, R., Elnitski, L., Shah, P., Zhang, Y., Blankenberg, D., I, A., Taylor, J., Miller, W., Kent, W., Nekrutenko, A., 2005. Galaxy: a platform for interactive large-scale genome analysis. Genome Research .

Goecks, J., Nekrutenko, A., Taylor, J., Team, T. G., 2010. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. Genome Biol. 11.

Golub, T. R., SLonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J., Coller, H., Loh, M. L., Downing, J., Caliguiuri, M. A., Bloomfield, C. D., Lander, E. S., October 1999. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. Science 286, 531–537.

Hardcastle, T., Kelly, K., 2010. bayseq: empirical bayesian methods for identifying differential expression in sequence count data. BMC Bioinformatic 11, 422.

Hartley, H., 1958. Maximum likelihood estimation from incomplete data. Biometrics 14, 174–194.

Hsu, A., Tang, S., Halgamuge, S., 2003. An unsupervised hierarchical dynamic self-organizing approach to cancer class discovery and marker gene identification in microarray data. Bioinformatics .

Izenman, A., 1991. Recent developments in nonparametric density estimation. American Statistical Association 86, 205–224.

Kaufman, L., Rousseeuw, P., 1987. Clustering by means of medoids, in statistical data analysis based on the l1norm and related methods .

Kohonen, T., 1982. Self-organized formation of topologically correct feature maps. Biol. Cybern 43, 59–69.

Kuo, L., Mallick, B., 1998. Variable selection for regression models. The Indian Journal of Statistics 60, 65–81.

Langmead, B., Trapnell, C., Pop, M., Salzberg, S., 2009. Ultrafast and memory-efficient alignment of short dna ssequence to the human genom. Genome Biol .

Li, H., Lovci, M., Kwon, Y., Rosenfeld, M., Fu, X., Yeo, G., 2008. Determination of tag density required for digital transcriptome analysis: application to an androgen-sensitive prostate cancer model. Proc Natl Acad Sci USA 105, 20179–20184.

MacQueen, J., 1967. Some methods for classification and analysis of multivariate observations. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability .

Parzen, E., 1962. On estimation of a probability density function and mode. Annals of Mathematical Statistics 33, 10651076.

Rosenblatt, M., 1956. Remarks on some nonparametric estimates of a density function. Annals of Mathematical Statistics 27, 832837.

Scott, D., 1985a. Average shifted histograms: Effective nonparametric density estimators in several dimensions. The Annals of Statistics 13, 1024–1040.

Scott, D., 1985b. Frequency polygons. Journal of the American Statistical Association 80, 348–354.

Scott, D. W., Thompson, J. R., 1983. Probability density esti- mation in higher dimensions. Computer Science and Statistics Proceedings of the Fifteenth Symposium on the Interface, ed. J. E. Gentle, Amsterdam: North-Holland, 173–179.

Smyth, G., 2004. Linear models and empirical bayes methods for assessing differential expression in microarray experiments. Statistical Applications in Genetics and Molecular Biology 3.

Smyth, G., Robinson, M., 2007a. Moderated statistical tests for assessing differences in tag abundance. Bioinformatics 23, 2881–2887.

Smyth, G., Robinson, M., 2007b. Small-sample estimation of negative binomial dispersion, with applications to sage data. Biostatistics 9, 321–332.

Sturn, A., Quackenbush, J., Trajonoski, Z., 2002. Gengene: cluster analysis of microarray data. Bioinformatics .

Tai, Y. C., Speed, T. P., 2006. A multivariate empriical bayes statistic for replicated microarray time course data. The Annals of Statistics 34, 2387–2412.

Toronena, P., Kolehmainen, M., Wong, G., Castren, E., 1999. Analysis of gene expression data using self-organizing maps. FEBS Letters .

Trapnell, C., Pachter, L., Salzberg, S., 2009. Tophat: discovering splice junctions with rna-seq. Bioinformatics 25, 1105–1111.

Trapnell, C., Williams, B. A., Pertea, G., Mortazavi, A., Kwan, G., van Baren, M. J., Salzberg, S. L., Wold, B. J., Pachter, L., 2010. Transcript assembly and quantification by rna-seq reveals unannotated transcripts and isoform switching during cell differentiation. Nature Biotechnology 28, 511–515.

Tusher, V., Tibshirani, R., Chu, G., 2001. Significance analysis of microarrays applied to the ionizing radiation response. Proc Natl Acad Sci USA 98, 5116–5121.

Wang, K., Singh, D., Zeng, Z., Coleman, S. J., Huang, Y., 2010. Marna-seq: Accurate mapping of rna-seq reads for splice junction discovery. Nucleic Acids Res. .

Xiao, L., Wang, K., Teng, Y., Zhang, J., 2003. Component plane presentation integrated self-organizing map for microarray data analysis. FEBS Letters .

Yuan, M., Kendziorski, C., 2006. A unified approach for simultaneous gene clustering and differential expression identification. Biometrics 62, 1089–1093.