Title
Time-Course Analysis and Clustering of Gene Expression Data

Author
DeGraaf, Stephanie

Publication Date
2020

Peer reviewed|Thesis/dissertation

Time-Course Analysis and Clustering of Gene Expression Data

by

Stephanie DeGraaf

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Statistics

and the Designated Emphasis

in

Computational and Genomic Biology

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Associate Professor Elizabeth Purdom, Chair
Professor Sandrine Dudoit
Professor Haiyan Huang

Spring 2020

Time-Course Analysis and Clustering of Gene Expression Data

Abstract

Time-Course Analysis and Clustering of Gene Expression Data

by

Stephanie DeGraaf

Doctor of Philosophy in Statistics

and the Designated Emphasis in

Computational and Genomic Biology

University of California, Berkeley

Associate Professor Elizabeth Purdom, Chair

High-throughput time-course studies collect measurements from samples across time. In particular, longer-duration high-throughput time-course studies are becoming more common, such as in the case of 16S sequencing of bacterial communities or single-cell mRNA sequencing of developmental lineages. A common focus of these studies is on significance testing per gene. However, in many settings, particularly those studying developmental processes, large numbers of genes show temporal changes, and the relevant question is instead to classify genes into different types of temporal changes. We propose a mixture-model clustering method that estimates a functional spline model for the mean of the cluster, in order to cluster the temporal patterns of genes independent of scale. The model allows for a wide range of likelihood models to suit a variety of data types. In addition, this clustering strategy accounts for time-course data under different experimental conditions or developmental lineages, and it provides a method for evaluating, per cluster, significant differences in temporal patterns between conditions. This allows for an integrated analysis of differential expression analysis and clustering. We demonstrate the benefits of our method using simulated data. In addition, we explore several real data sets to illustrate both the context for and the application of the mixture model.

To my parents,

who have supported me every day of my life,

and to my siblings,

Nathan, Melissa, Benjamin, and Daniel,

who have kept me laughing the whole way.

Contents

List of Figures

Acknowledgments

I owe a great deal to many people who have supported, encouraged, and inspired me throughout my graduate studies. I would like to extend my sincere gratitude to a few of those people who have made my time at Berkeley both rich and meaningful.

To Elizabeth Purdom, my advisor. I have learned much from Elizabeth's gentle teaching and by observing her natural curiosity and dedication to statistical rigor. More than this, she has consistently demonstrated kindness and grace, and has extended a real warmth of compassion to me. I would not have continued my graduate studies more than a year if it were not for her patient guidance. She is truly a gem in the statistics department.

To Nelle. Her dedicated work as a postdoc and beyond has been admirable. In our work together, she has generously provided me with a wealth of guidance in programming skills and collaboration tips and has patiently corrected countless bugs in my code.

To Sandrine and Haiyan. They have kindly and cheerfully supported my progress in the statistics department, provided valuable feedback on this work, and have always made the biostat lab an engaging environment. Thanks also to the students in the biostat lab for their helpful feedback and encouragement.

To Devin and Ling, the best biology collaborators. They first introduced me to the wonderful world of microbiome data, and their enthusiasm for plant microbial biology is fantastically contagious.

To my parents. My mother was my first teacher, who taught me to delight in learning as a child and nurtured me in all aspects of life. My father has always filled my life with laughter, wisdom, and a love for math, and was first to set me down the path to study statistics. I am deeply grateful for how they have paved my way in life, shepherded me, and supported me at every step along the way.

To my siblings: Nathan, Melissa, Benjamin, and Daniel. Each have cared for me in uniquely personal ways at every step of my studies. Thank you for being the best fan club, for sharing your homes and dinners and stories and fun facts, for filling my life with joy, and for never letting me take myself too seriously. I love having you as my sister and brothers.

To my dear family at Christ Church Berkeley. It is one of the greatest blessings of my life to be part of this flock. My especially heartfelt thanks goes to Jonathan, Andrew, Megan W., Sam V., and Meredith.

Finally, to Him who called me out of darkness and into His marvelous light, my Lord and Savior Jesus Christ. To Him be all glory and honor and praise both now and for ever.

Chapter 1

Introduction

Genomics experiments are a useful tool to investigate the biological behavior of organisms. With such experiments, we can study the a wide range of phenomenons, such as the effects of drought on plants, the response of mice to a stimulus, or the progression of disease in humans. While many types of observable data may serve to aid our understanding, genomics data in particular have the potential to illuminate the underlying genomic processes that are associated with external observations. For instance, we might discover which genes are crucial at specific times in a developmental process, or identify microbes that are essential to a plant's ability to survive under drought, or predict the success of a drug based on a patient's genetic profile. Such discoveries made by analyzing genomic processes could then enable us to treat and respond to biological problems such as drought, injury, or disease in a more effective manner.

Alongside this tantalizing potential lurk a variety challenges in analyzing genomics data that must be overcome in order to identify the true biological behavior that the data have captured. Failing to address these issues can lead to spurious conclusions and false detection of discoveries, leading researchers astray and leaving us blind to the true story in the data. Therefore, rigorous statistical methods must be employed in genomics analysis to facilitate accurate conclusions and appropriate responses.

1.1 Time-course gene expression context

One particular context that presents unique challenges is research studying changes across time, such as developmental patterns or changes in response to stimuli. Genomics data is a valuable tool that can identify temporal patterns of interest as well as genomic features that are particularly relevant from a biological standpoint.

The focus of this dissertation to analyze time-course gene expression data. Time course gene expression data involves two elements that make this an interesting area of study: the time-course nature of the data, and the high-dimensional nature of gene expression data. Time can be considered in a chronological context, as in an experiment where samples have

been collected across multiple time points, or in a developmental context, as in a single-cell experiment. Gene expression data can be of any type of omics data; to name a few, we can consider RNA-seq, ChIP-seq, 16S rRNA-seq, scRNA-seq, etc.

Time-course gene expression is an area of rich scientific interest that has long been explored in a variety of applications. Briefly, we note a few examples of time-course gene expression studies to demonstrate their wide breadth of applications. To study the regulation of leukocyte activities in the human immune system, Calvano et al. [2005] used microarrays to study gene expression in whole blood leukocytes at multiple time points before and after administration of bacterial endotoxin to subjects. Rodwell et al. [2004] studied age-related kidney degeneration by examining gene expression in the kidney cortex of human subjects ranging in age from 27 to 92 years. In Shoemaker et al. [2015], the authors modeled the gene expression dynamics of the mouse immune response to multiple influenza virus isolates across 14 time points following infection. In the context of plants, Varoquaux et al. [2019] conducted a field experiment of the sorghum crop, collecting transcriptomics data from plants under multiple watering conditions across the time from seedling to maturity. Each of these studies involve at least one type of high-dimensional genomics data collected across a time-course of long duration.

If we take a less conventional definition of time, we can consider time-course studies in the context of single cell data. The technological development of single cell assays has enabled researchers to isolate the transcriptomic profile of individual cells, rather than profiles of bulk collections of cells as in RNA-Seq or microarrays. Single cell data therefore provide the ability to distinguish between various cellular states, as well as to infer an ordered progression through those cell states. This kind of inference assumes that cells develop along an underlying time variable, typically called *pseudotime*. The challenge of inferring this developmental ordering is an interesting area of study, and several methods have been proposed to reconstruct the pseudotime variable, such as Monocle in Trapnell et al. [2014], or Slingshot in Street et al. [2018]. Typically the construction of the pseudotime variable is done by ordering cells along a path through a reduced-dimension space. Once a developmental ordering has been inferred by an appropriate method, the transcriptional changes of genes can be analyzed along this pseudotemporal progression of cells. One such study that analyzes scRNA-seq data in this way is Gadye et al. [2017], which investigated the transcriptional processes of stem cell renewal and differentiation in the regenerating mouse olfactory epithelium following injury. The authors in particular noted a handful of key genes that showed dynamic patterns of gene expression along the inferred developmental trajectory. As single-cell experiments continue to grow in popularity, it will be highly valuable to have methods that can analyze patterns of expression along developmental trajectories for large numbers of genes.

These examples serve to illustrate the broad range of interest in temporal and developmental changes and the corresponding need for statistical methods that are widely applicable to a variety of data types. The focus of this dissertation will be to provide a method to comprehensively analyze the temporal patterns observed in the data as well as identify the most significant patterns and genomic features.

1.2 Challenges with long time-course data

While time-course gene expression has long been an exciting area of study, many of the statistical methods have been developed for use in microarray experiments. As high-throughput sequencing data becomes more available, more time-course studies are taking advantage of the rich information provided by sequencing data in studying changes of gene expression across time. However, the statistical methods needed to analyze the data coming from these experiments of longer duration are underdeveloped. This is problematic because time-course designs of longer duration, while filled with rich potential for discoveries, also come with a number of statistical challenges that do not arise in the context of shorter time-course studies.

Common statistical approaches to analyzing time-course genomics experiments have relied on factorial models using packages like `edgeR` Robinson et al. [2010] or `limma` Smyth [2005]. These methods are designed to treat a variable such as time as a factor variable, creating a design where each time point has its own binary indicator variable. This type of factorial model is appropriate and performs well for studies without a time-course design, or those with a time-course of short duration, with only a few time points. However, with longer time course studies, the simple factorial model becomes less appropriate, since the factorial design essentially ignores the gradual progression of biological behavior across time. In addition, the factorial model quickly grows in complexity with increasing numbers of time points, especially if the model includes interaction terms. Therefore, a more advanced model of the time variable is needed for these longer time-course studies.

Statistical modeling of time course data is a rich area of study and provides many options for modeling time variables that can be applied to the genomics context. Indeed, some of these time course methods have begun to be applied to genomics data. Early work in this area was focused on microarray data, and a thorough review of many methods to identify patterns, clusters, and gene-regulatory networks in time-course settings is given in Bar-Joseph [2004]. Most of these methods are either exploratory in nature or rely on ANOVA procedures to conduct any type of inference. A variety of model-based methods have also been proposed, such as a polynomial model including full interaction terms for time and experimental conditions in Conesa et al. [2006], a Bayesian auto-regressive hidden Markov model in Leng et al. [2015], an analysis of the derivative Fourier coefficients in Kim [2011], and wavelet-based functional clustering in Kim et al. [2010]. Again, most of these methods are more appropriate for time courses of shorter duration and are not designed to handle longer time course studies. Among the more relevant methods for time courses of a longer duration in particular is a functional spline model described in Leek et al. [2006], which uses Gaussian smoothing splines to conduct differential expression testing for microarray data.

While many of these methods provide interesting avenues of exploration, in our work we explore a functional data analysis approach, as our focus is on longer time-course studies. A functional data analysis strategy seems most promising for studies with time-courses of long duration due to its flexible nature, which gives it the ability to model changes across many time points Furthermore, modeling a time variable using a functional model allows

us to perform our desired analyses of clustering and differential expression analysis with simplicity and clarity.

1.3 Challenges with high-dimensional genomics data

To compound the problem of modeling time variables of long duration, genomics data involve a number of challenges that require careful consideration. Among the most important is the high-dimensional nature of genomics data, in which a typical genomics study collects data for tens of thousands of genes. This can lead to an overwhelming set of results in which it is difficult to discern the most important conclusions from among all genes. An additional challenge with time course genomics studies is dealing with the characteristics of genomics data analysis that have been so carefully addressed by the `edgeR`, `limma`, or `zinbwave` models. In particular, two areas of consequence in analysis are selecting the appropriate data distribution and applying a reasonable normalization technique.

The major challenge of analyzing genomics data in time-course studies is to make sense of the vast numbers of changes observed in the data collected across time. In a genomics experiment involving tens of thousands of genes, it is typical that extremely large numbers of these genes will show differences across time. This is to be expected, since measuring gene expression across time captures the developmental processes of genes, and each gene will likely show activity at some point along a developmental process. Therefore, traditional methods of genomics analysis like differential expression analyses are not particularly illuminating in this context, because these methods will identify nearly all genes as being differentially expressed across time. While these conclusions are correct, they are also not the most useful for further biological interpretation and study. In order to discern meaningful conclusions from an excess of genes showing differential expression, the goal of our analysis changes from identifying the particular *genes* that demonstrate interesting behavior to identifying the typical *patterns* of behavior and then classifying genes into those groups of similar expression across time. To this end, the method that we present will focus on this broader question of pattern detection via clustering, as well as conducting differential expression analysis at the cluster level rather than the gene level.

Besides this overarching challenge of high dimensionality, a key issue to address in any model-based genomics analysis is to choose the data distribution that is most appropriate for the data type. Gaussian models such as in the `limma` package from Smyth [2005] work well for microarray data, since microarrays report a continuous measure of fluorescence intensity that has been shown to be normally distributed. Sequencing data like RNA-seq data, however, report integer count values that measure the discrete number of sequenced reads aligning to a reference transcript. Hence, a count distribution is most appropriate to model sequencing data. The negative binomial model is used effectively in the `edgeR` package from Robinson et al. [2010] and the `DESeq` package from Love et al. [2014]; other models rely on the Poisson distribution as in Witten et al. [2011] and Rau et al. [2015]. Furthermore, specific types of sequencing data, such as 16S amplicon data, or single cell RNA-seq data, require an

additional consideration for the high numbers of zero counts, due to technical dropouts or rare presence of certain genes. The negative binomial model alone has been shown to be insufficient to properly model the high frequency of zero counts arising in single-cell data, and a zero-inflated negative binomial model was proposed to more effectively model zero-inflated count data in Risso et al. [2018]; these concepts naturally extend to the context of 16S microbial data. These are just a few examples of selecting the appropriate distribution for the data type, and other types of genomics data require equally careful consideration of the appropriate distribution.

Furthermore, modeling genomics data requires a normalization step to account for global sample-level differences. These are differences in the magnitude of counts or frequencies between samples that arise from causes unrelated to a true biological signal. There are many factors that can create unwanted global variation across samples; without normalizing, true biological differences between samples are obscured or entirely masked. Typical sources of sample-level variation can come from technical variation, such as library preparation in sequencing experiments or probe effects in microarrays, as well as batch effects caused by differences in the environment, day, or scientist during sample processing. Removing this unwanted global variation consists of applying a normalizing correction to the data that seeks to put all samples on a comparable scale. Typical ways to do this are normalizing each sample by a function of library size or by the expression of control genes. The advantages of various normalization techniques have been discussed in Bullard et al. [2010] for mRNA-seq, in Lytal et al. [2020] for single-cell RNA-seq, and in Weiss et al. [2017] for microbial amplicon 16S data.

Failing to choose an appropriate distribution or normalization technique can lead to serious errors in data analysis, such as a high number of false positives and false negatives, as explored in Seyednasrollah et al. [2015] and Risso et al. [2018]. To avoid these errors, the method that we have developed will incorporate the normalization and data-distribution techniques used in the `edgeR` and `zinbwave` packages, both of which have a well-developed framework to address these characteristics of genomics data.

1.4 Goals for analyzing time-course gene expression data

Before presenting our method to analyze time-course gene expression data, it is important to make explicit the purposes that such a model will serve. From a statistical perspective, the main purpose of our method is to address the unique challenges that arise from a time course of long duration and the high-dimensional nature of genomics data. More broadly, these statistical concerns require first understanding the relevant biological questions of interest as they relate to differential expression testing and clustering.

A general goal of genomics studies in contexts involving long time courses is to identify typical patterns exhibited in the data, as well as to cluster features into groups represented by

those patterns. These patterns may then be used to investigate various biological hypotheses about developmental patterns of various genes. Furthermore, by clustering genes into shared patterns, we can explore which genes are behaving in correlated and anti-correlated manners. Our method relies on a mixture model clustering algorithm that will perform both functions of identifying patterns and assigning features to groups simultaneously. In doing so, the mixture model probabilistically fits a cluster mean to provide an accurate representation of the cluster patterns, as well as a probabilistic assignment of each feature to each cluster. The method relies on a functional model of temporal expression, making it particularly appropriate for our context of time-course studies of long duration.

In addition, an overarching goal of many genomics studies is selecting the most interesting features that perform a significant biological function. Typically, this is done by conducting a differential expression analysis at the gene level, using statistical tests like t-tests and F-tests for each gene to detect the most significant genes. As previously noted, this approach to differential expression analysis in the context of longer developmental time course studies is challenging, due to the high numbers of genes detected to be significant. Therefore, our method provides a way to test for differential expression at a cluster level, identifying the most interesting *groups* of genes, rather than attempting to select individually significant genes. Moreover, within a relevant cluster, we also provide the ability to identify the genes that most strongly match the cluster pattern, which are likely to be the most relevant genes for further investigation of a particular cluster pattern.

The remainder of this dissertation is organized as follows. In Chapter 2, we will explore a specific real data example of time-course genomics data analysis where our method could be applied. We then present the mixture model method, focusing on the clustering analysis in Chapter 3 and the differential expression analysis in Chapter 4. A concluding discussion is given in Chapter 5.

Chapter 2

Statistical analysis of genomic datasets

2.1 Introduction

To illustrate the context for the methods we have developed, we will explore a genomics dataset collected across a time course of long duration. This dataset comes from the EPICON project, which studies the epigenetic control of the drought response in the sorghum plant. Here, we briefly give a background to the EPICON project to provide context for the analysis that follows.

Sorghum is a cereal crop native to Africa and is cultivated in many areas of the world today with increasing popularity. Over the past 50 years, the area planted to sorghum worldwide has increased by 66 percent, and sorghum yield has increased by 244 percent Stroade [2020]. The grain is commonly used for food, liquors, animal feed, and ethanol, and is of scientific interest for many reasons. Among the most important reasons, the sorghum plant is very drought-tolerant. With drought causing annual losses of $2.9 billion, a more thorough understanding of the biological mechanisms of drought tolerance is both highly relevant and valuable (Pachauri et al. [2014]). In addition, sorghum is resistant to heat stress, soil toxicity, and pests and pathogens; it is also a highly efficient crop in terms of water and solar energy usage. Finally, sorghum is highly nutritious as food, making it an ideal crop to help meet the increasing global food demand.

In light of these appealing characteristics of sorghum, the EPICON project was designed to study the biological mechanisms of the sorghum plant that aid in its tolerance of drought. Of particular interest are the transcriptomic, proteomic, metabolomic, and microbial responses to drought. To study these processes, EPICON conducted a field experiment of sorghum grown under various watering conditions. Sorghum plants were grown at the University of California Kearney Agricultural Research and Extension Center, and plant samples were collected on a weekly basis throughout the life of the plant. Two drought conditions were imposed: a pre-flowering drought condition, where the drought took place before the

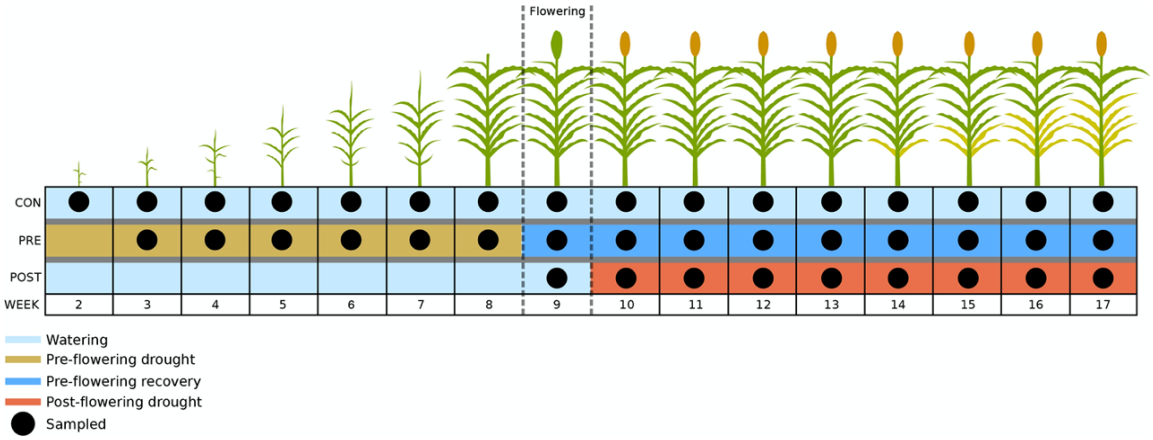


Figure 2.1: Sampling plan for the EPICON project.

plant flowered, and a post-flowering drought condition, where the drought took place after the plant flowered. In addition, two genotypes of sorghum were used that were hypothesized to perform particularly well under each drought condition: the pre-flowering drought-tolerant RTx430 and the post-flowering drought-tolerant BTx642. The experimental design is shown in Figure 2.1, where at each week, three replicate samples of each of the two genotypes of the sorghum plant were collected from each watering condition, and the flowering week is marked at Week 9.

Many types of genomic datasets were collected and analyzed from each weekly sample, including mRNA-Seq transcriptomics, small-Seq, 16S rRNA-Seq, ChIP-Seq histone data, BS-Seq methylation data, proteomics, metabolomics, metagenomics and metatranscriptomics. In general, the various datasets demonstrated rich and dynamic changes across time and between conditions, as detailed in Varoquaux et al. [2019], Gao et al. [2019], and Xu et al. [2018a]. To illustrate just one of these analyses, we will examine the 16S microbial dataset.

2.2 Development of the sorghum root microbiome under drought

The plant microbiome is of interest because root microbes are known to provide multiple benefits to the host plant. The root microbiome of a plant is composed of fungi, bacteria, and archaea that colonize in the roots of plants and perform a variety of functions. While some microbes are purely parasitic, feeding off the host plant's carbon source while providing no benefit tot he host, symbionts can increase the host plant's access to nutrients, regulate plant growth, boost immunity, and protect against environmental stress. Importantly, the root microbiome can provide protection against drought stress. However, despite these great benefits, little is known about the dynamics of the development of the plant microbiome,

under either regular watering conditions or drought conditions.

To investigate the role of the microbiome in sorghum's drought tolerance, the EPICON project collected weekly samples from the soil, rhizosphere, and roots of the sorghum plants and applied 16S rRNA amplicon sequencing to each sample. 16S amplicon sequencing is a protocol pioneered by Carl Woese that is widely used to study the identities and relationships of microbes present in a microbial sample. Briefly, the process for a given sample consists of extraction and PCR amplification of the 16S genes, high-throughput sequencing of the 16S gene, and clustering the resulting sequence reads by similarity into Operational Taxonomic Units (OTUs), which are thought to roughly approximate bacteria species. After OTU clustering, the number of sequences assigned to each OTU can be counted in each sample, creating a data matrix of the read counts of each OTU in each sample.

This protocol for sequencing the 16S gene only provides useful information about a microbial community because of the strikingly convenient characteristics of the 16S gene. This gene consists of both highly conserved and highly variable regions, providing rich information about what kinds of bacteria are present in a microbial sample as well as their taxonomic relationships. Specifically, the variable regions in the 16S gene allow for a phylogenetic structure to be established for each of the bacteria present in a microbial sample, so that each bacteria is assigned a genus, family, order, class, and phylum in the taxonomic hierarchy according to its genetic sequence. While 16S hypervariable regions are not able to differentiate between bacteria at the most specific rank of the species level, the broader classifications at higher levels still provide a wealth of useful information. This taxonomic information can be used to discover which types of bacteria are most and least abundant in a microbial community, and how their presence changes in response to time or experimental conditions.

2.2.1 Background analysis of microbiome dataset

The full 16S microbial dataset was analyzed in Xu et al. [2018a], where the authors found many varied and dynamic responses to drought. We will note a few observations to provide background context for the work that follows.

Bacterial community diversity is a common area of interest in microbiome studies. Shannon's diversity is a statistic that is used to measure the amount of diversity within a microbial sample, giving us an idea of the degree of variety of different kinds of microbes present in a given sample. Measuring Shannon's diversity for each time point's samples in the EPICON data revealed a period of rapid increase in community diversity following seedling emergence, followed by a rapid decrease after this initial colonization period. Samples from the soil demonstrated unchanged diversity throughout the development of the plant and under all conditions. In contrast, samples from the rhizosphere and roots showed significant decreases in diversity during both drought conditions. Bray-Curtis dissimilarity is another diversity metric that measures the diversity between microbial samples, providing a measure of how different the microbial communities are between two samples. Comparing the microbial composition of the soil, rhizosphere, and roots using Bray-Curtis dissimilarities indicated strong differences in composition across all three sample types. Furthermore, bacterial communities

showed high diversity across treatments and time points, although there was no observed difference in composition according to the genotype of the plant. These observations lead to the conclusion that microbial communities differ across the soil, rhizosphere, and root locations, and furthermore, that they undergo significant changes in response to time and watering condition.

A further investigation of the microbiome development across time revealed that drought has a strong delaying effect on the progression of the root and rhizosphere microbiome. Similar patterns of development were observed under both the pre-flowering drought condition and the control watering condition, but with a brief two to three week delay in the drought progression. Other than the delay in timing, however, the development of the microbiome under drought generally follows a similar progression as under control. An additional significant observation was made that during the period of delay in the pre-flowering drought condition, the root microbiome demonstrated a strong enrichment of bacteria primarily from the Actinobacteria phylum. These observations suggest that the microbial communities change in complex manners across time and between conditions, and in our analysis, we further explored these dynamics.

2.2.2 Statistical analysis of dynamic lineages

To better understand the delayed progression that drought induces in the root microbiome, we further investigated the temporal dynamics of the bacterial lineages that were most dynamic across time and condition. To identify these OTUs of interest, we tested each OTU for a significant difference in abundance across time between conditions by fitting a smoothing spline model to each OTU.

Before fitting the model, we first conducted a normalization step to account for confounding differences in the magnitude of counts between samples. As is a standard practice with 16S data, we simply normalized the data by relative abundance, dividing each sample by its library size. In addition, we took the log of the counts of the OTUs so that a Gaussian distribution would be an appropriate choice for the data distribution of the model.

We have many options when choosing a model for the abundance of each gene. The simplest model choice would be to treat time as a factor variable and conduct group comparisons across time conditions; however, this ignores the temporal dependence between time points that is at the heart of a time course study. Another option is to fit a simple linear regression model to the data. The problem here is that doing so makes assumptions about the data that in many cases can be unfounded. In particular, linear regression fundamentally assumes that changes in responses are proportional to changes in the predictor, i.e., that the change in abundance is the proportional to the change across time. However, this would be a faulty assumption in our context because it is evident that OTU abundances change at different rates across different prediction intervals. Splines are a promising method in functional data analysis that can address these issues and provide unique advantages over alternative modeling choices.

Functional models provide a way to model OTU abundances in a flexible manner, by considering the data as functional observations of an underlying trend. In this way, splines provide the ability to model the data in a different way for different intervals of the predictor. A spline is a piecewise function whose smoothness can be controlled by tuning a smoothness parameter or by specifying a series of breakpoints where the function changes. It is crucial to moderate how aggressively the splines fit the data, and a challenge with spline methods is to find a function that models the underlying trend without modeling noise in the system. Spline regression involves choosing a particular function to fit segments of our data. Functions are created from a chosen set of basis functions (such as Fourier, wavelet, polynomial, B-splines, etc.), and intervals are determined by a knot sequence, usually chosen so that more knots are placed in areas of the data that require a closer fit and fewer knots in smooth areas of the data. In terms of this representation, the balance between smoothness and model fit is controlled by choosing the number and location of knots used to construct the spline.

2.2.3 Spline model and results

Smoothing splines provide an advantageous method to model the dynamics of OTU abundance across time, and here we define the particular model applied to the microbial data set. The log relative abundance y_{ij} of OTU i in condition j at time t is modeled as

$$y_{ij} = \mu_{ij}(t_j) + \epsilon_{ij}(t_j),$$

where μ_{ij} is the population average time curve modeled by a p-dimensional basis of B-splines,

$$\mu_i(t) = \beta_{i0} + \beta_{i1}\Psi_1(t) + \cdots + \beta_{ip}\Psi_p(t),$$

and ϵ_{ij} are assumed to be independent errors from a Normal distribution. Here, we use a B-spline basis defined by three knots at equally spaced quantiles along the time from weeks one through 17.

This model was fit on a per-OTU basis. After the model was fit, significant OTUs were detected by testing the significance of the interaction of time point and treatment. This was done by using the likelihood ratio test comparing a model with an time-treatment interaction term to a model without the interaction term. The likelihood ratio test produces a p-value for each tested gene, and to control for multiple testing errors, the Benjamini-Hochberg multiple testing correction was applied to the resulting p-values from the gene-wise likelihood ratio tests. In total, this differential expression analysis identified 576 OTUs with significant differences in temporal behavior between conditions.

Next, to better understand the behavior of these 576 significant OTUs, we sought to identify their various patterns of temporal behavior between conditions. To do so, we performed hierarchical clustering of the fitted splines of the 576 significant OTUs. Results are shown in Figure 2.2, where for each of six clusters, the cluster mean under both the control condition and pre-flowering drought condition is plotted across time.

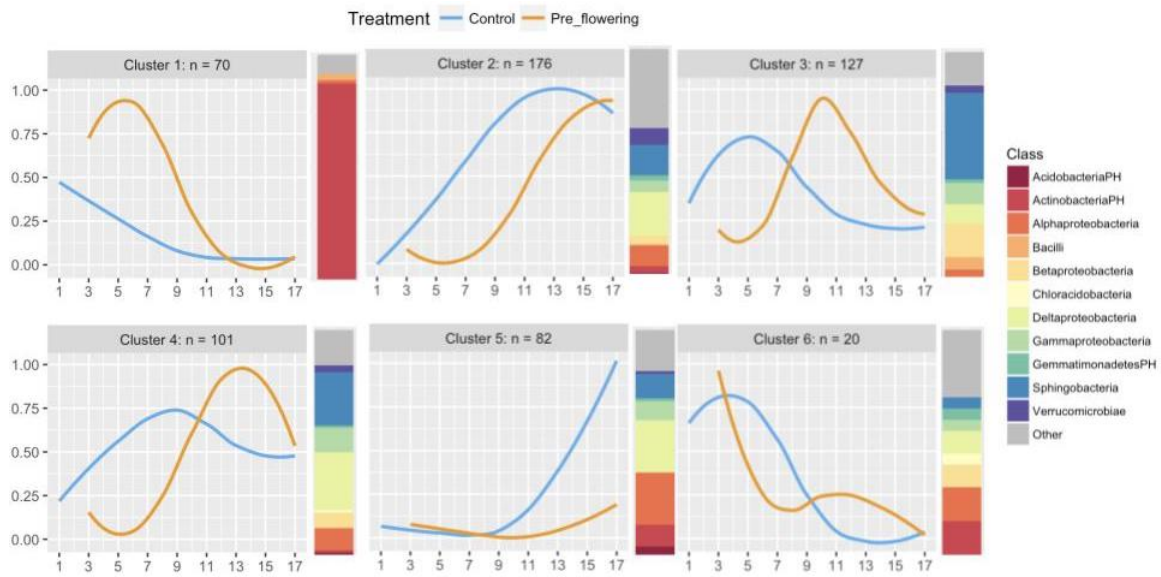


Figure 2.2: Cluster mean temporal expression and distribution of class membership of OTUs in each of the six clusters identified by hierarchical clustering.

From these cluster patterns, we confirm the observation made earlier of the delaying effect of drought on the progression of the microbiome, as the patterns that are identified in clusters two through six exhibit a similar progression across time under both control and drought conditions with a slight delay under the drought condition. Cluster One, on the other hand, shows a unique pattern of strong enrichment under drought which does not follow the typical delayed behavior demonstrated by the other clusters of OTUs.

For further exploration of Cluster One in particular, we investigated the distribution of taxonomic class membership of the OTUs belonging to each cluster. Each OTU was identified by its class assignment as either belonging to one of the top 11 most common classes or an "Other" category with all of the remaining classes. By a visual assessment of these distributions, we observe that most clusters are composed of a wide variety of different classes, indicating that the behavior of most OTUs is unrelated their taxonomic class assignment. However, Cluster One again stands out for being made up of almost entirely Actinobacteria OTUs; furthermore, there are relatively few Actinobacteria OTUs represented in other clusters. This again points to unique enrichment behavior by OTUs in the Actinobacteria class, suggesting that Actinobacteria OTUs are performing some unique function during a drought period.

2.2.4 Conclusions of microbiome analysis

The pattern of Actinobacteria enrichment shown in Figure 2.2 was noted in Xu et al. [2018a] and further explored experimentally. The authors concluded that the defining characteristic

of enrichment under drought by these OTUs was the structure of the bacteria's cell wall. Bacteria can be classified in terms of their cell wall structure as either monoderms, which possess one cell membrane and a thick peptidoglycan cell wall, or diderms, which have two cell membranes and a thin peptidoglycan layer. Interestingly, enriched bacteria under the drought period were all classified as monoderms, while depleted bacteria were classified as diderms. The few exceptions to this rule possessed cell wall structures with unique characteristics, which were explored further in Xu et al. [2018a].

The observed enrichment of monoderms raises questions about the underlying cause of this enrichment. A comparison with the metabolomics data from the EPICON project proposes a possible explanation for the cause of this enrichment. Based on a differential expression analysis of root metabolites, the most significantly enriched metabolite was found to be glycerol-3-phosphate, which was 4.34 log10-fold more abundant in drought-treated than control roots, and this enrichment was further shown to be host-generated. In addition, G3P is known to be a precursor to peptidoglycan biosynthesis used by monoderm bacteria in the development of their cell wall. Putting these observations together suggests that the abundance of G3P may be positively selecting for monoderms, leading to their enrichment under drought.

From a statistical perspective, we can conclude that our smoothing spline model produced both accurate and meaningful results, as the model and clustering supported the conclusions made by Xu et al. [2018a]. In addition, the spline model provides a helpful visualization of the various temporal patterns exhibited by the various OTUs in the root microbiome.

Chapter 3

A method to cluster time-course gene expression data

This chapter focuses on work done jointly with Nelle Varoquaux.

3.1 Introduction

High-throughput time-course studies collect measurements from samples collected across time via high-throughput experiments like mRNA-Seq or microarrays. Early microarray technologies used time-course designs along with corresponding methods for their analysis. Despite their long history, however, there has been relatively less attention on methods for time-course expression studies as compared to the more frequently studied design of group comparisons. Much of the analysis of time-course expression data comes from early work in gene expression microarray studies and focused on using functional data analysis techniques to estimate gene expression as a function of time and the resulting problems in estimating these parameters simultaneously across thousands of genes. However, many time-course expression studies are of relatively short duration (few time points) where a simple factorial model comparing time points is appropriate, via packages like `edgeR` or `limma` (Robinson et al. [2010], Smyth [2005]), rather than functional analysis techniques. Longer high-throughput time-course studies are also commonly used in mRNA studies and are becoming more widely used in the cases of 16S sequencing of bacterial communities, as in Xu et al. [2018b], or single-cell mRNA sequencing (scSeq) of developmental lineages, as in Gadye et al. [2017]. Due to these types of settings, there has been increased attention recently on time-course studies and the adaptation of functional analysis techniques in the presence of low sequencing counts and zero-inflation.

Much of this work has also been focused on the question of significance testing per gene – either testing whether there is any change over time or comparing between groups over time. There has been less attention to the common question of clustering of features from time course experiments. This is particularly unfortunate, since many longer time-course

studies result from developmental gene expression studies, where large numbers of genes show temporal changes. In this setting, the relevant question is less the detection of any genes with temporal effects but rather the classification of genes into different types of temporal changes. Some techniques attempt this classification via hypothesis tests per gene (Sander et al. [2017], Van den Berge et al. [2019]), but clustering is a more natural strategy that does not require a priori definition of the types of temporal changes of interest.

In this chapter, we propose a mixture-model clustering method that estimates a functional spline model for the mean of the cluster, in order to cluster the temporal patterns of features. The mixture-model conducts this clustering of each feature independent of scale, enabling the identification of genes with inverted patterns of expression. Furthermore, the mixture model allows for a wide range of likelihood models, including the normal model, the negative binomial model applicable for RNA-seq data, and the zero-inflated negative binomial model relevant for both 16S sequencing and scRNA-Seq.

We also show how the results of our clustering allow for integration into important downstream analyses. The resulting centroids from the mixture model provide an accurate representation of each cluster, allowing for a compact, visual understanding of the clustering. In addition, within each cluster, our scale parameter provides the unique opportunity to identify genes showing similar, but opposite patterns. This is useful from a biology perspective as a way to find genes that behave in exactly inverted ways. Finally, the mixture model provides probabilities that each gene belongs to its assigned cluster, measuring how well each gene fits that spline centroid. These probabilities are useful both to filter out genes with poor fits and to enhance downstream gene set enrichment analysis.

3.2 Existing clustering strategies

Our goal is essentially to cluster genes based on their temporal profiles. There are many available methods that can be used to cluster data, though none are particularly well-designed for our context of a developmental gene expression study involving a time course of long duration. Standard clustering procedures, while frequently used for time-course data, do not take advantage of the time-dependent nature of the data. For example, K-means and hierarchical clustering do not incorporate any available model information such as time or condition. Model-based approaches are more relevant for this time-course context, although existing strategies are again insufficient to handle our desired applications. In this section, we will explore the advantages and disadvantages of each existing method, and later compare the performance of our mixture model method to the following alternative cluster methods.

K-means is a highly popular clustering algorithm that clusters data by assigning each observation to the cluster with the nearest mean in terms of Euclidean distance. To do so, the algorithm minimizes within-cluster variances (squared Euclidean distances). K-means is a very simple method, and it is by far the quickest method we explored. However, while K-means is very quick and easy, it does have significant disadvantages. First, while K-means does provide cluster centroids as representations of each cluster's pattern, these centroids

are computed by simply taking the average of the all data points that belong to each cluster. This may not be the most accurate summary of a cluster's pattern. Secondly, K-means is a hard-clustering method, meaning that each gene is simply assigned a cluster label, with no additional information about how well it fits that cluster relative to the other clusters. As a third point of consideration, K-means relies on Euclidean distance, suggesting that it might have poor performance on data that do not follow a Normal distribution. Finally, K-means does not incorporate any available model information, such as time point or condition into its algorithm.

To address these concerns and to apply K-means to our time-course gene expression data, we can adapt K-means for our purposes in a variety of ways. The first method is the simplest, where we just apply K-means to the original data with no modifications. Next, because K-means relies on Euclidean distance as its metric, we apply K-means after first centering and scaling the data per gene. Finally, one of the drawbacks of using K-means is that because it is not a model-based algorithm, we cannot incorporate any of our available model information, such as which time point or condition are associated with each of our samples. In order to improve the performance of K-means, our third approach was to apply K-means to the values obtained from fitting a spline model per gene under the Normal distribution. The idea is that the fitted values provide a way to incorporate the model information in order to help K-means cluster more accurately. We fit this K-means spline model per gene using the `edge` package described in Leek et al. [2006], either on the original data for Normally distributed data, or the log counts for count data. As a further modification to this third option to specifically address zero-inflated data, we applied K-means to the values obtained from fitting a spline model per gene under the ZINB distribution. We fit this model using the `zinbwave` package described in Risso et al. [2018].

Hierarchical clustering is another popular clustering approach. This method clusters observations by constructing a hierarchy of similarity between observations, either in an agglomerative or divisive approach. In either approach, grouping the sets of observations is based on a measure of dissimilarity between observations. This dissimilarity measure can be any metric function; common measures used are Euclidean distance, Manhattan distance, and Cosine distance. To adapt this clustering technique for RNA-seq data, Witten et al. [2011] proposed a method to apply hierarchical clustering to count data by relying on the Poisson likelihood to construct a measure of dissimilarity. We further extended this approach by adapting this dissimilarity measure for zero-inflated data by using the ZINB likelihood in the dissimilarity measure. In comparison to K-means, hierarchical clustering suffers from many of the same disadvantages. Again, it cannot take into account the model information of time or condition. It is also a hard-clustering method, providing no information about how well an observation fits a given cluster. As it relies on a distance metric, careful consideration must be taken in choosing the appropriate distance metric for a data type. Finally, hierarchical clustering also does not provide any kind of cluster summary beyond the cluster labels, such as cluster centroids.

Turning to model-based approaches, the mixture model framework has been used successfully in a number of applications. In the context of count data, specifically RNA-seq

data, Si et al. [2014] adapts the typical Gaussian mixture model framework to rely on a negative binomial likelihood, providing a method suitable for clustering count data across various experimental conditions. Similarly, Rau et al. [2015] described a finite mixture model of Poisson log-linear models for the clustering of count-based gene expression profiles. Each of these methods account for the nature of count data but are designed for experimental settings of group comparisons and do not directly address the setting of time-course studies. A model-based clustering method developed specifically for time-course data is described in Ma et al. [2006], which uses a mixed-effect smoothing spline model in a Gaussian mixture model. However, while useful for Gaussian data, this mixture model does not directly apply to count settings. These model-based approaches show more promise than K-means and hierarchical clustering because they provide a framework to include model information, yet they are each lacking in their ability to apply to our time-course setting for a variety of genomics data types.

Furthermore, the mixture model methods currently available are limited in the amount of variation that is allowed among genes in a given cluster. The methods of Ma et al. [2006], Si et al. [2014], and Rau et al. [2015] each include in their models a gene intercept parameter to account for the overall mean expression level of a gene relative to the cluster mean. Essentially, this estimates a vertical shift for each gene relative to the cluster mean. This is a useful parametrization but limited in the sense that it assumes that each gene demonstrates changes in expression on the same scale. That is, none of the models account for gene-specific differences that vertically stretch or shrink the overall cluster mean. Ignoring these gene-specific differences in scale may affect both the clustering and cluster mean estimation, and one of the goals of our method is to incorporate a scale parameter to account for this gene-wise variability.

While each of these clustering methods are popular and reasonable approaches to our clustering problem, none of these methods are particularly well-suited for the context of time-course gene expression or are limited in their functionality. Therefore, we would like to propose a model-based framework that serves to address the needs and difficulties of clustering time-course gene expression data.

3.3 Mixture model method

For simplicity in what follows, we will assume that the features from the high-throughput experiment that we are clustering are genes, but the features could also be other features, such as operational taxonomic units (OTUs) from 16S sequencing. Let y_{ig} be the observed data for gene g and sample i at time t_i. We can also optionally allow the data to be observed across time under different conditions, which we notate as C_i. We assume that y_{ig} follows a known distribution with density f. Then we use a mixture model to cluster the data, such that

$$P(y_{ig}) = \sum_{k=1}^{K} \pi_k f(y_{ig}|\mu_{ig}^k, s_i, \theta_g)\,,$$

where μ_{ig}^k is the mean of sample i in cluster k for gene g, π_k are the mixing parameters, s_i is a normalization factor for each sample (as in Robinson et al. [2010], Risso et al. [2018]), and θ_g refers to other gene-specific parameters in the likelihood, such as the variance/dispersion.

We assume the mean μ_{ig} consists of a shared temporal pattern for cluster k, appropriately scaled and centered for each gene,

$$g(\mu_{ig}) = s_i + b_g + a_g \lambda_k^{C_i}(t),$$

where g is the appropriate link function, such as log, commonly used for modeling count-based sequencing data. We constrain $\lambda_k^C(t)$ to lie between 0 and 1 and let b_g and a_g be gene-specific scalars that appropriately adjust the scale-free pattern $\lambda_k^C(t)$ to the scale of the gene. We model the temporal pattern $\lambda_k^C(t)$ with a spline-basis function,

$$\lambda_k^C(t) = \sum_l \beta_{kl}^C \Phi_l(t)$$

where $\mathbf{\Phi}(t) = [\Phi_1(t), \ldots, \Phi_d(t)]^T$ is a pre-defined basis, and $\beta_k^C = [\beta_{k1}^C, \ldots, \beta_{kd}^C]^T$ is a d-dimensional unknown vector.

This model extends the model of Ma et al. [2006] in several critical ways. The model of Ma et al. [2006] is for the setting of normally-distributed microarray data, which is problematic for count data. For mRNA-Seq data, taking the logarithm of the y_{ig} and filtering low-count genes might suffice for clustering using a normal distribution, but in the context of scRNA-Seq or 16S sequencing, such a strategy will not account for either the high preponderance of low-count nor for the zero-inflation found in these data. Furthermore, the model of Ma et al. [2006] also does not provide for the scaling parameter a_g, meaning that they assume the temporal patterns are on the same scale across genes within a cluster, but this has the possibility of failing to find shared patterns of expression. Finally, since a_g can take on negative values in our model, our model seamlessly allows for the clustering together of genes with exactly inverted patterns. This can be of great biological interest as these inversions reveal anti-correlated behavior, indicating genes that behave in similar but opposite manners across time. This could suggest biologically significant relationships between genes behaving in such a manner. Within clusters containing anti-correlated genes, the sign of a_g of each gene can allow for further separation of genes between those that follow the pattern versus its inversion.

3.3.1 Parameter estimation details

We use a standard EM algorithm for estimating the parameters involving the cluster mean $\mu_g^k(t)$ and the mixing parameters π_k. The EM algorithm involves an iterative process of

estimating cluster parameters and cluster memberships. Specifically, let $Z_{gk} = 1$ if gene g belongs to cluster k and $Z_{gk} = 0$ otherwise. The cluster memberships

$$Z = \{Z_{gk}; g = 1, ...G; k = 1, ...K\}$$

are treated as missing data, and the EM algorithm iteratively calculates the conditional expectations of Z and updates the estimates of model parameters until convergence.

To initialize the EM algorithm, we are careful to use a good initialization point, since the EM algorithm is known to be very dependent on its starting values. To find this initialization point, we first apply K-means++ clustering on the matrix of fitted values obtained from fitting a per-gene spline model to the original data in the Normal setting or the log-transformed data in the case of count data. We repeat this K-means++ clustering multiple times and identify the K-means++ clustering with the highest overall likelihood, indicating the best starting position. This best clustering is used to obtain the cluster labels and associated model parameters to initialize the mixture model.

For the E-step, calculate the conditional expectation of Z_{gk} given parameters $\lambda^{(m)}, \pi^{(m)}, a^{(m)}$, and $b^{(m)}$ estimated from the mth step. Let $\hat{Z}_{gk}^{(m)}$ denote the conditional expectation

$$\hat{Z}_{gk}^{(m)} = \mathbb{E}(Z_{gk}|y, \lambda^{(m)}, \pi^{(m)}, a^{(m)}, b^{(m)}).$$

This expectation $\hat{Z}_{gk}^{(m)}$ is calculated by

$$\hat{Z}_{gk}^{(m)} = \frac{\pi_k^{(m)} f(y_g|a_{gk}^{(m)}, b_{gk}^{(m)}, \lambda_k^{(m)})}{\sum_l \pi_l^{(m)} f(y_g|a_{gl}^{(m)}, b_{gl}^{(m)}, \lambda_l^{(m)})}.$$

For the M-step, update the parameter estimates by

$$\pi_k^{(m+1)} = \frac{1}{G} \sum_g \hat{Z}_{gk}^{(m)}$$

$$\lambda_k^{(m+1)} = \underset{\{0 \leq \lambda_{ki} \leq 1\}}{\arg\max} \sum_g \hat{Z}_{gk}^{(m)} \log f(y_g|a_{gk}^{(m)}, b_{gk}^{(m)}, \lambda_k)$$

$$(a_{gk}^{(m+1)}, b_{gk}^{(m+1)}) = \underset{\{a_{gk}, b_{gk}\}}{\arg\max} f(y_g|b_{gk}, a_{gk}, \lambda_k^{(m+1)}).$$

The algorithm iterates between the E-step and M-step until the change in log likelihood is less than a small number, indicating convergence. Upon convergence, cluster labels are obtained by assigning gene g to cluster k, where $k = \arg\max_l \hat{Z}_{gl}$.

To estimate the parameters θ_g, we could estimate θ_g at each M-step along with the mean, scale, and shift parameters, and in the case of a Gaussian likelihood, where θ_g corresponds to the variance parameter, this is what we do. However, in the case of other distributions, such as the negative binomial or zero-inflated negative-binomial, this greatly increases the

computational overhead. Instead, we estimate the parameters θ_g initially using a per-gene spline model. In the setting of the negative binomial distribution, we estimate sample normalization factors and gene-specific dispersion parameters using the `edgeR` package from Robinson et al. [2010]. For the zero-inflated negative binomial setting, we estimate sample normalization factors, gene-specific dispersion parameters, and zero-inflation parameters using the `zinbwave` package from Risso et al. [2018]. We then assume these parameters to be known in the EM algorithm.

While the EM algorithm is commonly applied in the context of normal mixture distributions, it can also be applied in a count data setting, as described in McLachlan [1997]. Optimization of the parameter estimates at each step is achieved using the L-BFGS-B quasi-Newton method for the count distributions, and the closed-form solutions for the Normal model, described in detail in the next section.

3.4 Optimization details

At each M-step, the EM algorithm updates the estimates of the cluster centroids and gene parameters by maximizing the likelihood. Optimization details for the M-step of the EM algorithm are provided in this section.

3.4.1 NB and ZINB models

For the NB and ZINB likelihoods, there is not a closed-form solution for the maximum likelihood estimates. Therefore, optimization of the parameter estimates at each step of the EM algorithm is achieved using the L-BFGS-B quasi-Newton method. This method requires computing the derivatives of the likelihood. These derivatives are computed with respect to the mean μ, as well as the coefficients β and gene-specific parameters α and w, and are calculated as follows.

Denote the probability mass function of the negative binomial distribution with mean μ and inverse dispersion parameter θ as

$$f_{NB}(y;\mu,\theta) = \frac{\Gamma(y+\theta)}{\Gamma(y+1)\Gamma(\theta)}\left(\frac{\theta}{\theta+\mu}\right)^{\theta}\left(\frac{\mu}{\mu+\theta}\right)^{y},\ \forall y \in \mathbb{N}.$$

The probability mass function of the ZINB distribution can be written as

$$f_{ZINB}(y;\mu,\theta,\pi) = \pi\delta_0(y) + (1-\pi)f_{NB}(y;\mu,\theta),\ \forall y \in \mathbb{N},$$

where $\delta_0(\cdot)$ is the Dirac function.

For a particular gene g, the derivative of the negative binomial log likelihood with respect to μ is computed by

$$\frac{\partial}{\partial\mu}\log f_{NB}(y;\mu,\theta) = \frac{y}{\mu} - \frac{y+\theta}{\mu+\theta}.$$

We can then use this to obtain the derivative of the ZINB log likelihood with respect to μ by first writing

$$\frac{\partial}{\partial \mu} \log f_{ZINB}(y; \mu, \theta, \pi) = \frac{(1-\pi) f_{NB}(y; \mu, \theta) \frac{\partial}{\partial \mu} \log f_{NB}(y; \mu, \theta)}{f_{ZINB}(y; \mu, \theta, \pi)}.$$

Then for a single count y, if $y > 0$,

$$\frac{\partial}{\partial \mu} \log f_{ZINB}(y; \mu, \theta, \pi) = \frac{\partial}{\partial \mu} \log f_{NB}(y; \mu, \theta) = \frac{y}{\mu} - \frac{y+\theta}{\mu+\theta}$$

and if $y = 0$,

$$\frac{\partial}{\partial \mu} \log f_{ZINB}(0; \mu, \theta, \pi) = \frac{-(1-\pi)\left(\frac{\theta}{\mu+\theta}\right)^{\theta+1}}{\pi + (1-\pi)\left(\frac{\theta}{\mu+\theta}\right)^{\theta}}.$$

Finally, to obtain derivatives with respect to β, a, and w, let $L = \log f(y; \mu, \theta, \pi)$, where $\mu = b + aX\beta$, and f follows either the NB or ZINB distribution. The chain rule implies

$$\frac{\partial L}{\partial \beta} = \frac{\partial L}{\partial \mu}\frac{\partial \mu}{\partial \beta} = X^T \mu \frac{\partial L}{\partial \mu}$$

$$\frac{\partial L}{\partial a} = \frac{\partial L}{\partial \mu}\frac{\partial \mu}{\partial a} = X\beta \frac{\partial L}{\partial \mu}$$

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial \mu}\frac{\partial \mu}{\partial b} = 1 \frac{\partial L}{\partial \mu}.$$

With these derivatives and the given likelihood function, the L-BFGS-B optimizes the model parameters β, α, and b at each iteration.

3.4.2 Normal model

Optimization of model parameters with the Normal distribution is much simpler. For the Normal distribution, maximizing the likelihood is equivalent to minimizing the residual sums of squares. For cluster k, parameter estimates in the M-step are updated by:

$$b_{gk}^{(m+1)} = \frac{1}{J}\sum_{j=1}^{J}(y_{gj} - a_{gk}^{(m)}\lambda_{kj}^{(m+1)})$$

$$a_{gk}^{(m+1)} = \frac{\sum_{j=1}^{J}\lambda_{kj}^{(m+1)}(y_{gj} - b_{gk}^{(m+1)})}{\sum_{j=1}^{J}\lambda_{kj}^{(m+1)}\lambda_{kj}^{(m+1)}}$$

$$\sigma_k^{2(m+1)} = \sum_{g=1}^{G}\left[\left(\frac{p_{gk}}{\sum_{g=1}^{G} p_{gk}}\right)\sum_{j=1}^{J}(y_{gj} - b_{gk}^{(m+1)} - a_{gk}^{(m+1)}\lambda_{jk}^{(m+1)})^2\right].$$

Note that in the Normal model, we are additionally estimating the cluster variance, σ_k^2. We now have all the elements needed to write down the likelihood of the model.

$$\begin{aligned}\mathcal{L}(y,\pi,\mu) &= \prod_i \left(\sum_k \pi_k \prod_j \mathcal{N}\left(y_{ij} | a_i \sum_d \beta_d^k \phi_d(t_j) + b_i, \sigma_k^2 \right) \right) \\ &= \prod_i \left(\sum_k \pi_k \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(\frac{-1}{2\sigma_k^2} \sum_j \left(a_i \sum_d \beta_d^k \phi_d(t_j) + b_i - y_{ij} \right)^2 \right) \right).\end{aligned}$$

Now taking the log likelihood does not simplify nicely (as always for Gaussian mixture models):

$$\begin{aligned}\ell(y,\pi,\mu) &= \log \mathcal{L}(y,\pi,\mu) \\ &= \sum_i \log \left(\sum_k \pi_k \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(\frac{-1}{2\sigma_k^2} \sum_j \left(\sum_d \beta_d^k \phi_d(t_j) - y_{ij} \right)^2 \right) \right)\end{aligned}$$

On the other hand, we now have a log likelihood of (almost) the same form as a multivariate Gaussian mixture model with a diagonal covariance matrix. We can apply the same strategy as on a standard Gaussian mixture model, with a slight variation in the mean fit by computing

$$\frac{\partial \ell}{\partial \mu_l}(y,\pi,\mu) = \sum_i \frac{\pi_l \frac{1}{\sqrt{2\pi\sigma_l^2}} \exp\left(\frac{-1}{2\sigma_l^2} \sum_j \left(\mu_l(t_j) - y_{ij} \right)^2 \right)}{\sum_k \pi_k \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(\frac{-1}{2\sigma_k^2} \sum_j \left(\mu_k(t_j) - y_{ij} \right)^2 \right)} \frac{-1}{2\sigma_k^2} \sum_j \frac{\partial}{\partial \mu_k(t_j)} \left(\mu_k(t_j) - y_{ij} \right)^2$$

To simplify notation, denote the probability of gene i belonging to cluster k as τ_{ik}, i.e.,

$$\tau_{ik} = \frac{\pi_l \frac{1}{\sqrt{2\pi\sigma_l^2}} \exp\left(\frac{-1}{2\sigma_l^2} \sum_j \left(\mu_l(t_j) - y_{ij} \right)^2 \right)}{\sum_k \pi_k \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(\frac{-1}{2\sigma_k^2} \sum_j \left(\mu_k(t_j) - y_{ij} \right)^2 \right)}.$$

Then we can write

$$\frac{\partial \ell}{\partial \mu_l}(y,\pi,\mu) = \sum_i \tau_{ik} \frac{-1}{2\sigma_k^2} \sum_j \frac{\partial}{\partial \mu_k(t_j)} \left(\mu_k(t_j) - y_{ij} \right)^2$$

Finally, we proceed similarly to estimate the variance, and obtain a closed form solution:

$$\sigma_l^2 = \frac{\sum_i \tau_{il} \sum_j \left(\mu_l(t_j) - y_{ij} \right)^2}{\sum_i \tau_{il}}.$$

3.5 Implementation details

The mixture model is implemented in an R package, which allows a user to input a dataset, specify the desired number of clusters and model formula, as well as to choose the appropriate data distribution. For the normal model, the data is assumed to be normalized before applying the clustering function. For the count models, the user can either input sample normalization factors or allow the mixture model function to compute these parameters as part of the model. The method returns the clustering results in the form of cluster labels, cluster centroids, and gene-cluster membership probabilities, as well as the gene-specific parameters. An additional option to test cluster means for differential expression described in the next chapter is also available a function that uses the results of running the mixture model clustering. Pseudo-code describing the complete mixture model clustering procedure is given in Algorithm 1.

Algorithm 1: Mixture model clustering algorithm

Data: Gene expression data matrix y
Result: Optimized parameters: centroids μ, shifts b, scalings a, cluster probabilities t, and cluster labels
if y *is count data* **then**
  Compute fixed parameters:
    $s \leftarrow$ sample normalization factors;
    $\theta \leftarrow$ dispersion estimates;
    $\pi \leftarrow$ zero-inflation factors;
    return $\Psi = \{s, \theta, \pi\}$;
  ;
Initialize *data***:**
  apply K-means to data;
  $\mu^0 \leftarrow$ K-means centroids;
  $(a^0, b^0) \leftarrow \arg\max_{\{a,b\}} f(y|a, b, \mu = \mu_0, \Psi)$;
  $t^0 \leftarrow \mathbb{E}(Z|y, \mu^0, a^0, b^0, \Psi)$;
  return μ^0, a^0, b^0, t^0;
;

for $m \leftarrow 0$ **to** *max iterations* **do**
  E-step `Update`$(t^{(m)})$**:**
    $t^{(m+1)} \leftarrow \mathbb{E}(Z|y, \mu^{(m)}, a^{(m)}, b^{(m)}, \Psi)$;
    return $t^{(m+1)}$;
  ;
  M-step `Update`$(\mu^{(m)}, a^{(m)}, b^{(m)})$**:**
    for $k \leftarrow 1$ **to** K **do**
      $\mu_k^{(m+1)} \leftarrow \arg\max \sum_g \hat{Z}_{gk}^{(m)} \log f(y_g|a_{gk}^{(m)}, b_{gk}^{(m)}, \mu_k^{(m)}, \Psi)$;
      $(a^{(m+1)}, b^{(m+1)}) \leftarrow \underset{\{a_k, b_k\}}{\arg\max} f(y|b_k, a_k, \mu_k = \mu_k^{(m+1)}, \Psi)$;
      Rescale parameters `Rescale`$(\mu_k^{(m+1)}, a_k^{(m+1)}, b_k^{(m+1)})$**:**
        $b_k^{(m+1)} \leftarrow b_k^{(m+1)} + a_k^{(m+1)} * \min \mu_k$;
        $a_k^{(m+1)} \leftarrow a_k^{(m+1)} * (\max \mu_k - \min \mu_k)$;
        $\mu_k^{(m+1)} \leftarrow \frac{(\mu_k - \min \mu_k)}{(\max \mu_k - \min \mu_k)}$;
        return $\mu_k^{(m+1)}, a_k^{(m+1)}, b_k^{(m+1)}$;
      ;
    return $\mu^{(m+1)}, a^{(m+1)}, b^{(m+1)}$;
  ;
  $change \leftarrow \mathcal{L}(y; \mu^{(m+1)}, a^{(m+1)}, b^{(m+1)}, t^{(m+1)}, \Psi) - \mathcal{L}(y; \mu^{(m)}, a^{(m)}, b^{(m)}, t^{(m+1)}, \Psi)$;
  if $change < \epsilon$ **then**
    end;
Assign cluster labels `Classify`$(t^{(m)})$**:**
  for $g \leftarrow 1$ **to** G **do**
    $labels_g \leftarrow \underset{\{1 \le k \le K\}}{\arg\max}\ t_{gk}^{(m+1)}$;
  return *labels* ;
;
return $\mu^{(m+1)}, a^{(m+1)}, b^{(m+1)}, t^{(m+1)}, labels$;

3.6 Simulation results

To assess the performance of the mixture model, we ran many simulations of the mixture model under various settings. The primary focus of the simulations was to compare the clustering performance of the mixture model against the alternative clustering methods discussed earlier. In addition, we investigated the advantages of the various model specifications unique to our mixture model framework. Finally, the estimated times required to run each method were compared.

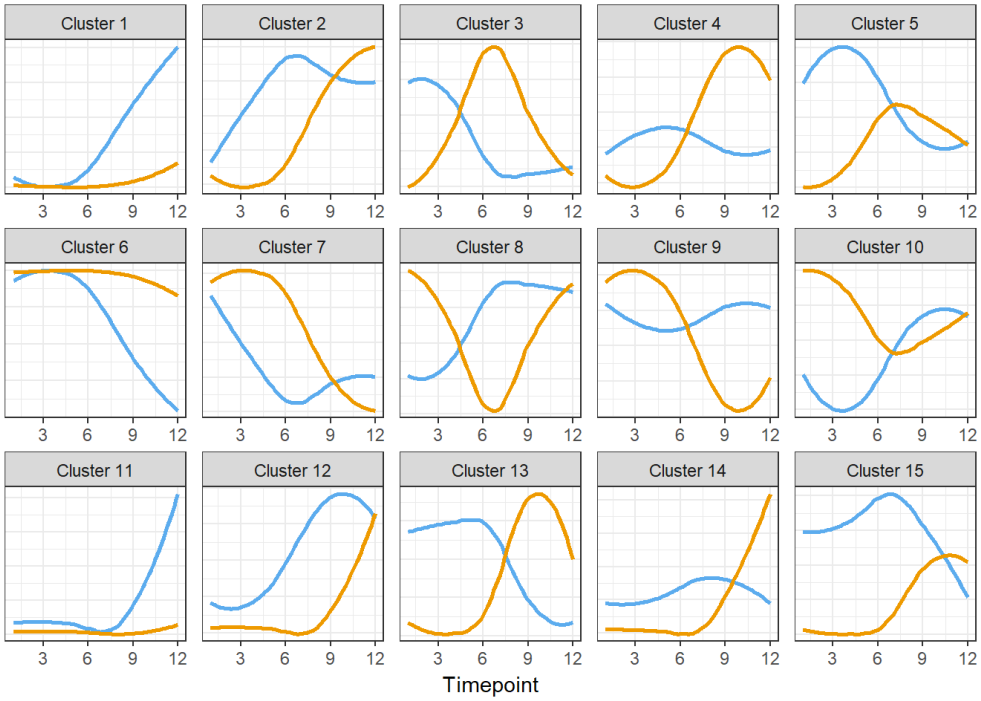


Figure 3.1: Visualization of the 15 simulated cluster patterns.

3.6.1 Simulated data description

In order to simulate realistic genomic data, simulated data sets were generated based on 15 patterns observed in the EPICON 16S microbial data set discussed in the previous chapters. The cluster patterns vary across 12 time points under two different conditions, and Figure 3.1 provides a visualization of these 15 dynamic cluster patterns. The first five patterns are each distinct patterns observed in the microbial data set. Patterns six through ten are the flipped images of the first five patterns and represent clusters that should be combined with patterns one through five with a scaling factor of the opposite sign. For example, cluster one and cluster six should be combined into one cluster in our mixture model framework, with the genes in cluster one receiving scaling parameters that are the negative of the gene scaling factors assigned to genes in cluster six. Patterns 11 through 15 are shifted versions of the first five patterns, shifted three time points to the right, in order to observe how accurately the mixture model can distinguish similar but different patterns. Finally, a 16th cluster of random noise was also included. These 16 patterns formed the centroids of 16 clusters, and gene expression datasets were simulated according to a Normal, NB, or ZINB distribution based on these cluster means.

Within each cluster, 100 genes were simulated with a noisy version of the cluster mean according to the cluster patterns shown in Figure 3.1, and each gene was further modified by

a gene-specific overall mean expression level and a gene-specific scaling factor. Each gene's mean followed the same pattern as its assigned cluster mean but with a small amount of gene-specific variation around this pattern, which was simulated by adding small amounts of noise to the coefficients of the cluster spline parameters. Gene-specific mean expression levels were generated from a Normal(12, 1) distribution. Gene-specific scaling parameters were generated from a Uniform(1, 5) distribution. Gene-specific dispersion parameters for the count models were generated from a Gamma distribution with shape according to the gene-specific mean and scale equal to 1/2. In addition, sample-specific library sizes were generated according to a Normal(90000, 1000) distribution.

Once the gene specific versions of the cluster mean, the scaling parameters, and the shift parameters were generated, the gene expression values were simulated by random draws from one of the three distributions of interest, the Normal, NB, or ZINB distributions. To simulate the setting of zero inflated data, datasets were simulated with probabilities of zero ranging from 0 to 0.8, where the setting of a zero inflated level of 0 represents the purely NB case. For each level of zero-inflation, where the proportion of zeros ranged from 0 to 0.8, 20 datasets were simulated. Clustering performance was based on the mean Adjusted Rand Index across these 20 replications at each level of zero-inflation.

3.6.2 Clustering performance

In order to compare the performance of the various clustering methods and investigations of model parameters, we ran each method on simulated datasets and computed the Adjusted Rand Index to measure the agreement between the observed clustering and the true cluster labels. The Adjusted Rand Index (ARI), proposed by Hubert and Arabie [1985], is a measure of agreement between two clusterings corrected for the random chance of agreement. The measure is based on the number of overlaps between any clusters. To be precise, suppose we have two clusterings $X = \{X_1, X_2, ..., X_r\}$ and $Y = \{Y_1, Y_2, ..., Y_s\}$. Their overlap can be summarized in a table with entries counting the size of the overlap, $n_{ij} = |X_i \bigcap Y_j|$. Such a table appears below.

	Y_1	Y_2	...	Y_s	Sums
X_1	n_{11}	n_{12}	...	n_{1s}	a_1
X_2	n_{21}	n_{22}	...	n_{2s}	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
X_r	n_{r1}	n_{r2}	...	n_{rs}	a_r
Sums	b_1	b_2	...	b_s	

From this table of overlaps, the Adjusted Rand Index is computed as

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - [\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}]/\binom{n}{2}}{\frac{1}{2}[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}] - [\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}]/\binom{n}{2}}.$$

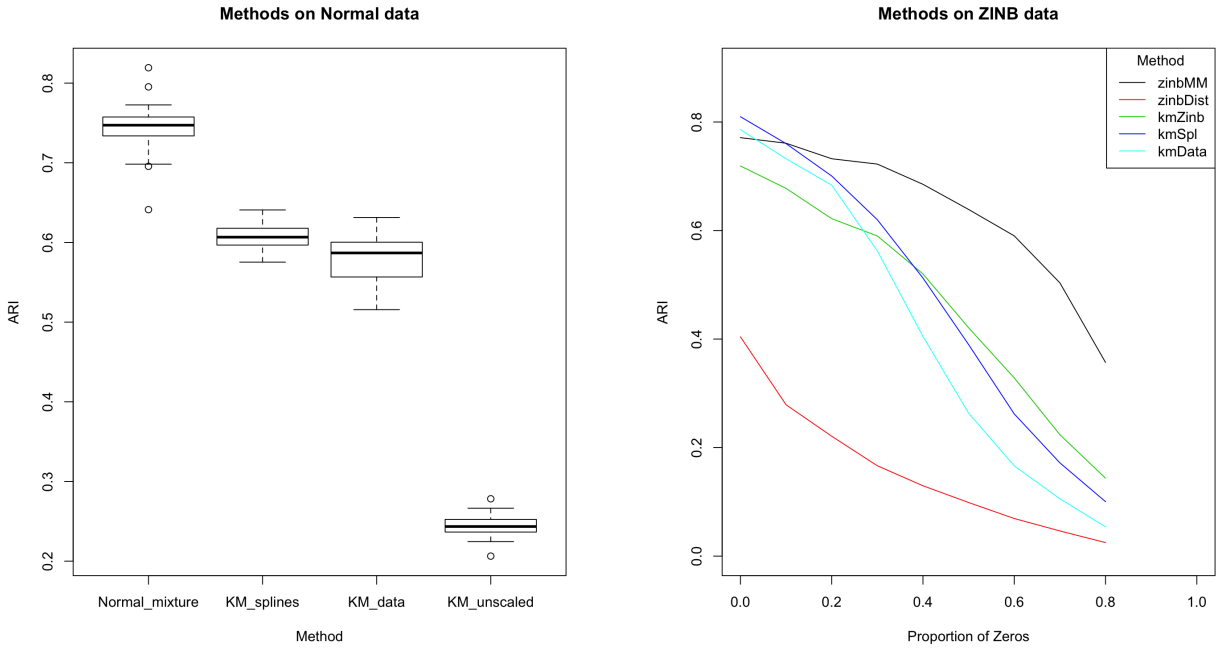
Here, the numerator represents the total number of overlaps, adjusted for random chance. The denominator represents the maximum possible number of overlaps, corrected for random chance. The Adjusted Rand Index can be negative but is bounded above by 1, where an ARI of 0 indicates an agreement no better than random change, and higher ARIs indicate greater agreement between clusterings. Hence, using this measure of cluster agreement gives us a way to determine how successfully a clustering method performs, by comparing the observed cluster labels to the true, known cluster labels.

In order to compare the clustering results of our mixture model to other available clustering methods, we have to first make a consideration of how to treat the flipped clusters. As described earlier, the intent in designing our simulated clusters was for the first five cluster patterns to be clustered together with the second five cluster patterns. However, as our mixture model framework is the only available method with the capability to cluster flipped patterns together, we need a way to enable a direct comparison with those methods that do not combine the flipped patterns. This is a simple process, where for the mixture model clusters that contain genes with both positive and negative scaling factors, we separate these clusters into two clusters based on the sign of each gene's scale parameter. This essentially creates a new, larger set of clusters, where the anti-correlated clusters have been divided into two clusters of only correlated genes. This provides a reasonable set of clusters that can be compared to the clusters achieved by alternative clustering methods.

Comparison of clustering performance between methods

First, we consider simulations from Normally distributed data, and we compare our Normal mixture model to three versions of K-means discussed previously. These K-means methods are to run K-means on the spline fitted values, K-means on the centered and scaled data, and K-means on the original unscaled data. The results of 50 simulations are shown in Figure 3.2a, where the distribution of the Adjusted Rand Index has been plotted for each of the four methods. From these results, it is clear that the Normal mixture model consistently achieves the highest Adjusted Rand Index, indicating the greatest level of agreement with the true cluster labels. Among the three K-means options, we observe that scaling the data is an essential step to perform before clustering, as this greatly increases the resulting ARIs. In addition, applying K-means to the spline fitted values yields higher ARIs over the other K-means options. This indicates that incorporating the time course model information is beneficial to the clustering procedure. The full Normal mixture model is then able to provide an additional advantage, indicating that the mixture model provides further advantages beyond the use of the spline model.

Next, we compare the methods designed for a zero-inflated negative binomial setting. Here, we have simulated data at various levels of zero-inflation, and the average ARI at each level is shown in Figure 3.2b. The ZINB mixture model has comparable or higher ARIs than the other methods at all levels of zero-inflation, and in particular has significantly higher ARIs at the higher levels of zero-inflation, indicating that accounting for zero-inflation is more necessary as the proportion of zeros increases. Each of the K-means options have



(a) Distribution of ARI for each method on Normal data.

(b) Mean ARI for each method on ZINB data across varying levels of zero inflation.

Figure 3.2: Comparison of clustering methods. The distribution of the Adjusted Rand Index across 50 simulations is plotted for each method of comparison. On the normal data, the methods are the Normal mixture model, K-means on spline fitted values, K-means on the centered and scaled data, and K-means on the original (unscaled) data. On the ZINB data, the methods are the ZINB mixture model, the hierarchical clustering ZINB distance method, K-means on ZINB spline fitted values, K-means on Normal spline fitted values, and K-means on centered and scaled data.

high ARI performance in the setting with low zero-inflation, but as the proportion of zeros increases, the K-means options that do not incorporate zero-inflation struggle to cluster well and their ARIs drop rapidly. Among the K-means options, there is an increase in ARI when fitting a spline model or ZINB spline model before applying K-means, again supporting our hypothesis that a spline model is beneficial to a clustering procedure. Finally, all methods outperform the hierarchical clustering method that uses the zero-inflated distance measure. Although this method accounts for the excess zero-inflation, it does not take into account the model information of time or condition, which likely explains its low ARIs across all levels of zero-inflation.

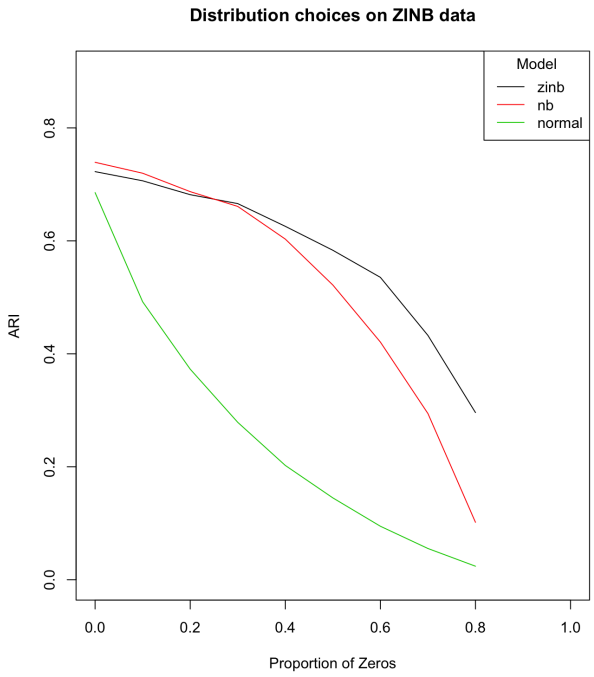
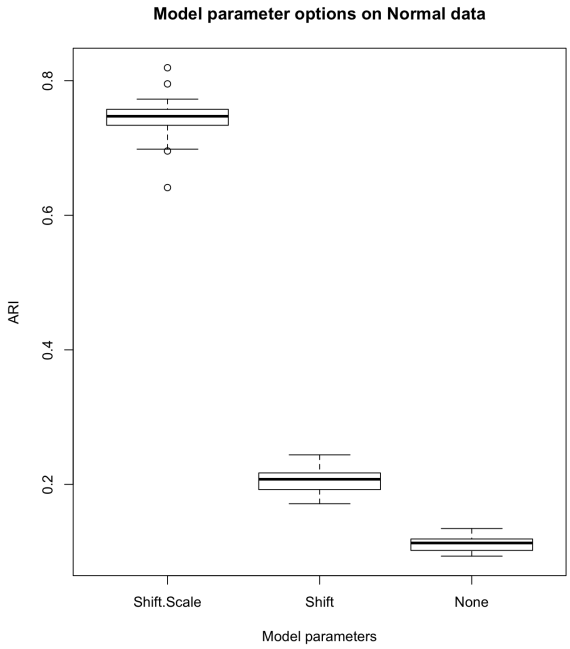


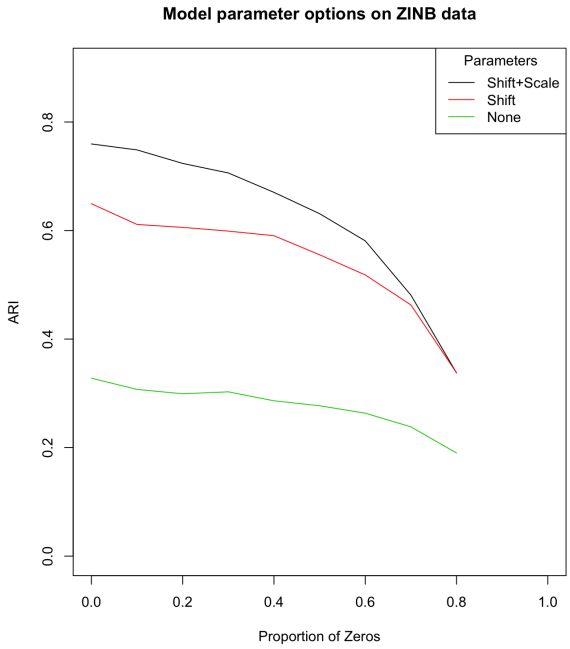
Figure 3.3: Comparison of distribution choices. The mean Adjusted Rand Index across 20 simulations is plotted for the results of running the mixture model with each of the three distribution choices: ZINB, NB, and Normal. The simulated data is from the ZINB distribution.

Effect of distribution choices

The mixture model provides the option to use the normal, negative binomial, or zero-inflated negative binomial distribution. To examine the effect of distribution choice on clustering performance, we simulated data under the ZINB distribution with varying proportions of zero inflation and ran the mixture model under each of the three distribution choices. As shown in Figure 3.3, the choice of distribution has a significant effect on clustering performance. As expected, since the simulated data are counts, the normal model does not perform well at any level of zero-inflation except the setting with no zeros. However, in the setting with no zeros, the normal model does perform remarkably well. In this kind of setting such as with bulk mRNA data, the negative binomial model may not provide a significant advantage. Since the negative binomial model is a much more computationally expensive option, the normal model could provide a preferable option in such cases. The negative binomial and zero-inflated negative binomial perform similarly at low levels of zero-inflation; as the proportion of zeros in the data increases, the ZINB distribution provides a greater advantage over the negative binomial.



(a) Advantage of including gene-specific shift and scale parameters with Normal data.



(b) Advantage of including gene-specific shift and scale parameters with ZINB data.

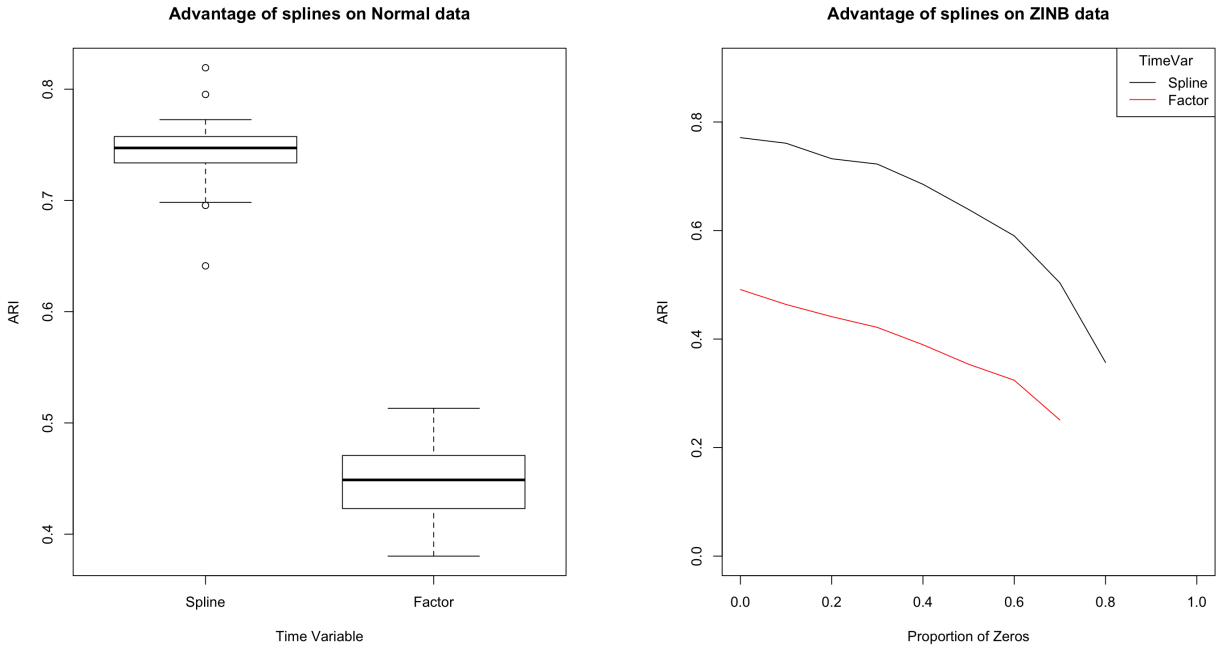
Figure 3.4: Advantage of model parameters. The mean Adjusted Rand Index across simulations is plotted for each of three model options: first, the model including both gene-specific parameters, second, the model with just a shift parameter, and third, the model with neither parameter.

Advantage of gene-specific parameters

To assess the effects of including the gene-specific shift and scale parameters, we compared the clustering performance under three different versions of the mixture model: one that included both the shift and scale parameters, one with just a shift parameter, and one with neither parameter. As shown in Figure 3.4, including both the shift and scale parameters provides an increase in clustering performance over the options that do not include the gene-specific parameters, both in the Normal setting as well as in the ZINB setting at all levels of zero-inflation.

Advantage of splines for time variable

Finally, to determine whether the spline model is useful to represent the time variable, we compared the performance of the mixture model using a spline model to represent the time variable against the mixture model using as a factor variable to model the time variable. As



(a) Distribution of the ARI for each model of the time variable on Normal data.

(b) Mean ARI across varying levels of zero inflation for each time model on ZINB data.

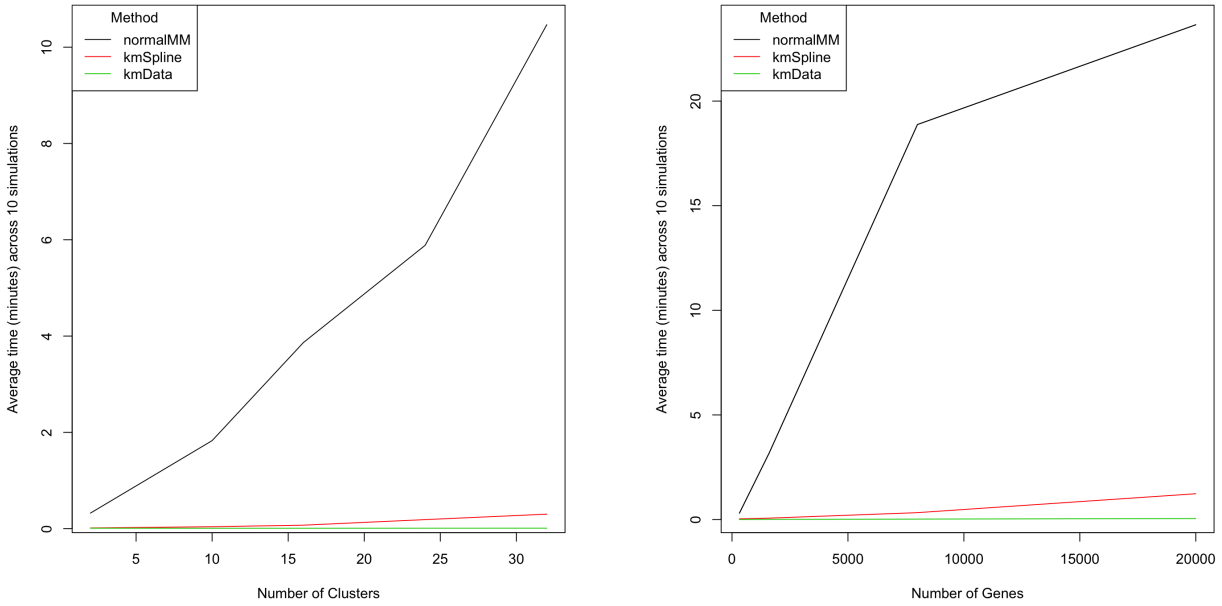
Figure 3.5: Advantage of spline model. The Adjusted Rand Index across simulations is plotted for each model option: first, the model using a spline model to represent the time variable, and second, the model using a factor variable to represent the time variable.

shown in Figure 3.5, the model that uses splines consistently outperforms the model that treats time as a categorical variable, under both the Normal distribution setting as well as the ZINB distribution setting at all levels of zero inflation.

3.6.3 Timing performance

Aside from clustering performance, a serious consideration to be made when choosing a clustering method is the amount of time needed to run the clustering method. This is a complicated question to answer, since the amount of time needed for the mixture model to converge depends on the number of genes, the number of clusters, and the choice of distribution.

To explore how the number of genes affects clustering times, datasets were simulated with genes ranging from 320 genes to 20,000 genes. The clustering time was collected for the various clustering methods, clustering with a fixed number of 16 clusters. To explore how the number of clusters affects clustering times, one dataset was simulated with 2000 genes and the timings of the various clustering methods were collected for varying numbers



(a) Average time to run clustering for various numbers of clusters.

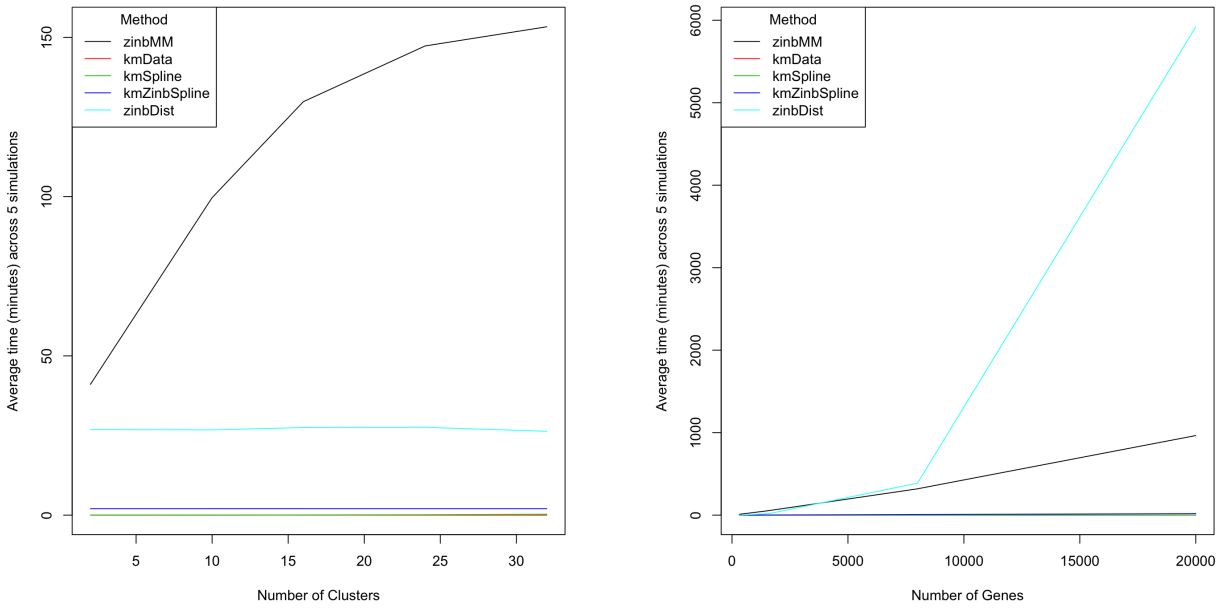
(b) Average time to run clustering for various numbers of genes.

Figure 3.6: Average run time of various clustering methods on Normal data.

of clusters ranging from 2 to 32 clusters. In each case, simulated datasets were generated both from a Normal distribution and a ZINB distribution. Between the two distributions, the Normal mixture model runs much faster than the ZINB model, since the Normal model can take advantage of closed-form solutions at each iteration of the EM algorithm.

Figure 3.6 shows the average time to run each clustering method using data simulated from a Normal distribution as the number of genes increases and the number of clusters increases. The timing for the Normal mixture model is compared to the timing for K-means on the original data and the timing for K-means on fitted splines. The Normal mixture model takes significantly longer than the alternative methods, with larger differences as the number of clusters and genes increase, although the demonstrably improved clustering performance by the mixture model likely makes the slight increase in time a worthwhile investment.

Figure 3.7 shows the timing results from clustering on data simulated from a ZINB distribution. The methods compared on the ZINB datasets are our ZINB mixture model, K-means on the scaled log-counts, K-means on the fitted splines of the log-counts, K-means on the fitted ZINB splines, and the ZINB distance method adapted from Witten et al. [2011]. The mixture model using the ZINB model suffers from increasingly long clustering times. However, the ZINB distance method suffers the most from an increase in the number of



(a) Average time to run clustering for various numbers of clusters.

(b) Average time to run clustering for various numbers of genes.

Figure 3.7: Average run time of various clustering methods on ZINB data.

genes, due to the fact that a likelihood must be computed for each gene at each sample in the dataset; on the other hand, since it relies on hierarchical clustering once the dissimilarity matrix is computed, increasing the number of clusters does not increase its time at all. These results demonstrate the major disadvantage of using the ZINB mixture model, which is simply unavoidable due to the time needed to optimize the model parameters under the ZINB distribution at each step in the EM algorithm. However, since the implementation of the mixture model takes advantage of parallelization in its optimization steps, running the mixture model on multiple cores can vastly improve clustering speed, making the cost of time significantly less prohibitive.

3.7 Real data results

To demonstrate the application of this method to real data, the mixture model was fit to two real data sets, both coming from the EPICON project described in Chapter 2. First, the model was applied to the 16S microbial dataset analyzed earlier, which provided an opportunity to apply the count-specific option using the ZINB model. Second, the model was applied to an mRNA transcriptomic dataset, where the Normal model has been applied.

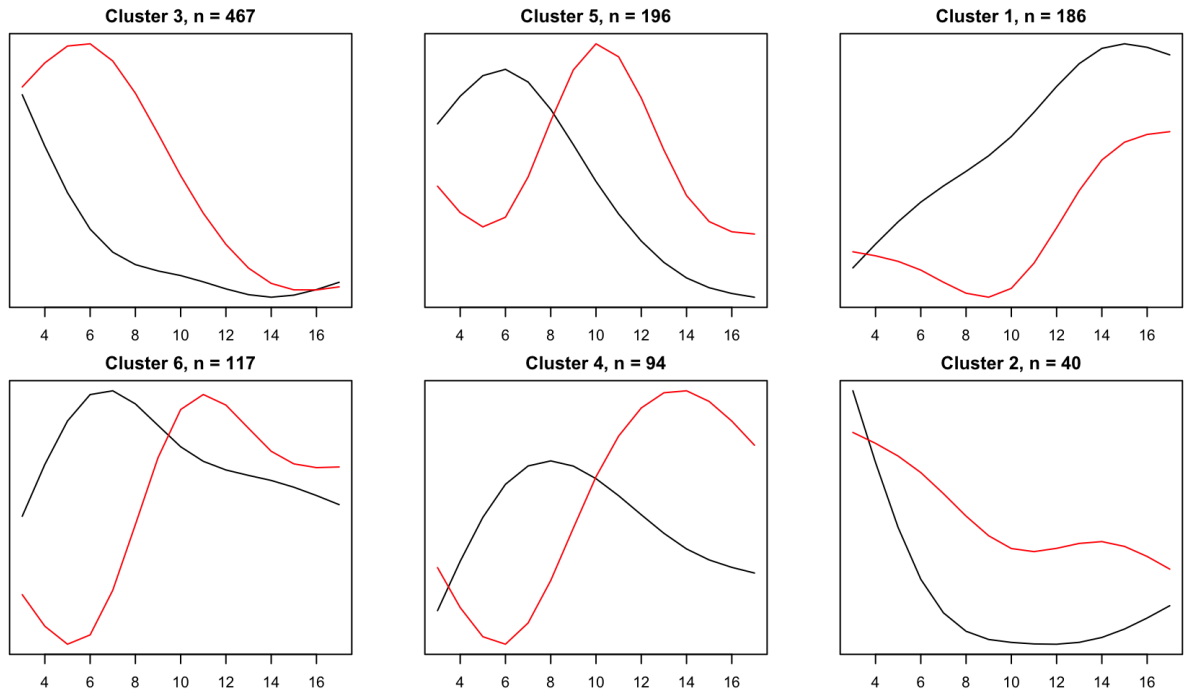


Figure 3.8: Cluster centroids obtained from running the mixture model on the 16S microbial data, ordered from largest cluster size to smallest. The centroid is plotted in black for the control condition and in red for the pre-flowering drought condition.

3.7.1 16S microbial data

As discussed earlier, 16S sequencing data is count data and highly zero-inflated, so ZINB option is the most appropriate model to be used in clustering. To apply the mixture model, we are considering the data under both genotypes, under the Pre-flowering and Control conditions, across Time points 3 through 17. After filtering out OTUs with the lowest counts and highest numbers of zeros across all samples, the dataset contained 1100 OTUs across 176 samples.

To explore the results, we look first at the cluster centroids shown in Figure 3.8, where clusters are ordered by cluster size, from largest to smallest, and the cluster size in given in the title of each centroid's plot. The centroids confirm our previous observations of a general pattern of similar but delayed progression across time under the drought condition. In addition, we again observe a cluster demonstrating enrichment under drought. Thus, these centroids appear to be reasonable, as they confirm our previous results; however, we would like to ensure that the mixture model has clustered OTUs in a reasonable way by looking at the OTUs assigned to each cluster.

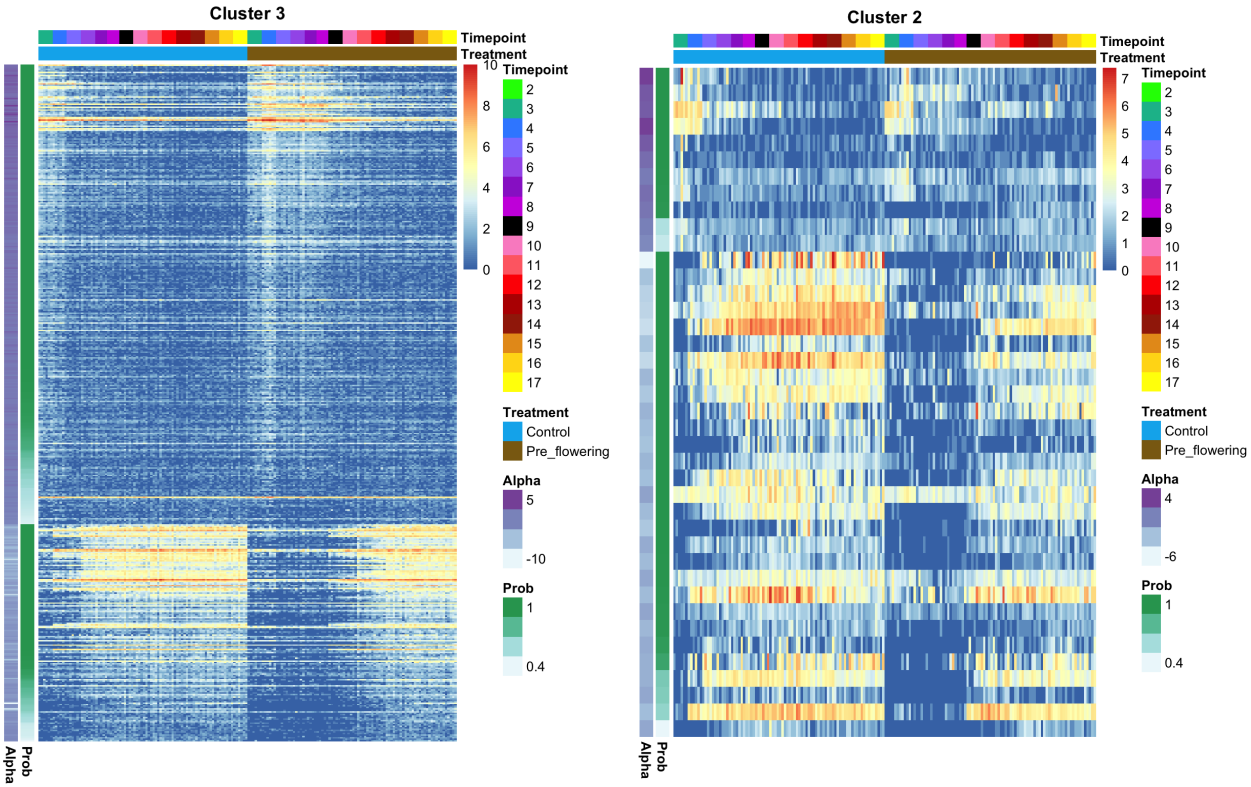
By looking at the abundances of the OTUs assigned to each cluster, we can get an idea of how accurately the changes in abundance of the clustered OTUs are represented by the

overall cluster mean. Heatmaps of OTU abundance provide a simple way to assess this question. Figure 3.9 provides heatmaps of the OTUs belonging to cluster 3 and cluster 2. These clusters were chosen because they are the largest and smallest cluster, and provide a good demonstration of the range of clustering behavior. Additional heatmaps from clusters 1 and 5 are provided and discussed later in Chapter 4. To read each heatmap, each row represents one of the OTUs assigned to that cluster. Each column represents a sample. The annotations to the left of the rows and above the columns provide further information about the OTUs and samples. On the left hand side, there are two tracks of classification: an *Alpha* variable, which represents the value of each OTU's scaling factor, and a *Prob* variable, which indicates the probability that each OTU belongs to this cluster. OTUs have been ordered by the sign of their scaling factor, and then from highest to lowest probability. Hence with this ordering, we can see two broad groups of OTUs demonstrating anti-correlated behavior. In both heatmaps, the upper group of OTUs follows the inverted cluster pattern, which the lower group of OTUs follow the cluster pattern shown in the Figure 3.8 centroids. The annotations on the top of the heatmap represent the sample information, where samples have been ordered according to their experimental condition and time point

The heatmaps from both clusters reveal the highly zero-inflated nature of microbial data; especially in Cluster 3, the majority of OTUs show a high proportion of zero counts across samples. Among the OTUs in Cluster 3, those with higher counts across more samples follow the overall cluster pattern well, as hoped. Cluster 2 represents a much noisier cluster, where the OTUs that follow the inverted cluster pattern (those OTUs with the negative scaling factors) seem to fit the cluster pattern well; however, the OTUs with positive scaling factors are much noisier and show less of a cohesive pattern. While the zero-inflated nature of the data clearly raises the difficulty of clustering, these heatmaps do indicate that the mixture model has found a reasonable clustering where the centroids are accurate representations of the overall cluster behavior.

One of the advantages of the parametrization that we have chosen in our mixture model is the inclusion of the gene-specific scaling factor. This allows us to identify, within each cluster, those OTUs that demonstrate the same pattern across time but with inverted behavior. These scaling factors thus identify OTUs with anti-correlated behavior. From the heatmaps in Figure 3.9, we can see two broad groups of OTUs showing anti-correlated behavior, which are identified by the *Alpha* annotation. These two anti-correlated groups have potentially great biological significance, if we consider the patterns of early enrichment shown by Actinobacteria discussed earlier. Cluster 3 contains these Actinobacteria OTUs showing early enrichment, but it also contains OTUs showing the opposite pattern of early depletion and later enrichment. From a biological perspective, this may suggest genuine biological relationships between the OTUs making up these two different groups within Cluster 3.

To see how well this pattern of anti-correlation carries over into other clusters, it is useful to look at the distribution of the scaling parameters of the genes assigned to each cluster. Figure 3.10 shows a histogram for each of the six clusters of the distribution of the gene-specific scaling parameter assigned to each of the genes belonging to that cluster. From these distributions, we can see that most clusters have a bimodal distribution of the



(a) Heatmap of OTUs in the largest cluster, with a total of 467 OTUs.

(b) Heatmap of OTUs in the smallest cluster, with a total of 40 OTUs.

Figure 3.9: Heatmaps of OTUs belonging to the smallest and largest cluster obtained from fitting the mixture model to the microbial data.

scaling factor, indicating that these clusters contain OTUs showing anti-correlated behavior. By clustering these OTUs in the same cluster, the mixture model provides a useful way to associate OTUs with inverted patterns of expression while still providing the ability to separate these anti-correlated groups for further comparison.

3.7.2 mRNA gene expression data

To consider a dataset with very different characteristics, we turn to the mRNA transcriptomic data coming from the roots of the EPICON project. This dataset was analyzed thoroughly in Varoquaux et al. [2019], and here, we apply our mixture model method to a subset of the data in that paper, focusing just on the root data from the RTx430 genotype under the pre-flowering drought and control conditions. This gene expression dataset is an mRNA-seq dataset; thus, a natural choice of the mixture model would be to use the Negative Binomial distribution. However, for convenience and speed, as well as to demonstrate the

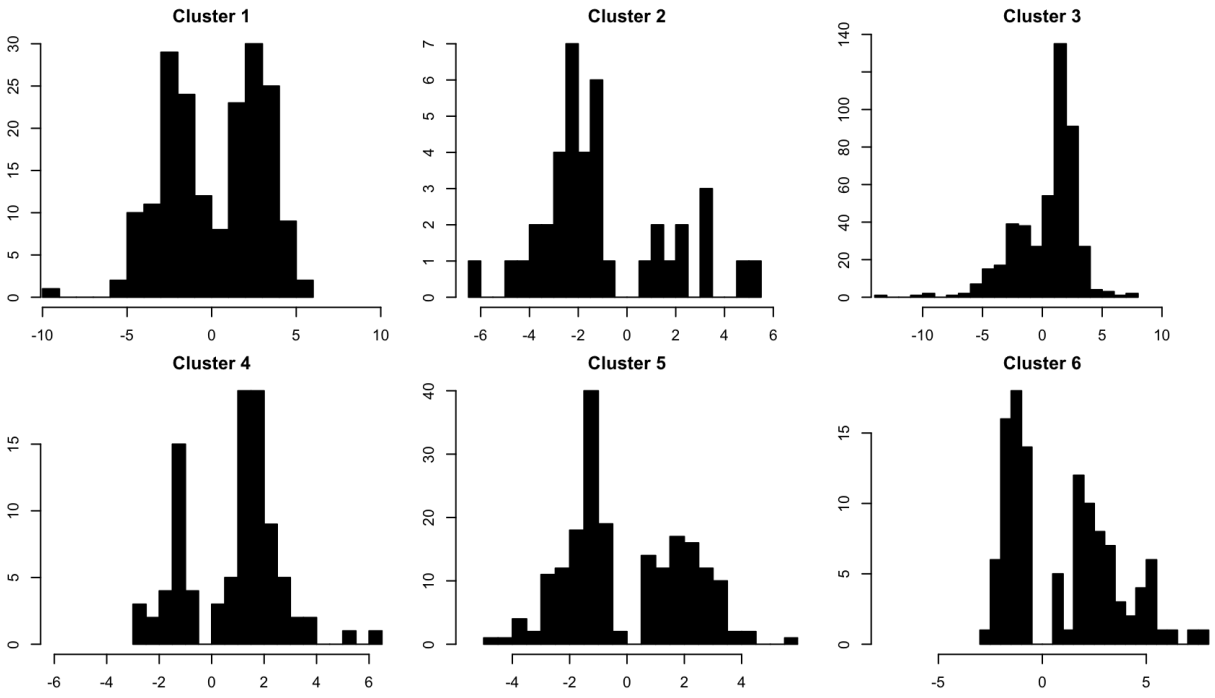


Figure 3.10: Cluster-specific histograms of the gene-specific scaling parameter estimated for each of the genes assigned to the cluster in the microbial data.

flexibility of the method, we here apply the mixture model to the log counts with the Normal distribution. The mRNA dataset is much larger than the microbial data: the original data contained 34,211 total genes, and after filtering by expression and variance, we used 22,388 genes in the clustering model. We are considering the data only for the RTx430 genotype under the pre-flowering drought and control conditions, across weeks 3 through 17, giving a total of 74 samples.

When using the Normal model, it is important to provide normalized data to the mixture model method, so we first applied to our data the normalization strategy implemented in the `EDASeq` package from Risso et al. [2011], which performs global scaling between samples using the upper-quartile. We then ran the mixture model on this normalized data. The clustering results obtained by the mixture model are shown in Figure 3.11, where the twenty clusters have been ordered from largest to smallest. For each cluster, the cluster centroid has been plotted, with the red color indicating the centroid in the drought condition and the black indicating the control condition. We observe a variety of different temporal patterns with some notable differences between conditions.

As with the microbial data, it is useful to explore how well the cluster centroids represent the behavior of the genes assigned to each cluster. Again, we consider heatmaps of the smallest and largest clusters, shown in Figure 3.12, where the gene and sample annotations are

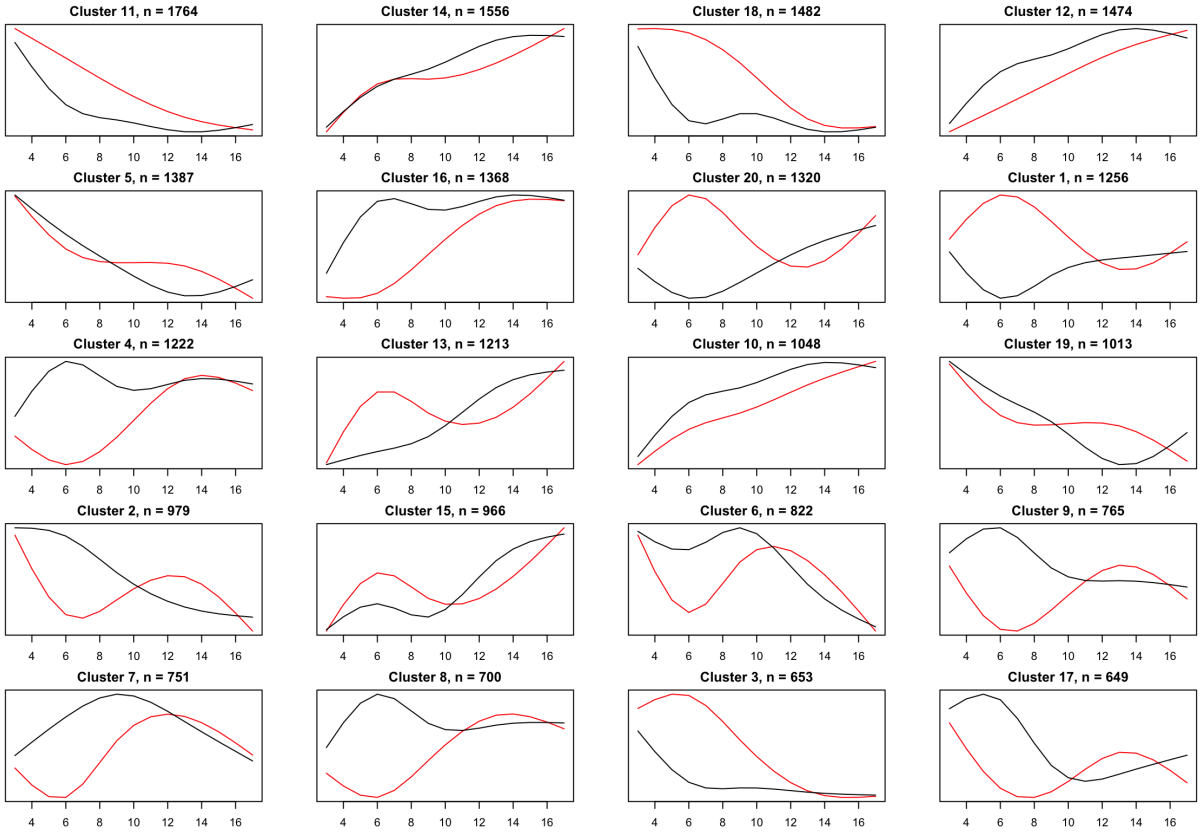
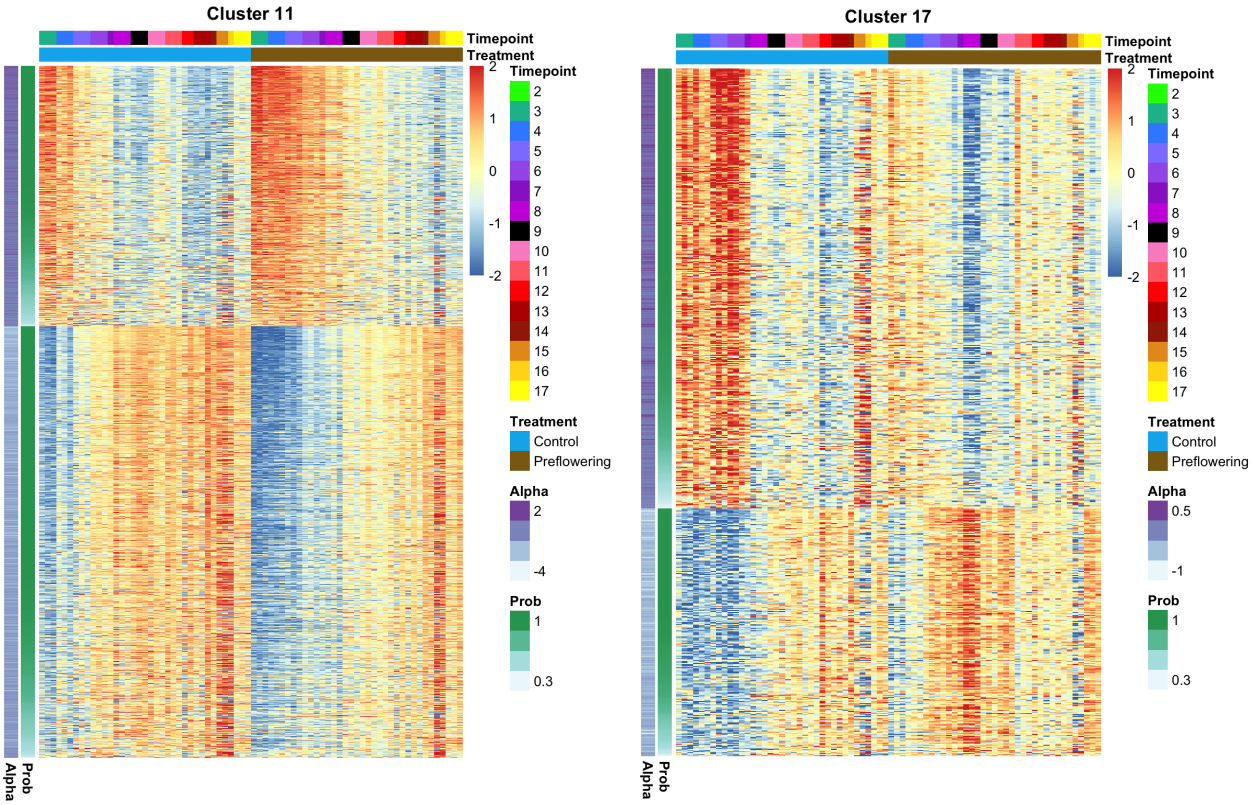


Figure 3.11: All of the twenty cluster centroids obtained from running the mixture model on the mRNA roots data, ordered from largest cluster size to smallest. The centroid is plotted in black for the control condition and in red for the pre-flowering drought condition.

the same as described in the heatmaps of the microbial data. Compared to the zero-inflated microbial data, it is much easier to see that the mRNA data within each cluster cleanly follow their assigned cluster centroid. In the largest cluster, cluster 11, we observe that all of the genes cleanly follow the pattern of the cluster centroid. Genes in the smallest cluster, on the other hand, have noisier data, although we can still observe that they reasonably follow their fitted cluster centroid. This indicates that the mixture model is producing reasonable clustering results, and that the cluster centroids are providing an accurate portrayal of the behavior of the clustered genes.

Both of these heatmaps reveal that their clusters contain two broad groups of genes with anti-correlated expression across time, identified by the a_g parameter, shown visually in the *Alpha* annotation track. To investigate the cluster-specific distributions of the scaling factors in the RNA data, consider the histograms in Figure 3.13. As in the microbial data, many of the clusters contain a bimodal distribution of positive and negative scaling factors. Again, this provides a convenient way to identify groups of anti-correlated genes that would



(a) Heatmap of genes in the largest cluster, cluster 11.

(b) Heatmap of genes in the smallest cluster, cluster 17.

Figure 3.12: Heatmaps of genes from clusters assigned by the mixture model, from the largest and smallest clusters.

have been missed by using traditional clustering methods. However, we also observe many clusters without a strong bimodal distribution, such as clusters 6, 13, 19, and 20. This indicates that the mixture model does not automatically force every cluster to contain both positively and negatively correlated genes, but that the anti-correlations present in the other clusters represent genuine anti-correlated behavior.

3.7.3 Downstream analysis

These examples serve to illustrate the application and results of running the mixture model on a typical genomics time-course experiment. In practice, these results would then serve as a launching point for further biological investigation. While not the focus of this dissertation, one area of further study that is filled with exciting potential is to conduct a gene set enrichment analysis on the resulting clusters obtained from the mixture model.

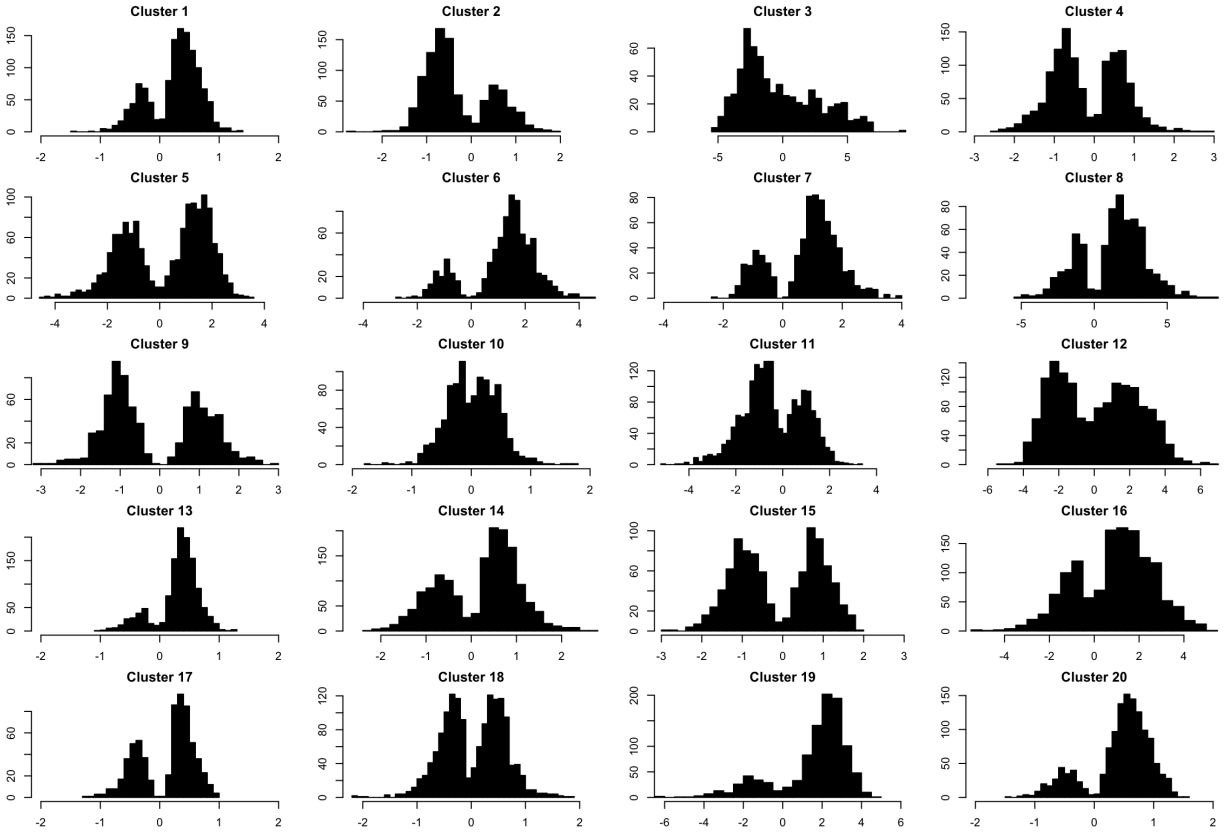


Figure 3.13: Cluster-specific histograms of the gene-specific scaling parameter estimated for each of the genes assigned to the cluster in the mRNA data.

A rich example of this kind of downstream analysis is in Varoquaux et al. [2019], which studied the transcriptional responses to drought. The authors applied a K-means clustering strategy using fitted splines and were able to draw many varied conclusions about the characteristics of genes assigned to the same cluster. After obtaining a set of clusters, the biological function of genes assigned to each cluster were investigated by performing gene-enrichment analysis to identify over-represented networks from Kyoto Encyclopedia of Genes and Genomes (KEGG) and Gene Ontology (GO) terms, and many interesting observations were made from the pathways identified by this analysis. In addition to these kinds of gene set enrichment analyses, our mixture model provides the unique potential for exciting discoveries through the use of the scaling parameter. Within a particular cluster, the scaling parameter can identify genes showing anti-correlated behavior, which could have potential relevance for a variety of biological hypotheses. Taken all together, our mixture model clustering method provides a wealth of tools for biologists to investigate statistically clustered groups of genes for biological relevance.

Chapter 4

Methods to test cluster means for differential expression

4.1 Background

A common goal in gene expression analysis is to conduct differential expression testing to determine which genes show significant changes, for example across time or between conditions. Although longer time-course settings typically identify too many genes as differentially expressed to be practically useful, the question can be adapted to identify which patterns in the data, represented by the cluster means, show differential expression in terms of a relevant hypothesis. The problem in this context is to test each cluster mean, rather than each gene, for differential expression.

The differential expression test of interest is to detect differences in the shape of temporal patterns between conditions. It may be obvious from the visual representation of cluster centroids that some clusters have very different patterns, as in Clusters 1 and 20 from the RNA clustering (shown in Figure 3.11). However, the assessment of differential expression may be much less clear in other clusters. In the RNA data, Clusters 11 and 14, for instance, show slightly different patterns of temporal change between conditions, but it is unclear if this is a significant change. Thus, we would like to apply a method to identify which clusters show statistically significant differences between conditions. Since our cluster centroids have been obtained from a mixture model clustering, we would like to incorporate in our differential expression test the variability of genes and their fit to the cluster. That is, we would like to consider all of the genes belonging to a cluster to assess the overall level of differential expression, as well as to use the probabilities of genes belonging to their assigned cluster in our assessment of whether the cluster as a whole is differentially expressed.

The majority of any work on the problem of inference on cluster means is in the context of Gaussian Mixture Models. Gaussian mixture models are a natural place to start to answer this question due to their ease of use and convenient likelihood properties. Typically, research has approached the problem of parameter inference in mixture models by seeking

confidence intervals around parameter estimates. While this is useful to estimate standard errors around model parameters, confidence intervals are not directly designed to answer our problem of detecting differential expression. However, confidence intervals can be a helpful starting point to build upon to more directly address the question of hypothesis testing.

Perhaps the simplest method to produce confidence intervals around cluster means in the Gaussian setting is to use the covariance matrix of the maximum likelihood estimates. For Gaussian models, the covariance matrix of the maximum likelihood estimate of a parameter, such as each cluster mean, is approximated by the inverse of the information matrix,

$$\mathrm{Cov}(\hat{\theta}) \approx I^{-1}(\hat{\theta}) := \left[-\mathbb{E}\left(\frac{\partial^2}{\partial \theta^2} \log f(X;\theta) | \theta \right) \right]^{-1}.$$

This covariance matrix can be used to construct confidence intervals around parameter estimates. As a simple extension to hypothesis tests, this covariance matrix can also be used to construct a test statistic such as the Wald statistic,

$$W = \frac{(\hat{\theta} - \theta_0)^2}{\mathrm{var}(\hat{\theta})},$$

which under the null hypothesis follows an asymptotic χ^2-distribution with one degree of freedom, providing a simple way to conduct a test of significance for each cluster mean. However, as noted in Basford et al. [1997], the assumption of asymptotic normality of the maximum likelihood estimates requires a large sample size before the assumption can be applied; furthermore, the resulting standard errors are very unstable unless the sample size is large or the means are well-separated.

Another approach is to use likelihood profiles, as presented in Visser et al. [2000], where this method was used to construct confidence intervals for hidden Markov model parameters. The idea of a likelihood profile is to search for a value of the parameter of interest that results in a significant change in the log-likelihood, treating other parameters as nuisance parameters. For example, suppose that the mean parameter θ is of interest. Then, all other parameters Ψ in the model are treated as nuisance parameters, and the likelihood profile function is defined as

$$L_p(\theta) = \max_{\Psi} L(\theta, \Psi).$$

Let the maximum value of this function be θ_M; then, search the space of θ by moving θ away from its max to some θ_0, and compute

$$R_p = -2 \ln \frac{L_p(\theta_M)}{L_p(\theta_0)}.$$

Continue searching until θ_0 is found that produces $R_p = 3.841$, the threshold of the χ^2-distribution with one degree of freedom at the 0.05 quantile. This value represents one side of the confidence interval, or the limit where the null model becomes significant over

the optimal parameter. The process is repeated to find the other side of the confidence interval. In this way, likelihood profiles can be used to generate confidence intervals for any parameter of interest. However, as with the covariance matrix approach, this method also relies on the asymptotic normality of the estimates. In addition, with complex models such as a mixture model with multiple gene-specific parameters and cluster membership probabilities, the likelihood profile optimization quickly becomes a computational challenge. Thus, a more general approach is needed.

Since the assumption of normality is a major hindrance, a great deal of interest has focused on using the nonparametric bootstrap to produce confidence intervals around mixture model parameters. For instance, the popular `mclust` package provides the MclustBootstrap function described in Scrucca et al. [2016], which applies the nonparametric bootstrap to Gaussian mixture model parameters. The idea of the bootstrap is to construct an empirical distribution representing the true population that generated the original data. This empirical bootstrap distribution is generated by resampling the data observations to create many new bootstrap datasets and then re-estimating the model parameter of interest on each new dataset. To be specific, the data observations are sampled with replacement to create a new dataset of the same dimension as the original, and the Gaussian Mixture Model is fit to each bootstrapped dataset to obtain a bootstrap distribution of the parameter of interest. This allows for a bootstrap estimate of the covariance matrix to be generated, which can then be used to produce confidence intervals. Thus, the bootstrap method provides a straightforward way to construct an estimate of standard errors without relying on any distributional assumptions. However, the bootstrap method is not free from difficulties. A thorough investigation of the use of the bootstrap method to obtain standard error estimates is covered in O'Hagan et al. [2019], where the authors point out that small clusters are not well represented by the traditional bootstrap method, and two alternatives using a jackknife and weighted likelihood bootstrap are presented to attempt to remedy this issue. Another evaluation of the bootstrap is given by Jaki et al. [2018], which addresses the concern that influential individual observations have a high likelihood of being over-represented in the bootstrap resamples. Similarly, Taushanov and Berchtold [2019] addresses the under-representation of small clusters by suggesting two modifications of the bootstrap: a separate bootstrap, which independently bootstraps the data within each cluster, and a stratified bootstrap, which draws proportional samples with respect to the original clustering results. These concerns are important to take into consideration, but a bootstrap approach seems to be the most promising strategy to conduct inference on our mixture model parameters.

A bootstrap method with some modifications appears to have the greatest ability to accurately provide confidence intervals around parameter estimates because it can conveniently avoid any distributional assumptions. However, constructing confidence intervals yet does not quite directly address our original question of differential expression testing. In this chapter, we will focus specifically on this question of differential expression testing from a hypothesis testing perspective.

4.2 Methods

The general strategy for all methods that follow is to use the clustering results given by the mixture model to conduct a test of differential expression for each cluster, providing a user with a statistical assessment of which clusters are most relevant. In this section, we will explore five possible methods that can be used to test for differential expression. Combined with the clustering method, these tests of differential expression allows us to integrate both problems of clustering and differential expression testing into one framework.

4.2.1 Likelihood ratio test

A standard approach to testing differential expression on a per-gene basis is to use a likelihood ratio test, and this method has had great success in the context of testing individual genes. Thus, a natural approach is to consider the same procedure applied to the cluster means.

For this method, once the clustering has been completed, the cluster labels are treated as known. That is, each cluster is treated as its own independent population to test each cluster separately for differential expression. This is a strong assumption to make, as this ignores the fact that these clusters are a result of a clustering algorithm. Hence, this test for differential expression does not incorporate any of the variability that comes from the clustering algorithm. Despite this strong assumption, however, the likelihood ratio test is a simple and convenient approach that merits consideration.

To demonstrate the likelihood ratio method, assume the cluster labels are known, and consider each cluster separately. For cluster k, of size n_k, the clustering results provide the fitted mean under the original hypothesis μ^k and gene-specific scale and shift parameters $\{(\alpha_i, w_i)\}, i = 1, ..., n_k$.

To test a given null hypothesis, the data in cluster k are fit under the null hypothesis to obtain a null mean μ_0^k and gene-specific null parameters $\{(\alpha_{0i}, w_{0i})\}, i = 1, ..., n_k$. The likelihood ratio statistic then computes the difference between the likelihood under the full model and the likelihood under the null model. Specifically, for cluster k,

$$LR_k = 2 * \sum_{i=1}^{n_k} \mathcal{L}(y_i|\mu^k, \{(\alpha_i, w_i)\}) - \mathcal{L}(y_i|\mu_0^k, \{(\alpha_{0i}, w_{0i})\}).$$

By Wilks' Theorem, this statistic converges to a Chi-square distribution with p degrees of freedom, where p is the difference in degrees of freedom between the two models. From the Chi-square distribution, a p-value can be obtained to indicate the significance of differential expression for each cluster. In the context of the full cluster with n_k genes, it is unclear what the correct degrees of freedom should be, and the more appropriate degrees of freedom for the test statistic might be $n_k * p$, since each cluster contains multiple genes sharing the same model.

In simulations, the likelihood ratio test did not perform well. While the method had satisfactory power, it produced a very high false positive rate, and distribution of the null

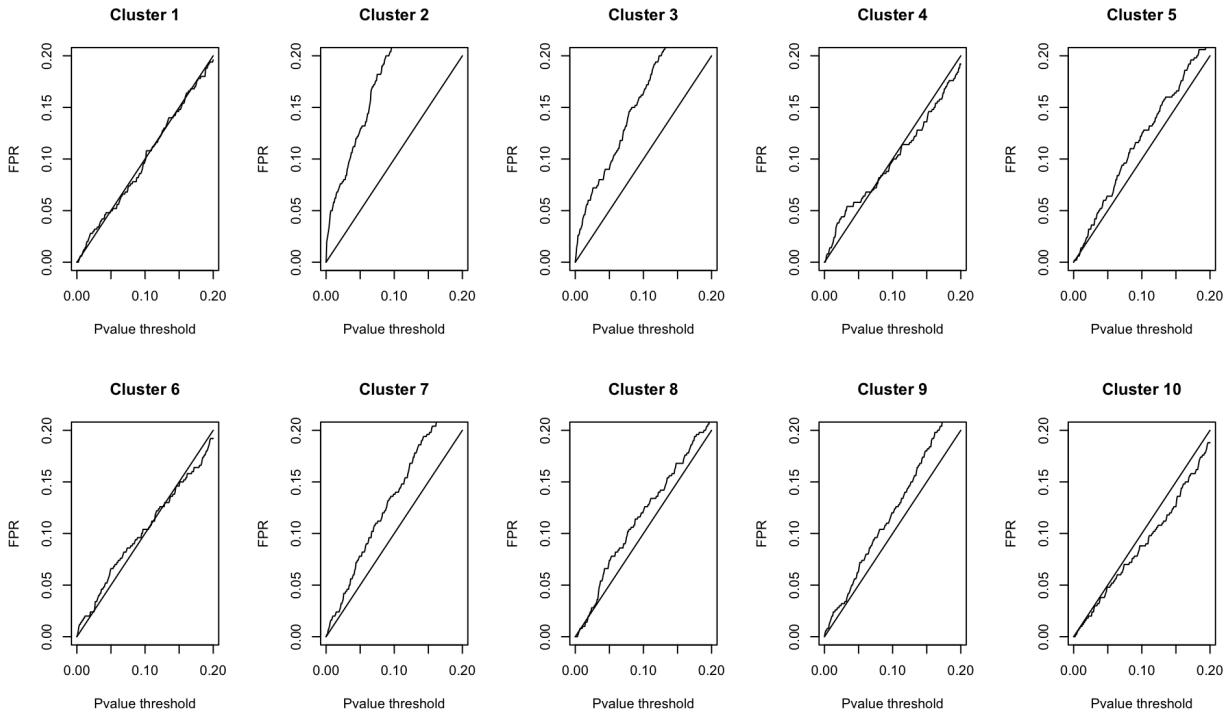


Figure 4.1: Per-cluster false positive rates of the Likelihood Ratio Test of differential expression conducted on data from a null setting under the normal model.

p-values was strongly non-uniform. Figure 4.1 shows the high false positive rates per cluster obtained from running the normal mixture model on data under the null setting, with no differential expression between conditions. The false positive rates vary significantly between clusters. Some clusters perform well, but many clusters, particularly Clusters 2 and 3 show an excessive amount of false positives. This poor performance is likely due to many contributing factors including many violations of assumptions. Thus, to avoid the problem of distributional assumptions, we turn to nonparametric methods.

4.2.2 Separate bootstrap

A simple approach to inference on cluster parameters from a mixture model is to apply the bootstrap to each cluster separately, as described in Taushanov and Berchtold [2019]. Here, the idea is to bootstrap the data within each cluster, again treating the cluster labels as known. To describe this method in the context of a Gaussian Mixture Model, the authors proposed resampling the observations. In the context of genomics data, the observations can be considered as the samples or as the genes. Resampling the genes in a given cluster most closely matches the approach of Taushanov and Berchtold [2019], but this is not a sensible

approach, since bootstrapping the genes would in essence be attempting to model some infinite population of genes. As this is not a realistic or desirable scenario, bootstrapping the samples within each cluster seems to be more appropriate.

The bootstrap is designed to generate an empirical distribution of a model parameter of interest, which in this case is the cluster mean. This empirical distribution mimics the true population distribution of the model parameter and provides an avenue to compute standard errors and confidence intervals around the model parameter of interest. The confidence intervals around the cluster means will be around each time point of the cluster mean per condition. These confidence intervals can then be used to conduct an overall test of differential expression.

To describe the separate bootstrap method, once again the strong assumption is made that once the clustering has been completed, the cluster labels are known. The method then considers each cluster k, with fitted mean μ_k, as its own independence population. Within each cluster k, the separate bootstrap method performs the following steps for each of B iterations. First, a new bootstrap dataset is created by sampling with replacement the cluster data's samples. Then, using this new, resampled dataset, a bootstrap cluster mean μ_k^b is fit to the data. The results of the B bootstrap iterations provide B cluster means μ_k^b that can be used to generate standard errors and confidence intervals around the original cluster mean μ_k at each time point This produces as many confidence intervals as we have time points in each condition. This is useful to obtain an idea of the standard errors around the cluster mean. However, to answer the original question of differential expression, one more step is needed.

Under a setting with only one condition, the null hypothesis of no differential expression at any time point can be tested with the hypothesis

$$H_0 : \mu_k(t) = 0, t = 1, ...T.$$

In a setting with two conditions A and B, the null hypothesis of no differential expression *between conditions* at any time point can be tested with the hypothesis

$$H_0 : \mu_k^A(t) = \mu_k^B(t), t = 1, ..., T.$$

This hypothesis can be written more clearly in terms of the differences between conditions by letting $d_t = \mu_k^A(t) - \mu_k^B(t)$, so that the null hypothesis is

$$H_0 : d_k(t) = 0, t = 1, ..., T.$$

To test these hypotheses using the output from the bootstrap results, consider the confidence intervals constructed around each time point For each time point j, a confidence interval C_j constructed at significance level α will have $P(d(j) \notin C_j) \leq \alpha$. In order to get an overall assessment of significance across all time points, these confidence intervals can be combined to test the hypothesis of no difference at any time point That is, the hypothesis of interest is

$$H_0 : d_k(1) = \cdots = d_k(T) = 0.$$

To make this a statistically valid hypothesis test, we simply need to control the family-wise error rate, which can be done by constructing each time point's confidence interval C_j at significance level $1 - \alpha/T$. This then ensures that

$$P(d(j) \notin C_j \text{ for some } j) \leq \sum_j P(d(j) \notin C_j) \leq \sum_j \frac{\alpha}{T} = \alpha,$$

as desired.

Similar to the likelihood ratio test, the separate bootstrap method is simple and relatively quick. In simulations, the separate bootstrap performs quickly and successfully, with power to detect truly differentially expressed clusters while also demonstrating a low false positive rate. However, this method is again not a completely satisfactory approach, because the separate bootstrap also treats each cluster as its own separate population, ignoring the variability due to clustering.

4.2.3 Full bootstrap

The bootstrap approach can be used more successfully by bootstrapping the full dataset and re-running the clustering algorithm on each bootstrapped data set. While this is a more computationally intensive approach, this method provides a distribution of the mean in each cluster that now incorporates the variability that comes from the clustering. As in the previous methods, various hypotheses can be tested per cluster.

Full bootstrap method

The process of the full bootstrap method is as follows.

1. Create a new bootstrap dataset by sampling with replacement the original dataset's samples.

2. Apply the mixture model clustering algorithm to the bootstrap dataset.

3. For each cluster k, compute the bootstrap statistic of interest. For the setting with one condition, this is simply the cluster mean

   $$\mu_k^B(t), t = 1, ..., T.$$

   For the setting with two conditions, this is the difference between conditions per time point,

   $$d_k(t) = \mu_k^A(t) - \mu_k^B(t), t = 1, ..., T.$$

4. Repeat steps 1-3 B times to obtain a distribution of μ_k or d_k for each cluster k.

Once the bootstrap distribution has been generated for each cluster, the process of testing differential expression is identical to the process in the separate bootstrap. Within each cluster, confidence intervals are constructed at level $1-\alpha/T$ around either the cluster mean $\mu_k(t)$ or difference between conditions $d_k(t)$, and the null hypothesis of no differential expression at any time point is tested as before.

In simulations, the full bootstrap performed very well, albeit with a high computational cost. Tests of differential expression were conducted using the confidence interval approach, and simulations achieved good power and low false positive rates.

Bootstrap label-switching problem

Since the full bootstrap method relies on re-fitting the mixture model to each bootstrap dataset, we are faced with a massive label-switching problem. Specifically, in step 2 of each bootstrap iteration, the clustering algorithm produces a set of clusters that have similar patterns to the original cluster results; however, although the clusters found by the algorithm in each of the B bootstrap clustering results are similar in pattern, the cluster labels of each bootstrap result are a random permutation of the original labels.

In order to obtain a distribution of the bootstrap statistics per cluster, the cluster labels therefore must be aligned across bootstrap iterations. Multiple approaches have been suggested to remedy the label-switching problem. The simplest approach is to impose identifiability constraints on the parameters, such as by ordering the cluster means smallest to largest; however, this approach only works in the simplest settings, and there is not obvious identifiability constraint that would apply to the spline curves of the cluster means in this setting. Various re-labeling algorithms have been proposed in Stephens [2000] and Celeux et al. [2000], but this would add an additional layer of computational complexity to an already computationally intensive bootstrap method. A more favorable approach discussed in McLachlan and Peel [2004] is to set the Expectation-Maximization algorithm's starting values to the posterior values achieved by the original clustering. Although this strategy sacrifices the random initialization of each bootstrap clustering, this is generally a successful solution to the label-switching problem, since the EM algorithm is known to be very dependent on starting values.

In our implementation of the full bootstrap, we addressed the label-switching problem by using a fixed initialization point. Specifically, each bootstrap clustering was initialized using the matrix of gene-cluster probabilities obtained from the original clustering result's posterior matrix. In practice, this method did not perform with perfect accuracy and some cluster labels ended up being misaligned, but in general, the cluster labels were successfully relabeled to the same order as the original clustering order.

Hypothesis testing with the bootstrap

The bootstrap method, therefore, is a very useful method to construct confidence intervals around model parameters, and these confidence intervals can be used to test a limited set of

hypotheses of differential expression. However, the confidence interval approach is limited in that it lacks the flexibility to test more specific hypotheses, such as testing for only a significant interaction of time point and condition, rather than any condition effect. To conduct a more traditional hypothesis test using the bootstrap, several considerations need to be made.

A hypothesis test requires a test statistic that can measure the amount of differential expression in each cluster. In the setting of differences across time, the simplest statistic is simply to take the sum of the absolute value differences between conditions at each time point,

$$\theta_k(F) = \sum_{t=1}^{T} |d_k(t)|.$$

A slightly more sophisticated version of this approach is the statistic

$$\theta_k(F) = \int_{t=1}^{T} \left| f_k^A(t) - f_k^B(t) \right| dt,$$

where f_k^A and f_k^B are the fitted functional curves across time within each condition. This statistic measures the area between the two curves, where a greater area between the two curves indicates greater differential expression between the two conditions.

Under the null hypothesis, $\theta_k(F) = 0$, since in the null setting the functional curves between conditions are identical. The original clustering produces the observed test statistics $\theta_k(\hat{F})$. The bootstrap samples produce an empirical distribution of the observed statistics created from the statistics $\theta_k(\hat{F}^*)$ produced by each bootstrap sample. From this distribution, it might seem natural to construct a confidence interval or conduct a hypothesis test to test for differential expression. However, neither of these approaches are valid in this setting.

First, to illuminate the problem with constructing confidence intervals around our proposed statistics θ_k, a quick observation of the definition of the statistic θ_k will immediately reveal that this statistic is strictly positive. Therefore, all of the bootstrap statistics will always be greater than zero. To test for differential expression using a confidence interval around this statistic, a hypothesis test would need to test that the confidence interval around the observed statistic covers zero. However, it is impossible for this to happen with a strictly positive bootstrap statistic, since none of the bootstrap statistics can possibly be less than zero. Hence, such a test of differential expression would have perfect power, but no control over the false positive rate.

Secondly, to consider hypothesis testing, there are again difficulties in comparing our observed test statistics $\hat{\theta}_k(F)$ to the bootstrap distribution. The problem with hypothesis testing using the bootstrap is that by its construction, the empirical distribution is close to the true population distribution, which may or may not follow the null distribution. To conduct a hypothesis test, a distribution satisfying the null hypothesis is required with which

to compare our test statistic. It might be tempting to calculate a bootstrap p-value as

$$p^*(\hat{\theta}) = \frac{1}{B}\sum_{b=1}^{B} I(\theta_b^* > \hat{\theta}).$$

However, this p-value is useless as a hypothesis test because it lacks any power. By construction of the empirical distribution, this p-value will reject the null hypothesis with probability α regardless of how far the observed population distribution is from a true null distribution.

Therefore, to do a valid hypothesis test, an accurate null distribution must be estimated. The remaining two methods provide approaches that can construct such an empirical null distribution.

4.2.4 Permutation test

One approach to generating a null distribution is to use a permutation approach. This method generates a null distribution by shuffling the condition labels of each replicate. The process is similar to the bootstrap, except that for each iteration, instead of resampling the dataset's samples, the condition labels of all the samples are permuted. Generating a large number of datasets with permuted condition labels mimics a set of datasets coming from the null distribution that has no condition effect.

However, in practice, immediately the major hindrance of this permutation method appears, which is that in settings with few replicates as in most genomics studies, there are a very limited number of possible permutations of condition labels. For example, with a study of two conditions with three replicates, there are only $\binom{6}{3} = 20$ total possible permutations. This severely limits the power of any hypothesis test. As a result, the permutation test is unlikely to be useful in a typical genomics setting.

4.2.5 Residual bootstrap

A more successful approach to generating a null distribution is to use the residual bootstrap method. To apply this method, a test statistic is needed to measure of the amount of differential expression in each cluster. For cluster k, consider the test statistic proposed earlier,

$$\theta_k(F) = \int_{t=1}^{T} \left| f_k^A(t) - f_k^B(t) \right| dt,$$

where f_k^A and f_k^B are the fitted functional curves across time within each condition. This statistic measures the area between the two curves, where a greater area between the two curves indicates greater differential expression between the two conditions.

Under the null hypothesis, $\theta_k(F) = 0$ for all clusters, since the functional curves under each condition will be identical. From the mixture model clustering of a dataset, the observed statistics $\theta_k(\hat{F})$ can be computed for each cluster. To test the significance of these observed

test statistics, a residual bootstrap method can be applied. This method constructs an empirical null distribution with which to compare the observed test statistics.

Unlike the traditional bootstrap, which generates an empirical distribution of the true population distribution, the residual bootstrap attempts to generate an empirical *null* distribution. The basic idea is to generate new null datasets by randomly creating new realizations of each gene according to the null hypothesis. This is done by resampling the observed residuals and adding them onto a null model fit. This simulates a population with no differential expression. The observed residuals are used in the resampling step because these residuals are an approximation of the true distribution of residuals regardless of whether the true population follows the null hypothesis or alternative hypothesis.

Residuals for count distributions

In the normal model, residuals conveniently have *i.i.d.* behavior, making them perfectly suited for a residual bootstrap resampling method. With non-normal distributions, however, residuals no longer have such nice behavior, so it is necessary to consider modifications to the basic residual. There are many ways to define residuals in the context of non-normal distributions, including standardized residuals, Pearson residuals, and deviance residuals.

Deviance residuals, defined as

$$\hat{\epsilon}_{ij}^{D} = \text{Sign}(y_{ij} - \hat{y}_{ij}) * \sqrt{2(y_{ij} \log(y_{ij}/\hat{y}_{ij}) - y_{ij} + \hat{y}_{ij})},$$

are particularly appealing in the setting of count distributions, because the non-normal models are fit by deviance optimization, which has a nice correspondence with the RSS in the normal setting. Moreover, the deviance residuals more closely follow an *i.i.d.* distribution, making them a natural choice for a resampling procedure. For this reason, Hartl [2010] devised a method to bootstrap deviance residuals in the context of developmental triangles in stochastic reserving. However, as noted by the authors, the major difficulty with working with these deviance residuals is that there does not exist a closed-form expression for the functional inverse of the deviance residuals. Thus, the process of mapping resampled residuals to responses in order to generate bootstrap null data is a computational challenge. As a result, the authors of Hartl [2010] made several simplifying assumptions in order to map the residuals to responses for their bootstrap method. In a separate context, deviance residuals were also proposed for Poisson generalized linear models in Renshaw and Haberman [2008]; however, the strategy for mapping the deviance residuals to the responses is again a problematic computational hindrance. The authors also consider the Pearson residuals as a simpler alternative option.

Although the deviance residuals are perhaps the most natural choice to use in generating null datasets in the context of generalized linear models, Pearson residuals present a favorable alternative, since the Pearson residual is much easier to invert. For this reason, England and Verrall [1999] used Pearson residuals in a bootstrap method for a Poisson regression in the context of computing an IBNR-forecast in claims reserving. The Pearson residual is defined

as

$$\hat{\epsilon}_{ij}^{P} = \frac{y_{ij} - \hat{y}_{ij}}{\sqrt{V(\hat{y}_{ij})}},$$

where $V(\hat{y}_{ij})$ is the variance of y. For a Poisson model as in England and Verrall [1999], the variance is simply $V(\hat{y}_{ij}) = \hat{y}_{ij}$. The negative binomial model, on the other hand, requires an additional term to account for the over-dispersion and is defined as

$$V(\hat{y}_{ij}) = \hat{y}_{ij} + d(\hat{y}_{ij}^2)$$

for dispersion parameter d. Generating a new null dataset with the Pearson residual is therefore a simple step of inversion, where the new data y^* is created by

$$y_{ij}^* = \hat{y}_{ij}^0 + \hat{\epsilon}_i^{P*}\sqrt{\hat{y}_{ij}^0 + d(\hat{y}_{ij}^0)^2},$$

where $\hat{\epsilon}_i^{P*}$ has been sampled with replacement from $\hat{\epsilon}_{ij}^P$ for fixed i. The Pearson residual is therefore a suitable option for the residual bootstrap. Moreover, since the Pearson residuals and deviance residuals are generally quite close in value, the computational simplicity of the Pearson residual makes them a highly preferable option for the residual bootstrap method.

For computational simplicity, therefore, the residual bootstrap method that follows uses the Pearson residuals for the NB and ZINB settings. The Normal setting relies on the basic residuals

$$\hat{\epsilon}_{ij} = y_{ij} - \hat{y}_{ij}^1.$$

Residual bootstrap method

The basic framework of the residual bootstrap method is similar to that in Leek et al. [2006], and is as follows.

1. Fit the mixture model to the full model, to obtain $\hat{y}^1$, and compute the test statistics of interest, $\hat{\theta}_k, k = 1, ..., K$.

2. Fit the mixture model to the null model, to obtain $\hat{y}^0$. The desired null model can be one of two options, depending on the question of interest. One option is a null model that includes a condition shift effect, meaning that the residual bootstrap will test only for significant difference between conditions in the *shapes* of the centroids. The alternative option is a model with no condition effect at all, which tests for significant differences in either a shift or the shape between conditions.

3. Use the full model fitted values $\hat{y}^1$ to compute residuals $\hat{\epsilon}_{ij}$. For the normal model, the residuals are computed as
$$\hat{\epsilon}_{ij} = y_{ij} - \hat{y}_{ij}^1.$$
If the data distribution is either the NB or ZINB, the Pearson residuals are used, which are computed by
$$\hat{\epsilon}_{ij} = \frac{y_{ij} - \hat{y}_{ij}}{\sqrt{V(\hat{y}_{ij})}}.$$

4. Generate new null data y^* by resampling the residuals $\hat{\epsilon}_{ij}$. Once the appropriate residuals have been computed, the new null data is created for the Normal setting by

$$y^*_{ij} = \hat{y}^0_{ij} + \epsilon^*_i,$$

and for the NB and ZINB setting by

$$y^*_{ij} = \hat{y}^0_{ij} + \hat{\epsilon}^*_i \sqrt{\hat{y}^0_{ij} + d(\hat{y}^0_{ij})^2},$$

where ϵ^*_i is sampled with replacement among $\hat{\epsilon}_{ij}$ for fixed i. For the count distributions, the generated values are then rounded to integer counts with minimum count 0.

5. Fit the mixture model with y^* under the full model.

6. Repeat steps (3-4) B times to get $K*B$ bootstrap statistics θ^{*b}_k, for clusters $k = 1, ..., K$ and $b = 1, ..., B$.

7. Compute a p-value for cluster k by

$$p_k = \sum_{b=1}^{B} \frac{\#\{j : \theta^{*b}_j \geq \theta_k, j = 1, ..., K\}}{K \cdot B}.$$

Here, the p-value per cluster is the proportion of null clusters that have a statistic as extreme or more extreme than the observed cluster statistic. By computing the p-value this way, the observed statistic is compared to all possible null clusters.

A more conservative approach would be to only consider the null statistics for that cluster, i.e.,

$$p_k = \frac{\#\{b : \theta^{*b}_k \geq \theta_k, b = 1, ..., B\}}{B}.$$

However, this requires aligning the clusters from each bootstrap sample to the original clustering. This is not unreasonable in the case where the true data is similar to the null data. However, when the null population is very different from the true population, the clustering on the real data has no relationship to the clusterings on the bootstrap null data. Thus, there is no obvious way to align the cluster labels across bootstrap iterations.

The residual bootstrap provides a flexible approach to differential expression testing, with the advantage of being able to test any null hypothesis of interest. The method demonstrated high levels of power and low false positive rates based on simulated datasets in both a Normal and ZINB setting. In addition, the method was used successfully to identify appropriately differentially expressed clusters in a real data setting.

4.3 Evaluating Performance

Among the five methods proposed in the previous section, the residual bootstrap suggests the most promise. To evaluate the success of this method from a statistical perspective, the residual bootstrap was tested on a variety of simulated data sets. However, the process of consolidating the results across simulations presents a number of challenges resulting from the clustering context.

Briefly, the process of evaluating the performance of simulations is as follows. First, many simulated datasets are generated from a set of clusters that follow either the null hypothesis or the differentially expressed hypothesis. Next, the mixture model is fit to each simulated dataset, and the residual bootstrap method is applied to the clustering results. In order to assess the per-cluster performance of the tests of differential expression, the clusters are aligned across simulations to provide distributions of p-values for each cluster. Once these cluster p-value distributions are obtained, assessments of the power and false positive rate can be investigated within each cluster.

However, just as was observed in the full bootstrap method, each simulation ends up having its own permutation of cluster labels, meaning that the conclusions of the test of differential expression for each cluster will not align with the true original cluster labels. Therefore, the cluster labels must first be aligned across simulations.

4.3.1 Aligning clusters across simulations

In order to compare cluster mean differential expression detection performance across simulations, we need a way to align the clusters between simulations. Specifically, in the context of simulations, where the true cluster labels are known, we would like to align each of the assigned cluster labels to the true cluster labels.

To explore this problem, represent the true cluster labels as $\{k : 1, ..., K\}$ and the observed labels obtained from the clustering as $\{j : 1, ..., J\}$. Here, $K = J$, although this is not necessary in general.

We are seeking an alignment of the two clusterings. Consider the two mappings

$$\sigma : \{k : 1, ..., K\} \to \{j : 1, ..., J\}$$
$$\tau : \{j : 1, ..., J\} \to \{k : 1, ..., K\}$$

The goal is to understand the distribution of a test statistic evaluated on the estimated parameters obtained from a clustering of data. The process can be represented by

$$F_\theta \to X \to \hat{\mu} \xrightarrow{\sigma} \hat{\theta} \to t(\hat{\theta}),$$

or

$$F_\theta \to X \to \hat{\mu} \xrightarrow{\tau} \hat{\omega} \to t(\hat{\omega}),$$

where F_θ represents the underlying population from which data X has been collected, $\hat{\mu}$ represents the unaligned cluster parameters, and $\hat{\theta}$ and $\hat{\omega}$ represent the aligned cluster parameters, using the σ or τ alignments. That is,

$$\hat{\theta} = \begin{bmatrix} \hat{\mu}_{\sigma(1)} \\ \vdots \\ \hat{\mu}_{\sigma(K)} \end{bmatrix}, \text{ and } \hat{\omega} = \begin{bmatrix} \hat{\mu}_{\tau(1)} \\ \vdots \\ \hat{\mu}_{\tau(J)} \end{bmatrix}.$$

We are then interested in the distribution of the test statistic, such as the area between the curves for the residual bootstrap method, evaluated on the aligned parameter estimates.

Upon closer inspection, the two label mappings σ and τ have several interesting properties. First, note that the mappings σ and τ are defined by the same function f; in this case,

$$f(k) = \left\{ j : |\{z_i^k | z_i^k = j\}| \geq |\{z_i^k | z_i^k = l\}|, \forall\ l \in \{1, ..., J\} \right\},$$

where $z_i^k \in \{1, ..., J\}$ is the assigned label of data point i in cluster k. Next, σ and τ are random, non-invertible mappings. Furthermore, σ and τ are not unique mappings, i.e.,

$$|\sigma(1), ..., \sigma(K)| < K, \text{ and } |\tau(1), ..., \tau(J)| < J.$$

Finally, each true cluster maps to one observed cluster under σ, while each true cluster can align to multiple observed clusters under τ:

$$|\{\hat{\mu}_j : \hat{\mu}_j = \hat{\mu}_{\tau(1)}\}| \neq 1, \text{ but } |\{\hat{\mu}_j : \hat{\mu}_j = \hat{\mu}_{\sigma(1)}\}| = 1.$$

These properties present several challenges with cluster alignment. Importantly, the σ mapping loses the observed clusters that are small in size or that are noisy in the sense that they contain genes belonging to a large number of true clusters. This is because larger clusters will preferentially be mapped to the true clusters, since they contain a higher number of the true genes. The obvious consequence of this is that larger clusters will be over-represented in the cluster alignment. This suggests a potential for overly optimistic representation of per-cluster differential expression results, since noisier clusters that could hurt the success of the method are eliminated in this step.

Distribution of $t(\hat{\mu}_j)$

Setting aside these concerns for the present and returning to the original goal, we would like to generate a distribution of $t(\hat{\mu}_j)$ for each cluster j across many simulations. From simulations, it is not possible to do this precisely; instead, simulations can only provide distributions of

$$t(\hat{\mu}_{\tau(1)} | \tau(1) = 1)$$

or

$$t(\hat{\mu}_{\sigma(1)} | \sigma(1) = 1).$$

The full distribution of $t(\hat{\mu}_j)$ can be specified as the sum of two terms:

$$\begin{aligned}\mathcal{L}(t(\mu_j)) =& \mathcal{L}(t(\mu_j)|j \in \{\sigma(1), ..., \sigma(K)\})\mathcal{L}(j \in \{\sigma(1), ..., \sigma(K)\}) \\ &+ \mathcal{L}(t(\mu_j)|j \notin \{\sigma(1), ..., \sigma(K)\})\mathcal{L}(j \notin \{\sigma(1), ..., \sigma(K)\})\end{aligned}$$

The first term can be estimated by simulations by using the σ alignment. The second term is missing, as this term represents the observed clusters where the true cluster did not match to an observed cluster. If we assume a best-case scenario where there is a perfect clustering alignment, the second term is zero and able to be ignored.

To further decompose the first term, $\mathcal{L}(t(\mu_j)|j \in \{\sigma(1), ..., \sigma(K)\})$ can be estimated by the distributions of $t(\mu_j)$ obtained from simulations. However, $\mathcal{L}(j \in \{\sigma(1), ..., \sigma(K)\})$ is not possible to estimate, because the cluster labels j are different in each simulation, there is no way to model this probability.

In light of these difficulties, the most practical approach to align clusters to assess the performance of our methods is to use the σ mapping. However, this has the effect of ignoring small and noisy clusters found by various simulations.

An alternative to the process of aligning clusters is to consider the distribution of $t(\mu_j)$ across all clusters, without separating the distributions by cluster. This ensures that every observed cluster is represented, giving a more accurate picture of the overall performance of the test for differential expression.

4.3.2 Performance measurement

To measure the success of each method to test differential expression, simulations were run under two settings: a null setting, and a differentially expressed setting. Datasets were simulated as described previously for the clustering assessment, under both the normal distribution and the ZINB distribution. The null setting had 15 clusters with changes across time, but no difference between conditions. The differentially expressed setting also had 15 clusters with changes across time, but with a difference between conditions. Both settings had an additional cluster of random noise, to give a total of 16 clusters.

On each simulated dataset, the mixture model clustering was performed, clusters were re-labeled to align with the true cluster labels, and then one of the tests of differential expression conducted. False positive rates and power were tested in various ways depending on the method. As indicated in the previous discussions, the results for the likelihood ratio test were not promising, and the separate bootstrap, full bootstrap, and permutation test were eliminated from further assessment for methodological reasons. Therefore, for the sake of brevity, we will focus on results obtained to evaluate the performance of the residual bootstrap method.

4.4 Residual bootstrap performance results

To demonstrate the effectiveness of the residual bootstrap for detecting differentially expressed clusters, we applied the residual bootstrap method to simulated datasets and a real data set. In both cases, the results indicate that the model performs in both an accurate and meaningful way.

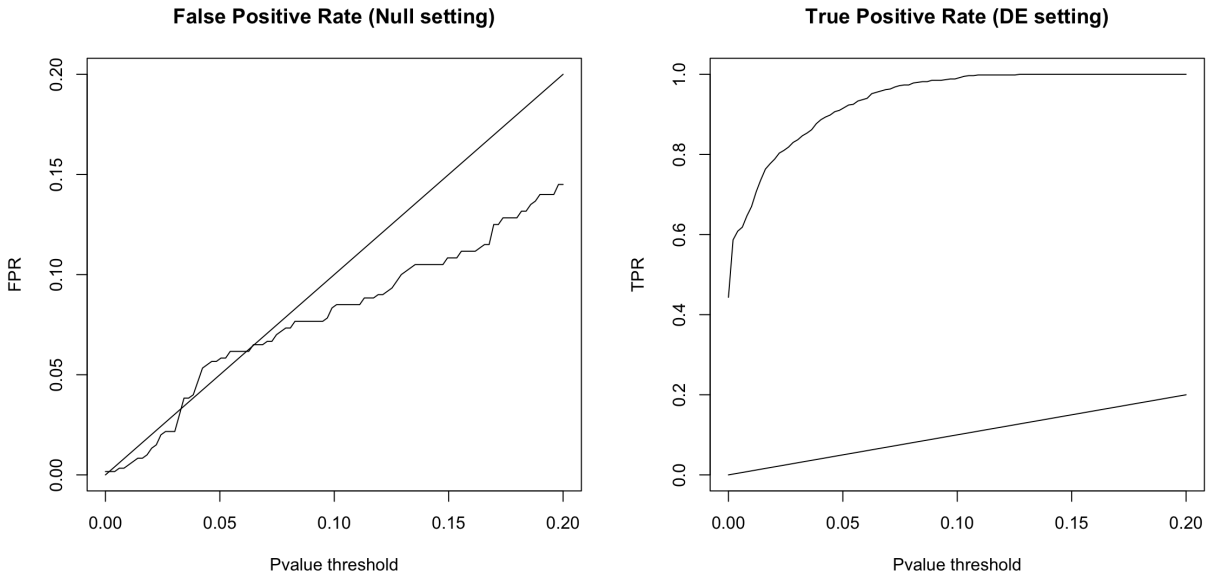
4.4.1 Simulation results

Simulations of the residual bootstrap method were run under both the Normal model and the ZINB model. Datasets were simulated as described previously according to the Normal and ZINB distributions. The mixture model was then fit to each of 20 simulated data sets, and the residual bootstrap method was applied to each set of results, using 100 bootstrap repetitions to generate the p-values. Under the null setting of no differential expression, all clusters were simulated to have no differential expression between conditions. Under the DE setting, all clusters were simulated to be differentially expressed between conditions. In each case, the residual bootstrap was testing the null hypothesis of no difference between conditions across time.

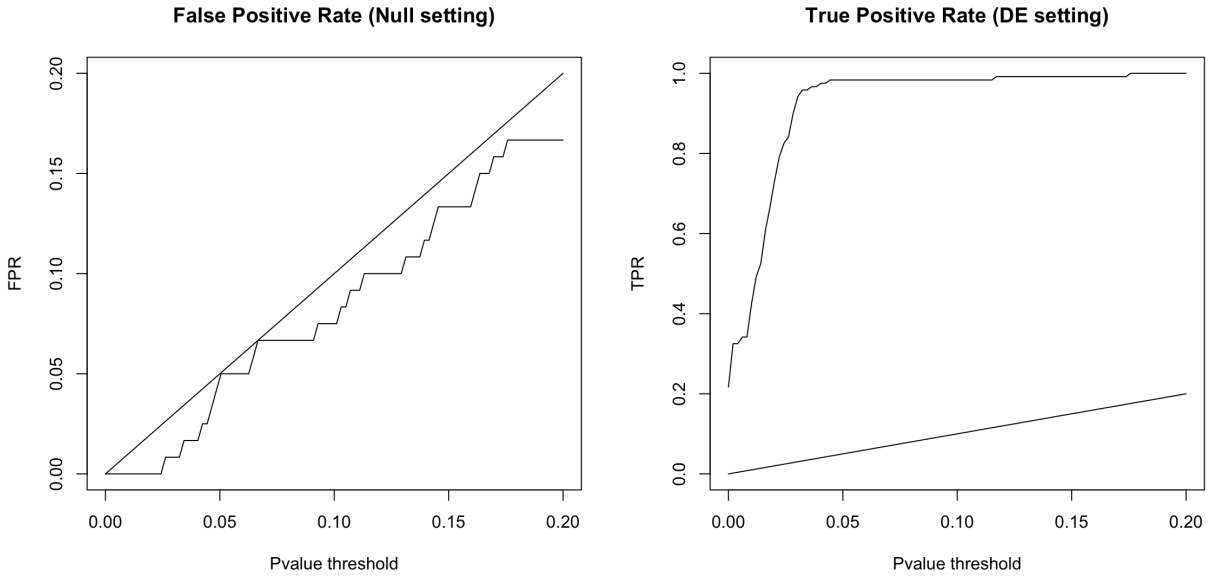
The proportions of false and true discoveries according to increasing values of significance levels are shown in Figure 4.2. These proportions were computed by the number of clusters that received a p-value less than or equal to the given cutoff along the x-axis. The diagonal $x = y$ line demonstrates the expected behavior under a null settings where the p-values should follow a uniform distribution. From these plots, we can observe that the residual bootstrap method performs very well in both the Normal and ZINB settings. The false positive rate is low or lower than expected for the null clusters, and the true positive rate is very high for the differentially expressed clusters. These plots indicate that the residual bootstrap is a statistically reasonable method to detect differential expression. Furthermore, from these successful simulations, we can conclude that the residual bootstrap method has the power to detect truly differentially expressed clusters without an excess of false discoveries.

4.4.2 Real data results

To explore the performance of the residual bootstrap on real data, we consider again the two EPICON datasets introduced and analyzed in Chapter 3. The 16S microbial dataset provides an interesting opportunity to apply the residual bootstrap to a count dataset, taking advantage of the Pearson residual option. To demonstrate the use of the residual bootstrap on Normal data, we also present the results of running the residual bootstrap on the log counts of the mRNA data.



(a) False positive rate and true positive rate of residual bootstrap on Normal data.



(b) False positive rate and true positive rate of residual bootstrap on ZINB data.

Figure 4.2: False positive rates and true positive rates for residual bootstrap, run on Normally distributed data using normal residuals, and ZINB distributed data using Pearson residuals.

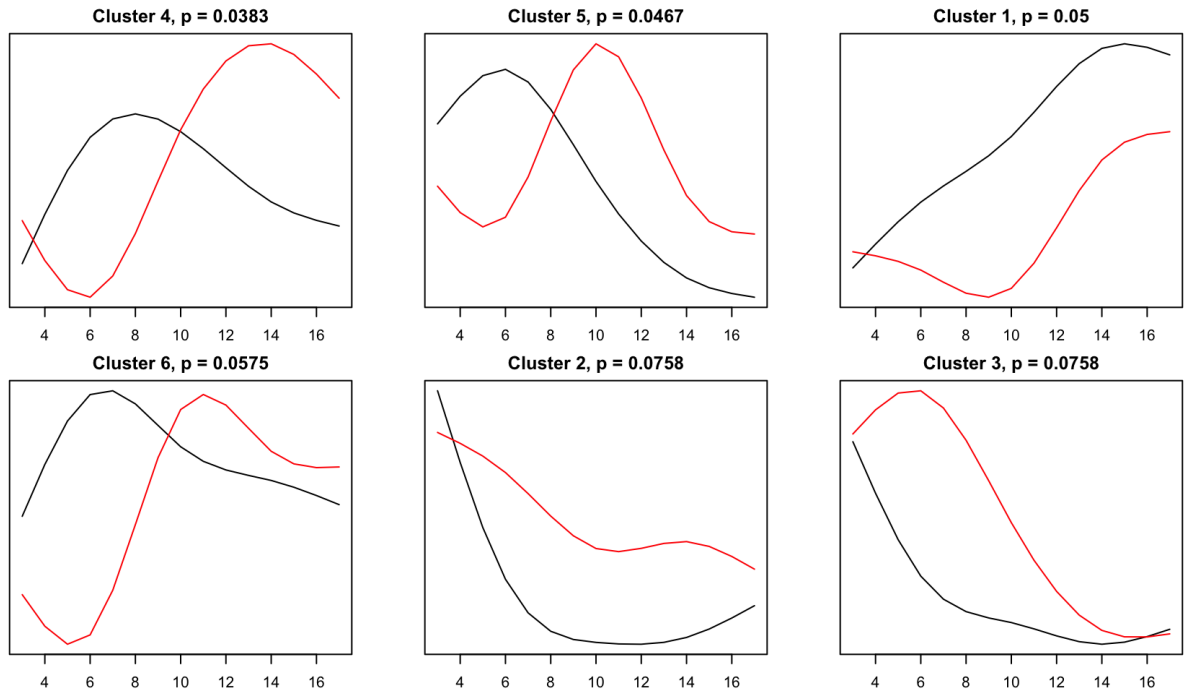
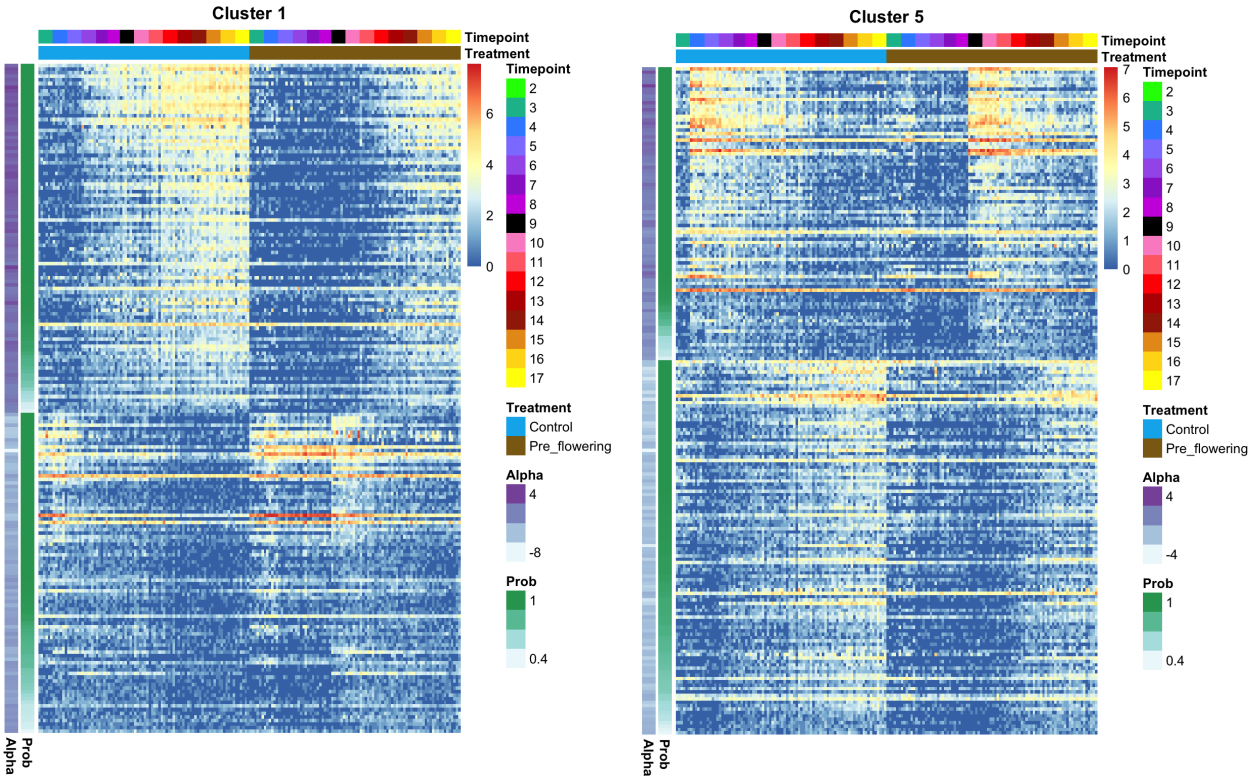


Figure 4.3: Cluster centroids obtained from running the mixture model on the 16S microbial data, ordered from smallest p-value to largest. The centroid is plotted in black for the control condition and in red for the pre-flowering drought condition.

16S microbial data

First, consider again the 16S microbial dataset. For each of the six clusters, we tested the hypothesis of a Condition effect in terms of an overall shift or shape difference between the two conditions. The residual bootstrap detected three of the six clusters as being differentially expressed. The results of the residual bootstrap can be explored visually by looking at the cluster centroids with their assigned p-values. The centroids are plotted in Figure 4.3 along with their p-value computed by the residual bootstrap. At a significance level of 0.05, the first three clusters shown in Figure 4.3 are declared significantly differentially expressed, while the remaining three are not. While these results may seem conservative, they do confirm our previous conclusion that many of the microbial OTUs display the same behavior under both conditions, but with a minor delay in progression.

Clusters 2 and 3 were explored in depth in Chapter 3, and their heatmaps from Figure 3.9 lend support to the conclusion that these clusters are not demonstrating significantly different behavior between conditions. To more closely examine clusters that have been declared differentially expressed between conditions, we consider heatmaps of the OTUs belonging to clusters 1 and 5, shown in Figure 4.4. The patterns of abundance shown by the OTUs in cluster 1 clearly show different temporal behavior between conditions: OTUs with a positive



(a) Heatmap of OTUs in cluster 1.

(b) Heatmap of OTUs in cluster 5.

Figure 4.4: Heatmaps of OTUs belonging to clusters detected to be differentially expressed between conditions according to the residual bootstrap test of their cluster means.

scaling factor display rapid increases in abundance starting at time point 6 under the control condition, while the increase in abundance under the drought condition is delayed until after time point 12. The opposite phenomenon can be seen in the OTUs with negative scaling factors, although this behavior is noisier. Likewise, the OTUs from cluster 5 display clearly different patterns of abundance between conditions, where a peak in abundance is observed at time point 4 under the control condition but does not appear until time point 9 under the drought condition. Each of these heatmaps support the overall conclusion of differential expression detected by the residual bootstrap.

mRNA gene expression data

The residual bootstrap method was also applied to the mRNA gene expression dataset coming from the EPICON project analyzed in Chapter 3. This dataset contains the root gene expression data from three replicate samples at each week for weeks 3 through 17, under the Control and Pre-flowering drought conditions, from the RTx430 genotype. The Normal

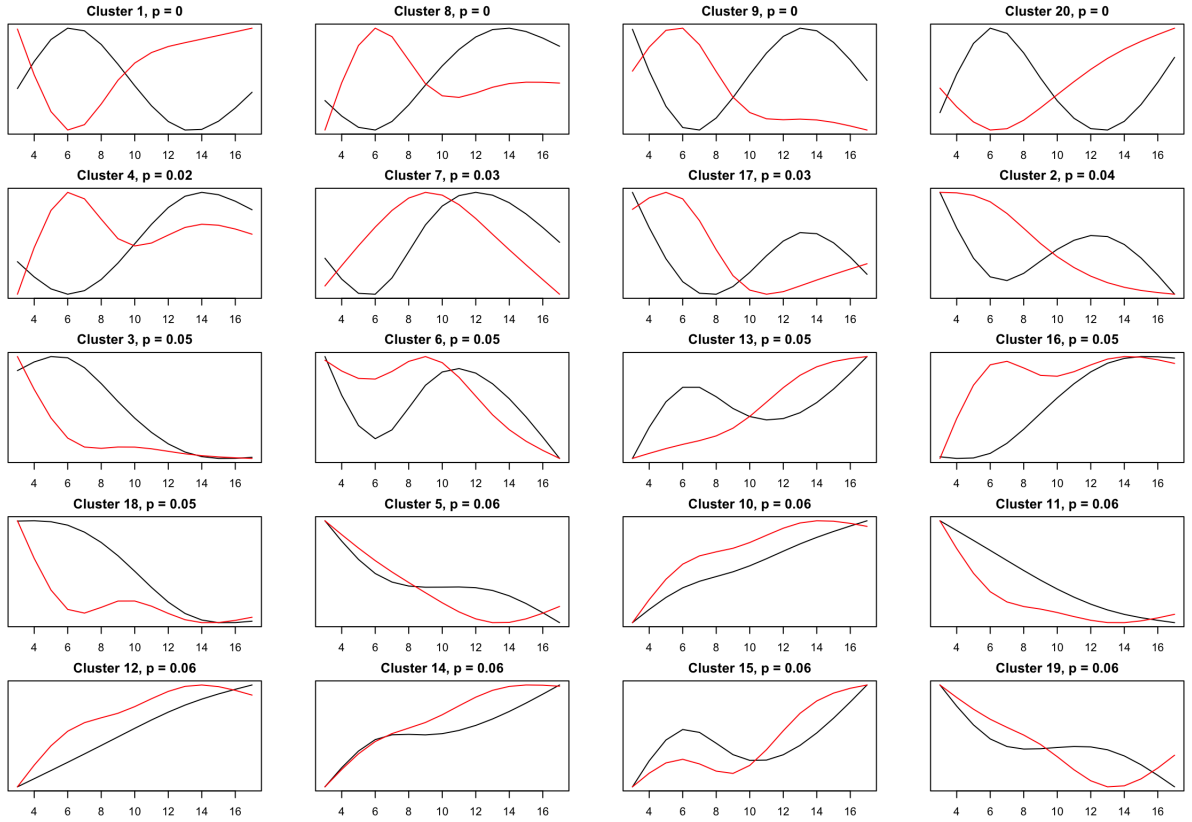


Figure 4.5: Residual bootstrap results of the Pre-flowering drought versus Control condition test of differential expression.

mixture model was applied to the log counts of the mRNA, and the residual bootstrap was then applied to test the cluster means. First, the difference between watering conditions was tested on this dataset from just the RTx430 genotype. In addition, for a further demonstration of the residual bootstrap method, we also consider separately the data coming from both the RTx430 and BTx642 genotypes under the control condition, where again the data has been clustered on the log counts with the Normal mixture model and the residual bootstrap method applied to detect differentially expressed clusters. In both cases, the mixture model was fit to the data for 20 clusters.

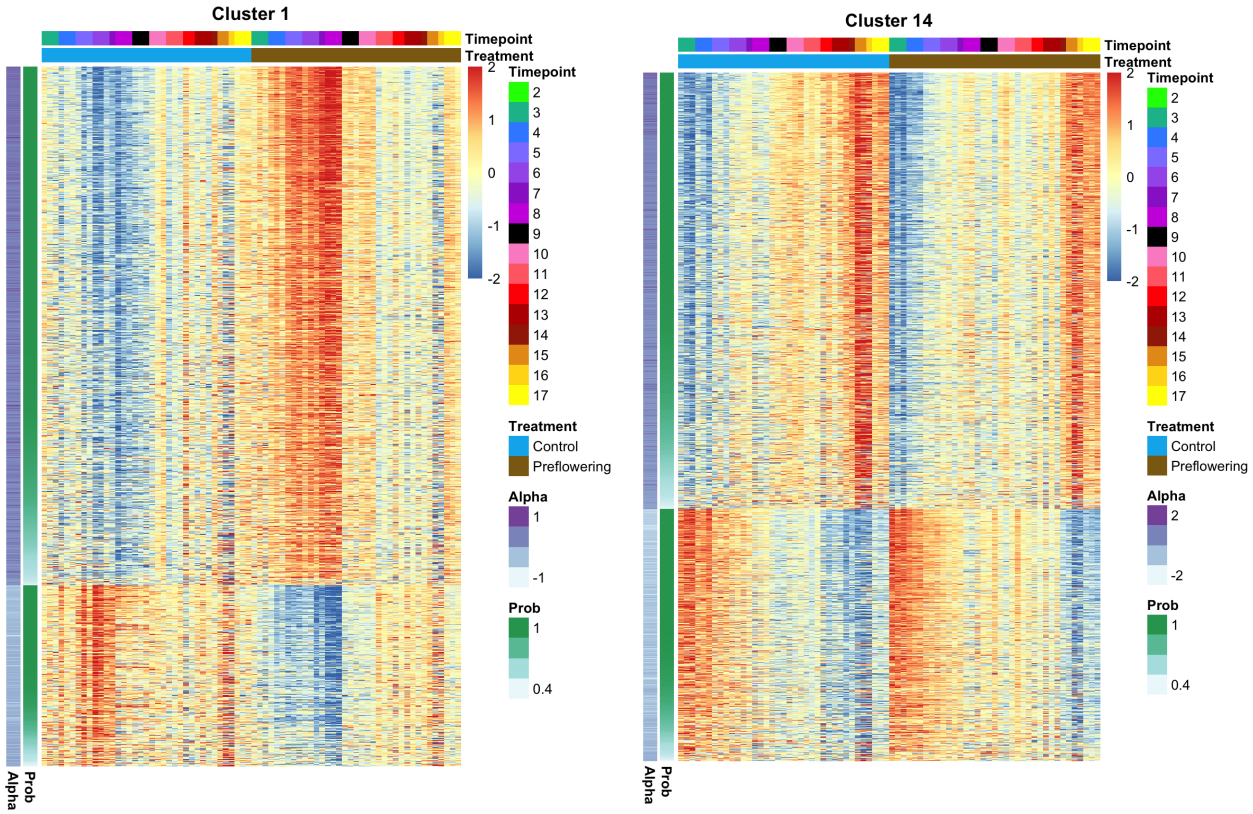
Test of condition difference For a first exploration, the residual bootstrap was used to test for a difference in the shape of temporal pattern between the Pre-flowering drought and Control conditions. The resulting cluster centroids and residual bootstrap p-values for the 20 clusters are shown in Figure 4.5. Clusters have been ordered from smallest p-value to largest, so that the most significant clusters appear first. The p-values have been corrected for multiple testing with the Bonferroni multiple testing correction.

From these results, we can observe that the residual bootstrap produces reasonable results. All of the centroids display a visual difference in temporal pattern between conditions; however, some differences are stronger than others. The clusters with lowest p-values are those with the strongest visual differences between conditions; conversely, those with the highest p-values do not appear to be strikingly different between conditions. Some of these p-values seem anti-conservative, and so the results should be checked with the behavior of the genes within each cluster to ensure that the residual bootstrap p-values are reasonable.

To confirm that the residual bootstrap is producing meaningful results, we explore the behavior of genes in clusters declared differentially expressed or not. Two representative heatmaps of such cases are shown in Figure 4.6. The genes in cluster 1, which has one of the lowest p-values, exhibit remarkably different patterns of temporal expression between conditions. On the other hand, cluster 14, which has one of the highest p-values, contains genes that demonstrate generally similar patterns of temporal expression between conditions. While there is a slight noticeable difference in the temporal patterns, the difference is not significant. This visualization confirms that the residual bootstrap is detecting differentially expressed clusters without obvious false discoveries. The seemingly anti-conservative p-values may be a result of small differences at certain time points that result in a centroid that appears to have more of a difference between conditions than there actually is. Therefore, it is important to investigate the behavior of genes within each cluster to confirm the results of the residual bootstrap test.

Test of genotype difference As a second test, we applied the residual bootstrap method to the slightly more challenging setting of the difference between genotypes. Because fewer differences between genotypes were observed from the per-gene differential expression analysis in Varoquaux et al. [2019], we would expect to discover fewer differentially expressed clusters. This is therefore a good test of the false positive behavior of the residual bootstrap. The cluster centroids visually indicate that there are very few differences in temporal patterns between genotypes. Furthermore, the residual bootstrap method reports p-values that are appropriately high in this non-differentially expressed setting. Figure 4.7 shows the cluster centroids with their associated p-values for the 20 clusters, ordered from most significant to least significant. As compared to the test of differential expression between conditions, the test of differences in genotype finds far fewer differentially expressed clusters, which is an encouragement that the residual bootstrap is not excessively anti-conservative.

Clusters 15 and 2, the only clusters declared significantly different between genotypes in their temporal expression deserve closer consideration. Since the residual bootstrap is testing for only a difference in the *shapes* of the cluster centroids between genotypes, the centroids that are plotted in Figure 4.7 have been scaled so that the centroid in each genotype ranges from zero to one. However, a quick look at the original fitted centroids reveals that the original centroids produced by the mixture model from clusters 15 and 2, shown in Figure 4.8 reveal nothing more than a strong overall shift between genotypes at all conditions. The small variations in temporal pattern within each genotype are magnified when rescaling



(a) Heatmap of genes in cluster 1, which was declared differentially expressed.

(b) Heatmap of genes in cluster 14, which was not declared differentially expressed.

Figure 4.6: Heatmaps of genes from clusters either detected to be differentially expressed between conditions or not, according to the residual bootstrap test of their cluster means.

within each genotype, thus producing the low p-values. This suggests that the residual bootstrap method is likely to be anti-conservative to detect a difference in condition shape when there is in reality only a difference in the overall shift between two conditions.

To further confirm the results of the residual bootstrap method applied to the genotype difference, consider the two heatmaps shown in Figure 4.9. Cluster 18 is a typical example of the behavior of genes from all of the clusters with high p-values: the genes in this cluster demonstrate changes across time but there is no difference in this pattern between the two genotypes. This behavior by the genes confirms what the cluster centroid indicates, which is no difference between the two genotypes; the remaining clusters not provided show the same results of no difference in temporal behavior between conditions, as suggested by their cluster centroids. Cluster 2 provides a further confirmation of what was observed earlier, that this cluster, while detected to have a significant difference in temporal shape between genotypes, only exhibits a difference of an overall shift in expression. There is no visual evidence of the

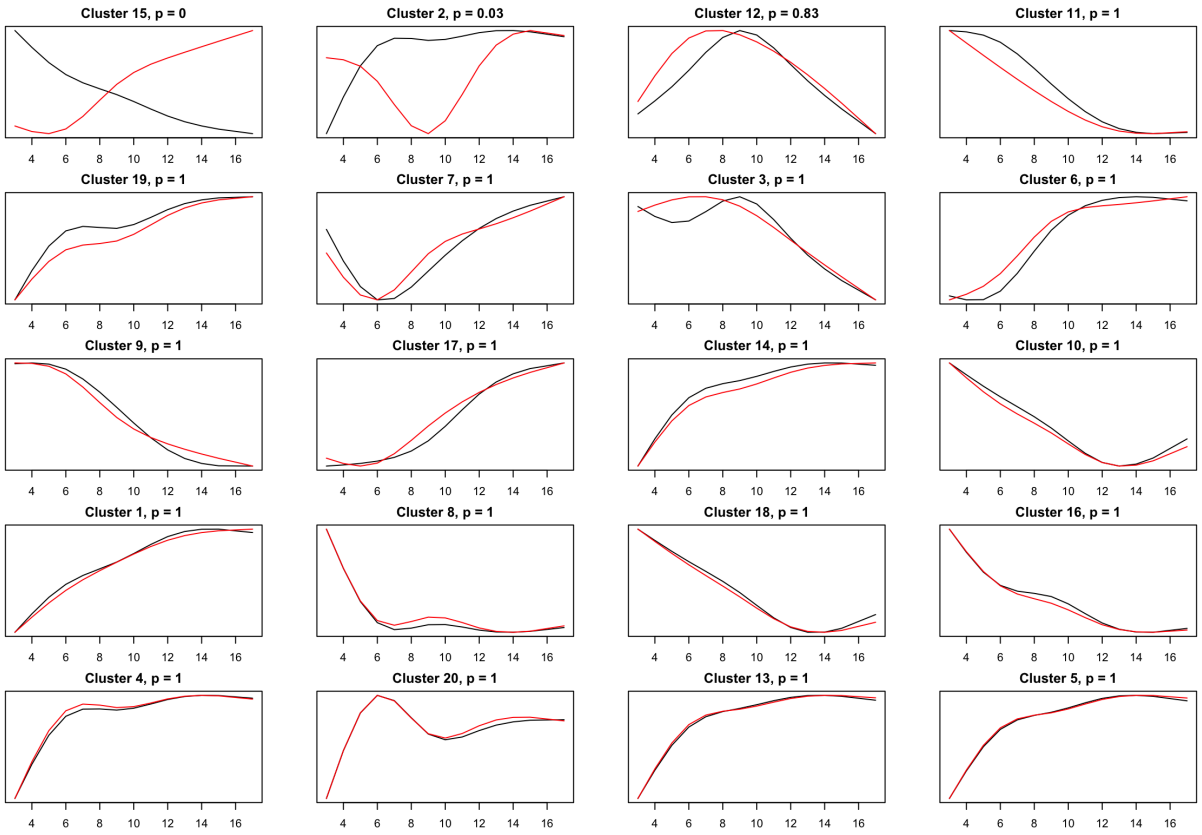


Figure 4.7: Residual bootstrap results of the Genotype differences test of differential expression.

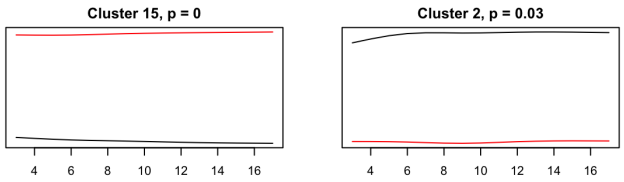
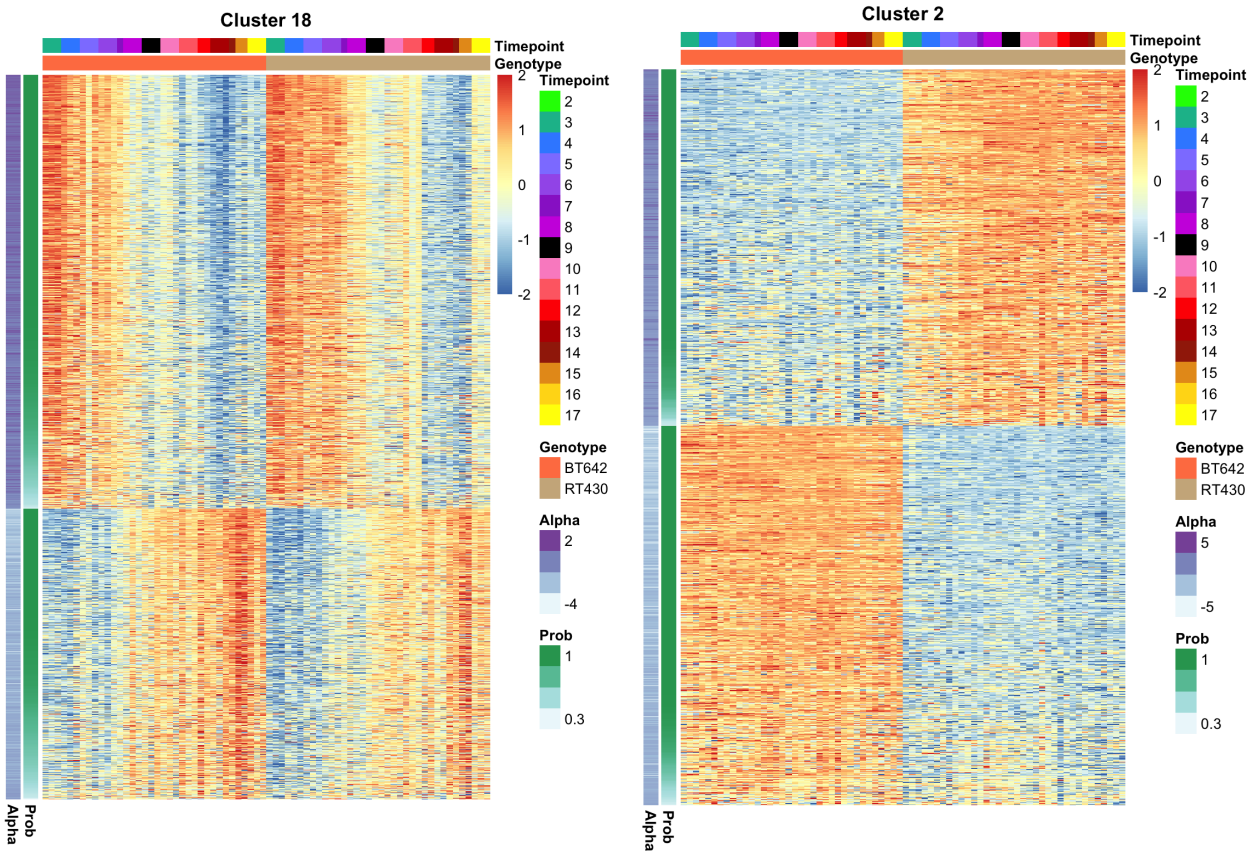


Figure 4.8: Original cluster centroids of the two clusters detected to have differentially expressed shapes of temporal pattern between genotypes.



(a) Heatmap of genes in cluster 18, which was not declared differentially expressed between genotypes.

(b) Heatmap of genes in cluster 2, which was declared differentially expressed, due to the large overall shift in expression.

Figure 4.9: Heatmaps of genes from clusters either detected to be differentially expressed between genotypes or not, according to the residual bootstrap test of their cluster means.

genes in this cluster demonstrating the behavior shown by the rescaled centroids, suggesting that the rescaling of these centroids is magnifying small differences that do not represent a genuine difference in temporal pattern between genotypes.

From this example of genotype comparisons, we have seen that the residual bootstrap is not too anti-conservative with its detection of differential expression, with the small caveat of its error in conflating the difference in condition shift with the difference in condition shape. However, in general, the residual bootstrap produces p-values that are appropriately high in settings where there is obviously not a difference between conditions. Considering together the results from the treatment comparison and the genotype comparison, we conclude that the residual bootstrap has the power to appropriately to detect differentially expressed clusters in a real data setting without an obvious overabundance of false positive results.

Chapter 5

Conclusions

Time course genomics experiments provide a wealth of data, with the potential to uncover rich discoveries related to developmental processes and temporal responses. As sequencing technologies advance and studies of longer duration grow more common, it is necessary to meet the exciting possibilities with careful and rigorous statistical techniques of data analysis. Time-course studies of long duration present unique challenges related to modeling the time-course nature of the data and dealing with the overwhelming number of significant changes shown by genes along a developmental process. To meet these challenges, we have presented in this work a comprehensive method of analyzing time course genomics data that relies spline-based clustering to make sense of the vast number of changes associated with developmental processes.

The general question of clustering similar features has been adapted for a time-course setting that can be applied to a wide variety of data types. By using a mixture model framework, this method takes advantage of probabilistic assumptions to assign each genomic feature a probability of belonging to a cluster, giving a user more information than simply a label assignment as in more basic clustering methods. These probabilities can also be used to filter out less relevant features, allowing a user to narrow the focus to only the most interesting features. In addition, relying on probability distributions has the advantage of making this method widely applicable to a variety of genomics datasets, from microarrays to single cell datasets. Importantly, each cluster is assigned a centroid that provides a simple and accurate representation of the mean temporal pattern within each cluster. These cluster centroids take advantage of a functional spline model to smooth the changes in expression across time, making this method especially useful and relevant to time-course studies of long duration. Additionally, the mixture model incorporates gene-specific parameters, allowing for a greater range of flexibility within each cluster, and leading to better clustering performance. The gene-specific scale parameter in particular provides the unique potential to identify patterns of anti-correlation among clustered genes.

We have built upon these clustering results obtained from the mixture model by developing a test of differential expression that can be applied to each cluster mean. Multiple strategies can be adopted to answer the question of differential expression testing, but we

have argued that the residual bootstrap method does this the most effectively. The residual bootstrap method provides a flexible way to test a variety of relevant biological hypotheses, and the method performs with statistically sound behavior. Careful consideration of the derivation of the residuals used in this method make this a method applicable to a variety of data types and distribution choices. This is a particularly valuable feature, as the majority of alternative approaches to conducting inference on cluster means are limited by assumptions of normality. The residual bootstrap method is therefore a valuable tool for time-course studies. By testing each cluster for significant differences in a relevant hypothesis, a user can statistically identify which clusters are the most relevant and worthy of further exploration.

Finally, we have demonstrated the success and value of these methods by applying the mixture model and differential expression analysis to many simulations and real datasets. Through a variety of simulations, we have shown that the mixture model outperforms alternative clustering methods and provides additional advantages beyond other methods. Applying these methods to real datasets confirmed previously known biological results, indicating the accuracy of the clustering method; the method also uncovered new patterns and discoveries in real datasets, demonstrating the value of this new method. Furthermore, we have shown how the results of the mixture model can be used to assist in downstream analysis including gene set enrichment analysis and the interpretations of the biological significance of clusters produced by the mixture model.

While these methods are interesting simply from an intellectual point of view, it is equally important to develop the practical tools that enable researchers to apply these methods to their own datasets. Therefore, we have developed a comprehensive R package that will fit the mixture model and test clusters for differential expression analysis. The functions do not require any filtering or normalization to be performed on the data beforehand, allowing a researcher to quickly apply this method to a dataset by simply inputting the data and relevant preferences. The package has been optimized for time and memory usage, and is freely available to use.

As time course gene expression studies continue to garner fresh interest in biological research areas, the clustering and differential expression testing methods presented in this work will provide convenient and useful tools to discover fresh biological insights with statistical validity.

Bibliography

Ziv Bar-Joseph. Analyzing time series gene expression data. *Bioinformatics*, 20(16):2493–2503, 2004.

KE Basford, DR Greenway, GJ McLachlan, and D Peel. Standard errors of fitted component means of normal mixtures. *Computational Statistics*, 12(1):1–18, 1997.

James H Bullard, Elizabeth Purdom, Kasper D Hansen, and Sandrine Dudoit. Evaluation of statistical methods for normalization and differential expression in mrna-seq experiments. *BMC bioinformatics*, 11(1):94, 2010.

Steve E Calvano, Wenzhong Xiao, Daniel R Richards, Ramon M Felciano, Henry V Baker, Raymond J Cho, Richard O Chen, Bernard H Brownstein, J Perren Cobb, S Kevin Tschoeke, et al. A network-based analysis of systemic inflammation in humans. *Nature*, 437(7061):1032–1037, 2005.

Gilles Celeux, Merrilee Hurn, and Christian P Robert. Computational and inferential difficulties with mixture posterior distributions. *Journal of the American Statistical Association*, 95(451):957–970, 2000.

Ana Conesa, María José Nueda, Alberto Ferrer, and Manuel Talón. masigpro: a method to identify significantly differential expression profiles in time-course microarray experiments. *Bioinformatics*, 22(9):1096–1102, 2006.

Peter England and Richard Verrall. Analytic and bootstrap estimates of prediction errors in claims reserving. *Insurance: mathematics and economics*, 25(3):281–293, 1999.

Levi Gadye, Diya Das, Michael A Sanchez, Kelly Street, Ariane Baudhuin, Allon Wagner, Michael B Cole, Yoon Gi Choi, Nir Yosef, Elizabeth Purdom, et al. Injury activates transient olfactory stem cell states with diverse lineage capacities. *Cell Stem Cell*, 21(6): 775–790, 2017.

Cheng Gao, Liliam Montoya, Ling Xu, Mary Madera, Joy Hollingsworth, Elizabeth Purdom, Robert B Hutmacher, Jeffery A Dahlberg, Devin Coleman-Derr, Peggy G Lemaux, et al. Strong succession in arbuscular mycorrhizal fungal communities. *The ISME journal*, 13 (1):214–226, 2019.

T Hartl. Bootstrapping generalized linear models for development triangles using deviance residuals. In *CAS E–Forum Fall*, 2010.

Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2(1): 193–218, 1985.

Thomas Jaki, Ting-Li Su, Minjung Kim, and M Lee Van Horn. An evaluation of the bootstrap for model validation in mixture models. *Communications in Statistics-Simulation and Computation*, 47(4):1028–1038, 2018.

Bong-Rae Kim, Timothy McMurry, Wei Zhao, Rongling Wu, and Arthur Berg. Wavelet-based functional clustering for patterns of high-dimensional dynamic gene expression. *Journal of Computational Biology*, 17(8):1067–1080, 2010.

Jaehee Kim. Clustering change patterns using fourier transformation with time-course gene expression data. In *Yeast Genetic Networks*, pages 201–220. Springer, 2011.

Jeffrey T Leek, Eva Monsen, Alan R Dabney, and John D Storey. Edge: extraction and analysis of differential gene expression. *Bioinformatics*, 22(4):507–508, 2006.

Ning Leng, Yuan Li, Brian E McIntosh, Bao Kim Nguyen, Bret Duffin, Shulan Tian, James A Thomson, Colin N Dewey, Ron Stewart, and Christina Kendziorski. Ebseq-hmm: a bayesian approach for identifying gene-expression changes in ordered rna-seq experiments. *Bioinformatics*, 31(16):2614–2622, 2015.

Michael I Love, Wolfgang Huber, and Simon Anders. Moderated estimation of fold change and dispersion for rna-seq data with deseq2. *Genome biology*, 15(12):550, 2014.

Nicholas Lytal, Di Ran, and Lingling An. Normalization methods on single-cell rna-seq data: An empirical survey. *Frontiers in Genetics*, 11:41, 2020.

Ping Ma, Cristian I Castillo-Davis, Wenxuan Zhong, and Jun S Liu. A data-driven clustering method for time course gene expression data. *Nucleic acids research*, 34(4):1261–1269, 2006.

Geoffrey J McLachlan and David Peel. *Finite mixture models*. John Wiley & Sons, 2004.

GJ McLachlan. On the em algorithm for overdispersed count data. *Statistical Methods in Medical Research*, 6(1):76–98, 1997.

Adrian O'Hagan, Thomas Brendan Murphy, Luca Scrucca, and Isobel Claire Gormley. Investigation of parameter uncertainty in clustering using a gaussian mixture model via jackknife, bootstrap and weighted likelihood bootstrap. *Computational Statistics*, 34(4): 1779–1813, 2019.

Rajendra K Pachauri, Myles R Allen, Vicente R Barros, John Broome, Wolfgang Cramer, Renate Christ, John A Church, Leon Clarke, Qin Dahe, Purnamita Dasgupta, et al. *Climate change 2014: synthesis report. Contribution of Working Groups I, II and III to the fifth assessment report of the Intergovernmental Panel on Climate Change*. Ipcc, 2014.

Andrea Rau, Cathy Maugis-Rabusseau, Marie-Laure Martin-Magniette, and Gilles Celeux. Co-expression analysis of high-throughput transcriptome sequencing data with poisson mixture models. *Bioinformatics*, 31(9):1420–1427, 2015.

Arthur E Renshaw and Steve Haberman. On simulation-based approaches to risk measurement in mortality with specific reference to poisson lee–carter modelling. *Insurance: Mathematics and Economics*, 42(2):797–816, 2008.

Davide Risso, Katja Schwartz, Gavin Sherlock, and Sandrine Dudoit. Gc-content normalization for rna-seq data. *BMC bioinformatics*, 12(1):480, 2011.

Davide Risso, Fanny Perraudeau, Svetlana Gribkova, Sandrine Dudoit, and Jean-Philippe Vert. A general and flexible method for signal extraction from single-cell rna-seq data. *Nature communications*, 9(1):1–17, 2018.

Mark D Robinson, Davis J McCarthy, and Gordon K Smyth. edger: a bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics*, 26(1): 139–140, 2010.

Graham EJ Rodwell, Rebecca Sonu, Jacob M Zahn, James Lund, Julie Wilhelmy, Lingli Wang, Wenzhong Xiao, Michael Mindrinos, Emily Crane, Eran Segal, et al. A transcriptional profile of aging in the human kidney. *PLoS biology*, 2(12), 2004.

Jil Sander, Joachim L Schultze, and Nir Yosef. Impulsede: detection of differentially expressed genes in time series data using impulse models. *Bioinformatics*, 33(5):757–759, 2017.

Luca Scrucca, Michael Fop, T Brendan Murphy, and Adrian E Raftery. mclust 5: clustering, classification and density estimation using gaussian finite mixture models. *The R journal*, 8(1):289, 2016.

Fatemeh Seyednasrollah, Asta Laiho, and Laura L Elo. Comparison of software packages for detecting differential expression in rna-seq studies. *Briefings in bioinformatics*, 16(1): 59–70, 2015.

Jason E Shoemaker, Satoshi Fukuyama, Amie J Eisfeld, Dongming Zhao, Eiryo Kawakami, Saori Sakabe, Tadashi Maemura, Takeo Gorai, Hiroaki Katsura, Yukiko Muramoto, et al. An ultrasensitive mechanism regulates influenza virus-induced inflammation. *PLoS pathogens*, 11(6), 2015.

Yaqing Si, Peng Liu, Pinghua Li, and Thomas P Brutnell. Model-based clustering for rna-seq data. *Bioinformatics*, 30(2):197–205, 2014.

Gordon K Smyth. Limma: linear models for microarray data. In *Bioinformatics and computational biology solutions using R and Bioconductor*, pages 397–420. Springer, 2005.

Matthew Stephens. Dealing with label switching in mixture models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 62(4):795–809, 2000.

Kelly Street, Davide Risso, Russell B Fletcher, Diya Das, John Ngai, Nir Yosef, Elizabeth Purdom, and Sandrine Dudoit. Slingshot: cell lineage and pseudotime inference for single-cell transcriptomics. *BMC genomics*, 19(1):477, 2018.

Jeri Stroade. *AGMRC Sorghum profile*, 2020. URL `https://www.agmrc.org/commodities-products/grains-oilseeds/sorghum` [Accessed: 2020-04-21].

Zhivko Taushanov and André Berchtold. Bootstrap validation of the estimated parameters in mixture models used for clustering. *Journal de la Société Française de Statistique*, 160 (1):114–129, 2019.

Cole Trapnell, Davide Cacchiarelli, Jonna Grimsby, Prapti Pokharel, Shuqiang Li, Michael Morse, Niall J Lennon, Kenneth J Livak, Tarjei S Mikkelsen, and John L Rinn. The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells. *Nature biotechnology*, 32(4):381, 2014.

Koen Van den Berge, Hector Roux De Bezieux, Kelly Street, Wouter Saelens, Robrecht Cannoodt, Yvan Saeys, Sandrine Dudoit, and Lieven Clement. Trajectory-based differential expression analysis for single-cell sequencing data. *BioRxiv*, page 623397, 2019.

Nelle Varoquaux, Benjamin Cole, Cheng Gao, Grady Pierroz, Christopher R Baker, Dhruv Patel, Mary Madera, Tim Jeffers, Joy Hollingsworth, Julie Sievert, et al. Transcriptomic analysis of field-droughted sorghum from seedling to maturity reveals biotic and metabolic responses. *Proceedings of the National Academy of Sciences*, 116(52):27124–27132, 2019.

Ingmar Visser, Maartje EJ Raijmakers, and Peter CM Molenaar. Confidence intervals for hidden markov model parameters. *British journal of mathematical and statistical psychology*, 53(2):317–327, 2000.

Sophie Weiss, Zhenjiang Zech Xu, Shyamal Peddada, Amnon Amir, Kyle Bittinger, Antonio Gonzalez, Catherine Lozupone, Jesse R Zaneveld, Yoshiki Vázquez-Baeza, Amanda Birmingham, et al. Normalization and microbial differential abundance strategies depend upon data characteristics. *Microbiome*, 5(1):27, 2017.

Daniela M Witten et al. Classification and clustering of sequencing data using a poisson model. *The Annals of Applied Statistics*, 5(4):2493–2518, 2011.

Ling Xu, Dan Naylor, Zhaobin Dong, Tuesday Simmons, Grady Pierroz, Kim K Hixson, Young-Mo Kim, Erika M Zink, Kristin M Engbrecht, Yi Wang, et al. Drought delays development of the sorghum root microbiome and enriches for monoderm bacteria. *Proceedings of the National Academy of Sciences*, 115(18):E4284–E4293, 2018a.

Ling Xu, Dan Naylor, Zhaobin Dong, Tuesday Simmons, Grady Pierroz, Kim K Hixson, Young-Mo Kim, Erika M Zink, Kristin M Engbrecht, Yi Wang, et al. Drought delays development of the sorghum root microbiome and enriches for monoderm bacteria. *Proceedings of the National Academy of Sciences*, 115(18):E4284–E4293, 2018b.