THE UNIVERSITY OF CHICAGO

POISSON MULTISCALE METHODS FOR HIGH-THROUGHPUT SEQUENCING DATA

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCES
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

DEPARTMENT OF STATISTICS

BY
ZHENGRONG XING

CHICAGO, ILLINOIS
DECEMBER 2016

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my adviser, Matthew Stephens, for his endless support and superb guidance over the course of this entire project. I am grateful for all the invaluable insights, constructive feedbacks and intellectual conversations he has shared over the years, and am truly honored to have worked with such an outstanding faculty member.

I would also like to thank the other members of my advising committee, Dan Nicolae and John Lafferty, for taking time out of their busy schedules to provide insightful advice to the writing of this dissertation.

I am grateful to my peers, both from the Department of Statistics and the Stephens Lab, for their encouragement, support and constructive debates. It has been a pleasure to work in such a collaborative environment, where ideas and knowledge are so readily exchanged. In particular, I would like to thank Heejung Shim for involving me in an absolutely engaging project that is part of this dissertation. I would also like to thank Mengyin Lu, Wei Wang, and Kushal Dey for their statistical expertise, Gao Wang, Siming Zhao, Michael Turchin, and Hussein Al-Asadi for their knowledge of computational and applied biology, as well as Raman Shah and John Zekos for their computational proficiency.

Finally, I would like to thank my parents for their unconditional love and support throughout the course of my PhD, making the journey that much smoother. In particular, special thanks to my dad for taking time away from his own research to provide valuable feedback to mine.

ABSTRACT

In this dissertation, we focus on the problem of analyzing data from high-throughput sequencing experiments. With the emergence of more capable hardware and more efficient software, these sequencing data provide information at an unprecedented resolution. However, statistical methods developed for such data rarely tackle the data at such high resolutions, and often make approximations that only hold under certain conditions.

We propose a model-based approach to dealing with such data, starting from a single sample. By taking into account the inherent structure present in such data, our model can accurately capture important genomic regions. We also present the model in such a way that makes it easily extensible to more complicated and biologically interesting scenarios.

Building upon the single-sample model, we then turn to the statistical question of detecting differences between multiple samples. Such questions often arise in the context of expression data, where much emphasis has been put on the problem of detecting differential expression between two groups. By extending the framework for a single sample to incorporate additional group covariates, our model provides a systematic approach to estimating and testing for such differences. We then apply our method to several empirical datasets, and discuss the potential for further applications to other biological tasks.

We also seek to address a different statistical question, where the goal here is to perform exploratory analysis to uncover hidden structure within the data. We incorporate the single-sample framework into a commonly used clustering scheme,

and show that our enhanced clustering approach is superior to the original clustering approach in many ways. We then apply our clustering method to a few empirical datasets and discuss our findings.

Finally, we apply the shrinkage procedure used within the single-sample model to tackle a completely different statistical issue: nonparametric regression with heteroskedastic Gaussian noise. We propose an algorithm that accurately recovers both the mean and variance functions given a single set of observations, and demonstrate its advantages over state-of-the art methods through extensive simulation studies.

CHAPTER 1

INTRODUCTION

The advent of high-throughput next generation sequencing (NGS) data has revolutionized the way scientists think about genomic data. Previously unfeasible research questions are now the focus of biologists and statisticians worldwide, and the amount of information cataloged so far has been tremendous. Even so, researchers have yet to take full advantage of the full depth of information provided by NGS data. In this dissertation, we aim to provide a model-based framework for analyzing NGS data at the single base resolution, and attempt to tackle a variety of biological problems by extending this framework in multiple directions.

1.1 Background

We first give a very brief overview of NGS technology and the data produced by such technology, ignoring certain details for brevity. In its current form, NGS refers to post-Sanger technology for sequencing DNA fragments. However, there exists a variety of sequencing experiments aimed at answering different biological questions. Here we list three of the most common sequencing experiments discussed later in this dissertation, although they are by no means comprehensive:

1. RNA sequencing, or RNA-seq (e.g. Wang et al. (2009)). In molecular biology, gene expression is the most fundamental quantitative measure of the proteins synthesized by the gene, and also the most widely-used quantity for determining the observable traits of an organism. Transcription is the process by which

messenger RNA (mRNA) is produced using DNA as the base template, before being translated into proteins. RNA-seq measures the amount of mRNA transcribed from a region of DNA corresponding to a particular gene, and hence is often used to study differences in gene expression (differential expression) between groups of samples using methods such as *edgeR* (Robinson et al. (2010)) and *DESeq* (Anders and Huber (2010)).

2. Chromatin Immunoprecipitation sequencing, or ChIP-seq (e.g. Robertson et al. (2007)). To better understand the mechanisms underlying gene expression and the associated regulatory processes, much work has been focused on a particular class of proteins that directly control the transcription process. These proteins, aptly named transcription factors (TFs), bind to specific sites on the genome, and one of the main goals of methods developed for ChIP-seq data (e.g. Zhang et al. (2008)) is to find the location of these binding sites. ChIP-seq experiments can also be used to identify sites where histones are modified, and locating histone modification sites is an important step in understanding the epigenetic factors that affect gene expression.

3. DNase I hypersensitive sites sequencing, or DNase-seq (e.g. Song and Crawford (2010)). While ChIP-seq experiments target specific TF binding sites or histone modification sites, DNase-seq experiments serve to understand gene regulation from another angle. Regions where regulatory processes occur (such as TF binding sites or histone modification sites) typically affect chromatin structure by opening up the originally condensed portions of the chromatin. DNase I is an endonuclease that targets these accessible portions of the chromatin, also

known as DNase I hypersensitive sites (DHSs), and thus acts as a surrogate for locating regulatory regions. DNase-seq experiments use DNase I to target DHSs, and methods (e.g. Pique-Regi et al. (2011)) have been developed for quantifying the location of these regions.

Although we have listed three seemingly different types of sequencing experiments, the process of sequencing is actually identical for all types of sequencing experiments, as the inputs to the sequencing machines are always DNA fragments. The key difference between different sequencing experiments is thus how the DNA fragments are generated. In RNA-seq, transcribed mRNA sequences are targeted using special enzymes, before being reverse transcribed to complementary DNA (cDNA) sequences. These cDNA sequences are then fragmented and then amplified through polymerase chain reaction (PCR) before being sequenced. In ChIP-seq, DNA fragments bound by TFs are sheared from the rest of the DNA sequences and extracted from the rest of the cell debris using immunoprecipitation. The DNA fragments are then unlinked from the bound TFs, then amplified using PCR and subsequently sequenced. In DNase-seq, nuclei from cells are lightly digested using DNase I and the cleaved DNA fragments at DHSs are extracted. The DNA frament are then amplified using PCR before being supplied to sequencing machines.

Having obtained the DNA fragments from any given sequencing experiment, the main purpose of the sequencing machines is to determine the exact nucleotide sequence of the DNA fragments. For the most widely used Illumina sequencing platforms, this is typically done by using an independent pool of fluorescently labeled nucleotide fragments. This way, one can determine the sequence reads for each DNA

fragment by recording the colors of the nucleotides that pair complementarily with the nucleotides on the DNA fragment. Current technology allows for simultaneous parallel sequencing of a massive number of DNA fragments to greatly speed up the process.

Given the sequence information for the DNA fragments, the next step is to align these fragments to a reference genome. While there will be differences between the reference genome and the genomes of the individuals from which the DNA fragments originated from, the fragments are typically long enough (>30 base pairs) for relatively accurate alignment. After passing through various quality control steps and other bioinformatics pipelines, we can eventually obtain counts at each base of the genome, which is simply the total number of DNA fragments mapped to that location. For the rest of this dissertation however, we will use Y_b to denote the number of fragments whose first nucleotide maps to location b. Note that this differs from the more commonly-used "coverage", which is the number of fragments that cover base b. As will be expounded upon in Chapter 2, this procedure limits the amount of correlation between adjacent locations when modeling the sequencing counts.

1.2 Previous related work

Although NGS technology provides data at an unprecedented resolution, traditional methodologies rarely exploit the structure of the data at this resolution. For example, previously mentioned methods for detecting differences in gene expression such as *edgeR* and *DESeq* aggregate RNA-seq counts within each gene and perform statistical tests for differential expression at the gene level; methods aimed at detecting "peaks"

in ChIP-seq data, such as MACS (Zhang et al. (2008)) and GEM (Guo et al. (2012)), use sliding windows and DNA sequences respectively, ignoring the counts at the base pair resolution for the most part. Although these methods have proven to be extremely useful so far, our goal is to uncover more intricate structure by making use of counts at each base.

More recently, several approaches have been proposed that attempt to answer biological questions at the base pair resolution. Frazee et al. (2014) tests for differential expression at each base by first estimating differences between samples using a linear model, and then modelling the presence or absence of differential expression using a Hidden Markov Model. Shim and Stephens (2015) tests for association between a covariate of interest and sequencing counts from multiple samples by exploiting the spatial structure of sequencing profiles from each sample. Specifically, they transform the matrix of counts using wavelets, and then model the relationship between the transformed data and the covariate of interest using a linear model. Lee and Morris (2016) uses a similar wavelet transformation on the matrix of sequencing counts and can model more complicated types of association in the transformed space, but uses a computationally expensive Markov Chain Monte Carlo (MCMC) algorithm to estimate parameters of the model. Fusi and Listgarten (2016) also exploits the spatial structure present in the sequencing profiles, but models the structure using Gaussian Processes instead of wavelet transformations. Wavelet transformations are also used in Liu and Song (2016), where the wavelet decompositions of ribosomal sequencing counts at different scales are regressed on neighboring codon features using a regularized linear regression, with the goal of predicting ribosome profiles.

While these recent methods do target at the base pair resolution, they are not easily extensible. For example, Frazee et al. (2014) and Liu and Song (2016) develop context-specific approaches for testing differential expression and prediction of ribosomal profiles respectively, but do not directly model the sequencing counts in a manner applicable to other problems. On the other hand, while Shim and Stephens (2015), Lee and Morris (2016) and Fusi and Listgarten (2016) do indeed model the sequencing counts directly, they make use of Gaussian models. In practice, sequencing counts are typically very low (with a lot of 0's), and sequencing experiments often have small sample sizes. These two factors may substantially lower the power of models relying on Gaussian assumptions, and can be problematic for certain scenarios.

1.3 Dissertation outline

In Sections 1.1 and 1.2 we have provided the background behind NGS data and some of the methodologies developed to analyze such data. We also discussed some of the shortcomings of these methods, and now propose a flexible framework for modeling NGS data at the base pair resolution.

In Chapter 2 we start off by modeling the sequencing counts for a single sample. We justify a Poisson model for the counts at each base pair, and frame this in the context of nonparametric Poisson regression. While there have been various methods developed specifically for this problem, we propose a novel Bayesian multiscale framework that is adaptive and provides more accurate estimates of the underlying Poisson intensities compared to state-of-the art methodologies, as desmonstrated in

the simulation studies that follow. We also show the potential for our approach to analyze ChIP-seq data using a sample ChIP-seq dataset. By modeling the sequencing counts for a single sample in this manner, extensions to multiple samples and other biological problems of interest will be made much simpler.

In Chapter 3 we extend the one sample model to include multiple samples. In this situation the primary interest lie in detecting associations with a covariate of interest and estimating differences between groups. We extend the framework introduced in Chapter 2 to model the relationship between multiple samples and a covariate of interest using a Poisson Generalized Linear Model (GLM). We conduct simulation studies to demonstrate its superiority over both Shim and Stephens (2015) and methods targeted at gene-level resolution data such as *edgeR* and *DESeq*. We then apply our method on a sample dataset to demonstrate its utility, and explore potential alternatives to our method. Finally, we highlight the potential application of our method in inferring TF binding sites, which is another area of active research.

In Chapter 4 we extend the one sample model in another direction. In this case, we are interested in grouping together samples based solely on their sequencing profiles. We include the one-sample multiscale model as part of a grade-of-membership clustering approach, which is an extension of traditional clustering methods which allows each sample to "belong" to more than one cluster. We then demonstrate the advantages of including the multiscale model through a simple simulation study as well as an example RNA-seq dataset, before applying it to RNA-seq data from a well known study and highlighting some of the interesting structures we discovered. This particular application demonstrates the considerable potential of our cluster-

ing method, which could more generally serve as a convenient tool for exploratory analysis before narrowing down on promising patterns and structures.

In Chapter 5 we extend the one sample model described in Chapter 2 in a completely different direction. Specifically, we apply the same shrinkage procedure used in Chapter 2 to nonparametric Gaussian regression. Unlike most methods targeted at this problem that deal with homoskedastic data, we show that the same shrinkage procedure allows our method to accurately denoise Gaussian observations with fluctuating variances. We demonstrate the superiority of our method over competing methods through an extensive simulation study, and apply it to a sample dataset to illustrate its uses.

Finally, we note that most of our methods and analyses are available on GitHub, allowing one to easily reproduce and extend upon the results presented in this dissertations. More details can be found Appendix A.11.

CHAPTER 2

MODELING THE SEQUENCING PROFILE FROM A SINGLE SAMPLE[1]

2.1 Overview

Suppose, initially, that we have the sequencing reads from a single biological sample, where the number of reads at base pair (bp) b is given by Y_b. As discussed in Section 1.1, the sequencing reads are generated by mapping sequenced DNA fragments to the reference genome (although we use only the first base of any DNA fragment). As such, we can consider the observations as being generated from a multinomial experiment, where the total number of "trials" refers to the number of DNA fragments n, and each "category" corresponds to a base pair b, with the categorical probability p_b being the probability that (the first base of) each DNA fragment maps to the said base pair. Note the number of DNA fragments is typically very large (10^7 or more for a genome-wide experiment), while the categorical probabilities are correspondingly extremely small. Furthermore, the covariance between any two Y_b and Y_b' is given by $-np_bp_b'$, which is negligible for small p_b and p_b'. These facts allow one to readily approximate the multinomial setup by an inhomogeneous Poisson Process, where the Poisson intensities are directly proportional to the multinomial probabilities (with the constant of proportionality being n). In fact, in certain scenarios it might be appropriate to model the total number of DNA fragments n as random and coming from a Poisson distribution, in which case the counts at each base exactly follow a

1. Parts of this chapter are included in a preprint https://arxiv.org/abs/1605.07787.

Poisson distribution. See Marioni et al. (2008) for a more detailed discussion.

Specifically, given a region of length B, we model the sequencing reads as

$$Y_b \sim \text{Poi}(\lambda_b), \qquad (b = 1, \ldots, B). \tag{2.1}$$

with the Y_b's independent. The Poisson intensities λ_b are of interest, as regions with λ values larger than background noise (which is low on average) typically indicate some sort of biological activity in that region, with the sequenced DNA fragments being more likely to come from these regions. For example, larger λ values in ChIP-seq is usually an indication of a transcription factor binding site or histone modification site, while a similar observation in DNase-seq implies higher chromatin accessibility. The goal is thus to estimate the λ_b's as accurately as possible.

Estimating the λ's as in (2.1), also known as nonparametric regression or signal de-noising, is a widely studied problem. Here we make the assumption that the λ_b's are spatially-structured. By "spatially-structured" we mean that λ_b will often be similar to $\lambda_{b'}$ for small $|b - b'|$, though we do not rule out occasional abrupt changes in λ_b. Wavelet-based and similar multi-scale methods have been popular approaches to tackle this problem. For example, variance stabilizing transforms together with normal approximations have been used in Donoho (1993) and Fryzlewicz and Nason (2001), by noting that the variance of a Poisson count is equal to its mean. Kolaczyk (1996) derived suitable thresholds achieving optimal asymptotic properties in the context of wavelet transformations, similar to the thresholds in the i.i.d. Gaussian case. Multiscale analysis using recursive dyadic partitions within a Bayesian framework has also been developed by Kolaczyk (1999) and Timmermann and Nowak

(1999) making use of a particular form of likelihood factorization.

Here, we introduce a new method *smash* (SMoothing by Adaptive Shrinkage) to perform the de-noising task. While *smash* is similar to Kolaczyk (1999) and Timmermann and Nowak (1999) in that it makes use of the same likelihood factorization, we improve upon the conjugate Beta priors by using a highly adaptive shrinkage procedure *ash* (Adaptive SHrinkage; see Stephens (2016)) in its stead. To explain *smash* in greater detail, first recall the following elementary distributional result: if Y_1, Y_2 are independent, with $Y_j \sim \text{Poi}(\lambda_j)$ then

$$Y_1 + Y_2 \sim \text{Poi}(\lambda_1 + \lambda_2) \tag{2.2}$$

$$Y_1|(Y_1 + Y_2) \sim \text{Bin}(Y_1 + Y_2, \lambda_1/(\lambda_1 + \lambda_2)). \tag{2.3}$$

To extend this to $T = 4$, introduce the notation $v_{i:j}$ to denote, for any vector v, the sum $\sum_{t=i}^{j} v_t$. Then

$$Y_{1:4} \sim \text{Poi}(\lambda_{1:4}) \tag{2.4}$$

$$Y_{1:2}|Y_{1:4} \sim \text{Bin}(Y_{1:4}, \lambda_{1:2}/\lambda_{1:4}) \tag{2.5}$$

$$Y_1|Y_{1:2} \sim \text{Bin}(Y_{1:2}, \lambda_1/\lambda_{1:2}) \tag{2.6}$$

$$Y_3|Y_{3:4} \sim \text{Bin}(Y_{3:4}, \lambda_3/\lambda_{3:4}). \tag{2.7}$$

Together these models are exactly equivalent to $Y_j \sim \text{Poi}(\lambda_j)$, and they decompose the overall distribution $Y_1, \ldots, Y_4$ into parts involving aspects of the data at increasing resolution: (2.4) represents the coarsest resolution (the sum of all data points),

whereas (2.6) and (2.7) represent the finest resolution, with (2.5) in between. Further, this representation suggests a reparametrization, from $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \lambda_3, \lambda_4)$ to $\lambda_{1:4}$ plus the binomial parameters $\boldsymbol{p} = (\lambda_{1:2}/\lambda_{1:4}, \lambda_1/\lambda_{1:2}, \lambda_3/\lambda_{3:4})$, where p_1 controls lower-resolution changes in the mean vector λ and p_2, p_3 control higher resolution changes. This idea extends naturally to $B = 2^J$ for any integer J, reparametrizing $\boldsymbol{\lambda}$ into its sum $\lambda_{1:B}$ and a vector $\boldsymbol{p}$ of $B-1$ binomial probabilities that capture features of $\boldsymbol{\lambda}$ at different resolutions. This can be thought of as an analogue of the Haar wavelet transform of $\boldsymbol{\lambda}$ for Poisson data. For a detailed description of the likelihood factorization, see Appendix A.1. While Appendix A.1 uses a double index notation, as is standard in the wavelet literature, we use a single index here for notational simplicity.

Note that, in this reparametrization, $p_j = 0.5 \, \forall j$ corresponds to the case of a constant mean vector, and values of p_j far from 0.5 correspond to large changes in μ (at some scale). Thus estimating a spatially-structured $\boldsymbol{\lambda}$ can be achieved by shrinkage estimation of $\boldsymbol{p}$, with shrinkage towards $p_j = 0.5$. Both Kolaczyk (1999) and Timmermann and Nowak (1999) use purpose-built Bayesian models to achieve this shrinkage, by introducing a prior distribution on elements of $\boldsymbol{p}$ that is a mixture of a point mass at 0.5 (creating shrinkage toward 0.5) and one or more (fixed number of) Beta distributions. Here we take a different approach, reparametrizing $\alpha_j = \log(p_j/(1-p_j))$, and then using an adaptive shrinkage method *ash* (Stephens (2016)) to shrink α_j towards 0. The main purpose of the reparametrization will be more readily apparent later, when we deal with multiple samples. For now, it serves as a convenient tool for us to approximate the logit-transformed likelihood by

a Gaussian likelihood, which is the default likelihood in the *ash* software.

In practice it is important to group the α_j's by their resolution level before shrinking, so that a different distribution g is estimated for each resolution. This is the usual way that EB approaches are applied in this context (e.g. Johnstone and Silverman (2005)) and indeed is crucial because the underlying distribution g will vary with resolution (because smoothness of $\boldsymbol{\lambda}$ will vary with resolution).

To explain briefly the shrinkage procedure in *ash*, we note that it was originally intended as a flexible Empirical Bayes (EB) procedure for estimating quantities $\beta = (\beta_1, \ldots, \beta_n)$ from noisy estimates $\hat{\beta} = (\hat{\beta}_1, \ldots, \hat{\beta}_n)$ and their corresponding standard errors $\hat{s} = (\hat{s}_1, \ldots, \hat{s}_n)$. In its simplest form it assumes the hierarchical model

$$\beta_j \,|\, \hat{s}_j \sim g(\cdot) \tag{2.8}$$

$$\hat{\beta}_j \,|\, \beta_j, \hat{s}_j \sim N(\beta_j, \hat{s}_j^2), \tag{2.9}$$

where the distribution g is modelled using a mixture of zero-centered normal distributions. That is,

$$g(\cdot) = \sum_{k=0}^{K} \pi_k N(\cdot; 0, \sigma_k^2), \tag{2.10}$$

where the mixture weights $\pi_0, \ldots, \pi_K$ are non-negative and sum to 1, and $N(\cdot; \mu, \sigma^2)$ denotes the density of a normal distribution with mean μ and variance σ^2. A key idea, which substantially simplifies inference, is to take $\sigma_0, \ldots, \sigma_K$ to be a fixed grid of values ranging from very small (e.g. $\sigma_0 = 0$, in which case g includes a point mass at 0) to very large. Estimating g then boils down to estimating the mixture weights π, which is done by maximum likelihood. Maximizing this likelihood is a

convex optimization problem, and can be performed very efficiently using interior point methods (Koenker and Mizera (2014)), or more simply (though less efficiently for large problems) using an accelerated EM algorithm (Varadhan (2014)).

Given an estimate $\hat{g}$ for g, the conditional distributions $p(\beta_j \mid \hat{\beta}, \hat{s}, \hat{g})$ are analytically tractable, and the posterior mean $E(\beta_j \mid \hat{\beta}, \hat{s}, \hat{g})$ provides a shrinkage point estimate for β_j.

The representation (2.10) provides a flexible family of unimodal and symmetric distributions g. Specifically, with a sufficiently large and dense grid of $\sigma_0, \ldots, \sigma_K$ the distribution g in (2.10) can arbitrarily accurately approximate any scale mixture of normals. This family includes, as a special case, many distributions used in shrinkage estimation, such as the "spike and slab", double-exponential (Laplace), and t distributions (Clyde and George (2000); Johnstone and Silverman (2005)). In this sense *ash* is more flexible than previous EB approaches. Furthermore, in many ways this more flexible approach actually *simplifies* inference, because the only parameters to be estimated are the mixture proportions (the variances are fixed).

Stephens (2016) also introduces various embellishments that are implemented in the `ashr` package, including generalizing the normal likelihood to a t likelihood, and dropping the symmetric constraint on g by replacing the mixture of normals with a more flexible (though less smooth) mixture of uniforms. We do not use these embellishments here.

Hence, to apply *ash* the main ingredients we need are estimates $\hat{\alpha}_j$ and their corresponding standard errors $\hat{s}_j$ for each j, assuming a Gaussian likelihood. This involves estimating the log-odds ratio $\alpha_j = \log(p_j/(1-p_j))$, and its standard error,

which is a well-studied problem (e.g. Gart and Zweifel (1967)). The main challenge is in dealing satisfactorily with cases where the maximum likelihood estimator for α_j is infinite. We use estimates based on results from Gart and Zweifel (1967); see Appendix A.1.1 for details on the choice of the estimator and its standard error.

As a last step, we need to estimate $\boldsymbol{\lambda}$. The simplest way to to achieve this is to estimate α_j by its posterior mean, as output by *ash*, and then reverse the reparametrization (2.4)-(2.7). The resulting estimate of λ_t is the exponential of the posterior mean for $\log(\lambda_t)$ (because each $\log(\lambda_t)$ is a linear combination of α_j's). Alternatively we can estimate λ_b by approximating its posterior mean using the Delta method; see Appendix A.1.2 for details. Both methods are implemented in *smash*; for the results here we use the latter method to be more comparable with previous approaches that estimate λ on the raw scale rather than log scale.

Note that we actually use a translation invariant (TI) version of the above procedure; see Appendix A.2 for motivation and details on this procedure.

2.2 Simulation results

Here we apply *smash* to both simulated and real data to assess its performance. For simulated data, we compared *smash* with six other methods, but we focus here on the comparisons with the best-performing other methods, which are Haar-Fisz (HF) (Fryzlewicz and Nason (2004)) and BMSM (Kolaczyk (1999)). Results for other methods are given in Appendix A.1.3. The performance of each method is

determined through the mean integrated squared error (MISE), which is defined as:

$$MISE = \sum_{r=1}^{R} \frac{10000 * \sum_{i=1}^{n} (\hat{\mu}_i^{(r)} - \mu_i)^2}{\sum_{i=1}^{n} \mu_i^2} \tag{2.11}$$

which is simply the standard MSE rescaled to produce more comparable numbers. Here R is the number of runs in each simulation (100 in our study) and n is the sample size (1024 in our study).

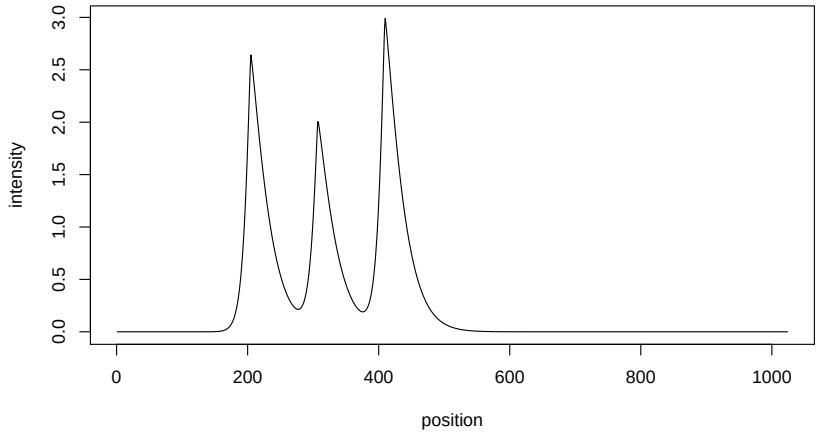
Note that BMSM, like *smash*, is an EB method, but with a less flexible prior distribution on the multiscale coefficients. The HF method involves first performing a transformation on the Poisson counts, and applying Gaussian wavelet methods to the transformed data. There are many choices for Gaussian wavelet methods, and performance depends on these choices, with different choices being optimal for different data sets. The settings we used here for HF are documented in Appendix A.1.3, and were chosen by us to optimize (average) performance through moderately extensive experimentation on a range of simulations.

To compare the methods we simulated data from several different test functions from Timmermann and Nowak (1999), Fryzlewicz and Nason (2004) and Besbeas et al. (2004). We varied the minimum and maximum intensity of each test function, using (min,max) intensities of (0.01,3), (1/8,8) and (1/128,128). For each test function and intensity level we simulated 100 datasets, each with $T = 1024$ data points. We focus on results for the first two intensity settings, which have smaller average intensities. These settings produce smaller average counts, making them more challenging, and also more representative of the kinds of genomic application that we

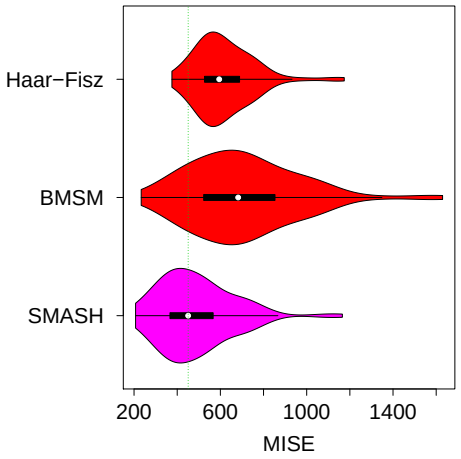
consider below. Complete results are included in Appendix A.1.3.

In summary, *smash* outperformed both HF and BMSM in the majority of simulations, with the gain in accuracy being strongest for the more challenging lower-intensity scenarios. A typical result is shown in Figure 2.1. Among the other two methods, BMSM is the more consistent performer. HF, with the settings used here, performs quite variably, being worse than the other methods in most scenarios, but occasionally performing the best (specifically for the Angles, Bursts and Spikes test functions, with (min,max) intensity (1/128,128)). As noted above, the HF transform can be used with many settings, so results here should be viewed only as a guide to potential performance.

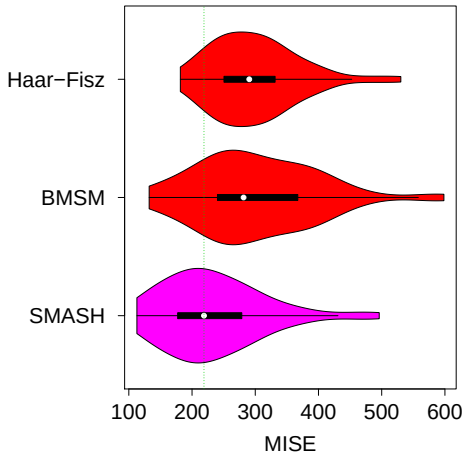
One disadvantage of the HF transform is that, to achieve translation invariance, the transform has to be done explicitly for each shift of the data: the tricks usually used to do this efficiently (Coifman and Donoho (1995)) do not work here. Thus, making HF fully translation invariant increases computation by a factor of T, rather than the factor of $\log(T)$ for the other methods. Here we follow advice in Fryzlewicz and Nason (2004) to reduce the computational burden by averaging over 50 shifts of the data rather than T. Even so, HF was substantially slower than the other methods. A direct comparison of computational efficiency between *smash* and BMSM is difficult, as they are coded in different programming environments. Nevertheless, similarities between the two methods suggest that they should have similar computational cost. Both *smash* and BMSM took, typically, less than a second per dataset in our simulations.



(a)



(b)



(c)

Figure 2.1: Comparison of methods for denoising Poisson data for the "Bursts" test function. Panel (a) shows the (unscaled) test function. The violin plots in (b) and (c) show distributions of MISE for each method over 100 datsets, with smaller values indicating better performance. Panel (b) is for simulations with a (min,max) intensity of (0.01,3), and panel (c) is for simulations with a (min,max) intensity of (1/8,8). The dashed green line indicates the median MISE for *smash*.

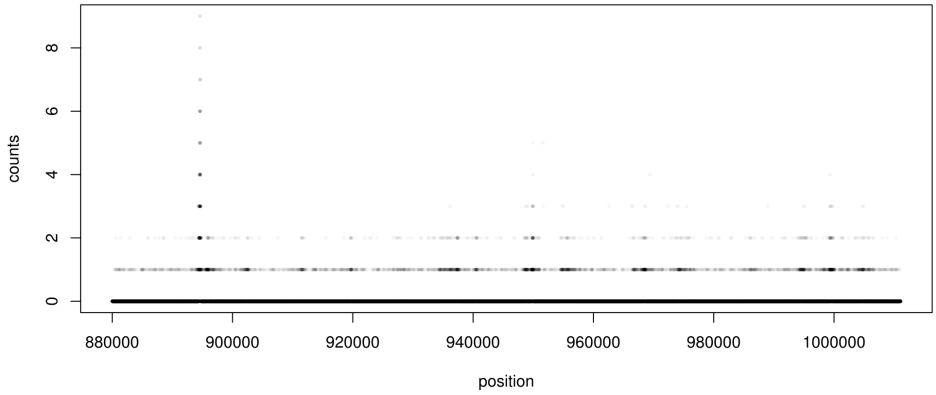
2.3 Case study: detecting peaks in ChIP-seq data

We apply *smash* to a small example ChIP-seq dataset. As noted above, in ChIP-seq data the Poisson intensity at base b is related to the strength of the binding of the transcription factor near b. As such, identifying peaks in the intensity functions can help discover regions where binding occurs, which is an important component of understanding gene regulation. Consequently there are many methods published for "peak detection" in ChIP-seq data (Wilbanks and Facciotti (2010)). Our goal here is to briefly outline how *smash* could provide an alternative approach to the analysis of ChIP-seq data. The idea is simply to estimate the underlying intensity function, and then identify "peaks" as regions where the estimated intensity exceeds some threshold.

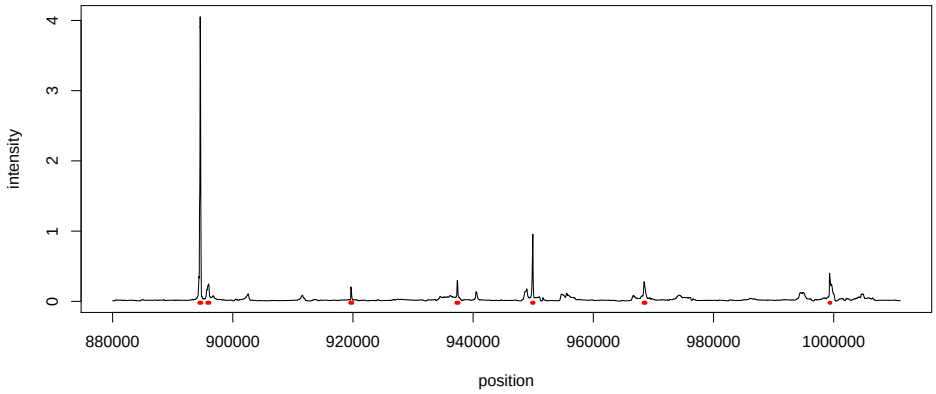
The resulting estimated intensity from *smash* is shown in Figure 2.2b, overlaid on peaks called by the popular peak calling software MACS (Zhang et al. (2008)). The locations with the strongest *smash* intensity estimates corresponds to peaks found by MACS. However, the intensity estimate also suggests the presence of several additional weaker peaks not identified by MACS.

The reliable calling of peaks in ChIP-seq data is a multi-faceted problem, and a full assessment lies outside the scope of this discussion. Nonetheless, these results suggest that this approach could be worth pursuing. One nice feature of our multi-scale Poisson approach is that it deals well with a range of intensity functions, and could perform well even in settings where peaks are broad and/or not especially well defined. In contrast, the performance of different peak-finding algorithms is often reported to be quite sensitive to the "kinds" of peak that are present, so an algorithm

that performs well in one setting may perform poorly in another.



(a)



(b)

Figure 2.2: Illustration of the potential for *smash* to identify peaks in ChIP-seq data. The data are ChIP-seq for the transcription factor *YY1* in cell line GM12878 collected by the ENCODE (**Enc**yclopedia **O**f **D**NA **E**lements) project, from a region of length $2^{17}(\approx 131k)$ base pairs from chromosome 1 (hg19 chr1:880001-1011072). Panel (a) shows counts (summed across two replicate experiments). Due to over-plotting, darker regions of the plot correspond to higher concentrations of data points. Panel (b) shows the estimated intensity function from *smash* (black solid line) and location of peaks called by MACS (red markers beneath the estimated intensity).

CHAPTER 3

ESTIMATING DIFFERENCES BETWEEN MULTIPLE SAMPLES[1]

3.1 Overview

While estimating sequencing profile for a single sample is an interesting problem in itself, a fundamental goal in biological applications is to infer differences between groups of samples. For example, detecting differential expression between groups has been an important goal in genomics for many years, and remains an area of active research today. While such studies are typically conducted using gene-level data (from both traditional microarrays and modern NGS platforms), the availability of NGS data allows one to potentially infer differences at much finer resolutions than a gene-level analysis. Nevertheless, estimating and testing for differences between groups at the single-base resolution remains a challenging task, mostly due to the small number of counts associated with such types of data (and the cost of obtaining higher read counts). Extending the idea of a multiscale representation from *smash*, we present a model-based approach to infer differences between sequencing profiles for multiple groups of samples.

Specifically, suppose we have sequencing data $Y_b^i, b = 1, ..., B, i = 1, ..., m$ measured over a region of length B for m samples. In addition, associated with each sample is a covariate g^i, which can be categorical (e.g. control vs treatment) or

1. Work in this chapter is developed in collaboration with Heejung Shim.

quantitative (e.g. genotype information). We then extend (2.1) to become

$$\begin{aligned} Y_b^i &\sim \text{Poi}(\lambda_b^i) \\ \log(\lambda_b^i) &= \mu_b^o + \beta_b^o g^i + v_b^i \end{aligned} \tag{3.1}$$

where μ_b^o denotes the intercept and β_b^o the slope as in a standard Poisson Generalized Linear Model (GLM). We make the assumption that both μ_b^o and β_b^o are spatially-structured. Note that an additional random effect v_b^i is added to model any extra-Poisson biological variation. Our primary interest lies in estimating β_b^o, which is the difference between the two groups if g is categorical with two levels, or the effect of a unit change in g on $\log(\lambda)$ if g is quantitative.

Motivated by the multiscale idea outlined in 2.1, we apply the same reparametrization described in 2.1 to the observations in each sample to obtain α_j^i for each sample i, and model the data in the multiscale space instead:

$$Y_{1:B}^i \sim \text{Poi}(\lambda_{1:B}^i) \tag{3.2}$$

$$Y_{1:(B/2)}^i | Y_{1:B}^i \sim \text{Bin}(Y_{1:B}^i, \text{logit}^{-1}(\alpha_2^i)) \tag{3.3}$$

$$... \tag{3.4}$$

$$Y_{B-1}^i | Y_{(B-1):B}^i \sim \text{Bin}(Y_{(B-1):B}^i, \text{logit}^{-1}(\alpha_B^i)) \tag{3.5}$$

$$\log(\lambda_{1:B}^i) = \mu_1 + \beta_1 g^i + u_1^i \tag{3.6}$$

$$\alpha_j^i = \mu_j + \beta_j g^i + u_j^i \qquad (j = 2, ..., B) \tag{3.7}$$

where μ_j and β_j represent the intercept and slope respectively in the multiscale

space, and u_j^i is the random effect in the multiscale space. Note that the model (3.2)-(3.7) is identical to (3.1), while also being a mixed effects logistic model, thus making it easy to fit.

In order to apply the shrinkage procedure *ash*, we need to obtain estimates and standard errors for both $\boldsymbol{\mu} = (\mu_1, ..., \mu_B)$ and $\boldsymbol{\beta} = (\beta_1, ..., \beta_B)$, similar to the univariate case (where we applied *ash* to the α_j's directly). While fitting a mixed effects logistic model to (3.2)-(3.7) does provide the desired quantities, there are three major modifications we have make for this procedure to work in practice:

1. If g is a two-level categorical variable (the more commonly seen scenario), we modify $\hat{\alpha}_j$ and $se(\hat{\alpha}_j)$ as described in Appendix A.1.1, and estimates and standard errors for μ_j and β_j can be computed analytically from $\hat{\alpha}_j$ and $se(\hat{\alpha}_j)$ when assuming an asymptotic Gaussian likelihood (see Appendix A.3.2 for details). Otherwise, if g is quantitative, we simply add pseudocounts to the Binomial successes and failures whenever they are 0 and fit a logistic model.

2. While the most natural way to impose shrinkage is to apply *ash* separately to $\boldsymbol{\mu}$ and $\boldsymbol{\beta}$ (for each scale) i.e. set independent priors for these two quantities, the likelihoods for μ_j and β_j do not necessarily factorize for any j, thereby inducing dependence in the posterior quantities, which is problematic when reconstructing μ_b^o and β_b^o. As such, we reparametrize the pairs (μ_j, β_j) as described in Appendix A.3.1 so that the posteriors are independent.

3. Fitting a mixed effects model is in general much slower than a fixed effects model, and even more so for logistic regression. As such, we make an approx-

imation by fitting a quasibinomial model instead (see McCullagh and Nelder (1989)), capturing the random effect by an overdispersion factor.

With these modifications we can then apply *ash* to obtain posterior means and variances for the μ_j's and β_j's. Estimating μ_b^o and β_b^o from μ_j and β_j is conceptually similar to estimating λ_b from α_j as in Chapter 2, albeit with differences in the approximations used. Full details for the reconstruction procedure are given in Appendix A.3.3-A.3.5 for various different scenarios.

In practice, the implementation of our method outlined above, which we call *Multiseq*, involves two additional features which greatly enhances the efficiency and accuracy of the method:

1. As with the univariate case (involving only 1 sample), we actually apply the TI version of the multiscale transform to each sample, as this improves estimation accuracy of both μ_b^o and β_b^o over the non-TI version.

2. An alternative to fitting the logistic model (3.2)-(3.7) is to approximate this by a running a Weighted Least Squares (WLS) regression instead, replacing the Binomial errors by Gaussian errors. This is conceptually very similar to truncating the iterative algorithm used to fit the logistic model at the first iteration, greatly enhancing computational speed without significant loss in accuracy. Details for implementing the WLS approach can be found in Appendix A.3.6.

An additional practical concern to note is that different samples in the same sequencing experiment can have different sequencing depths, which we define here

as the total number of reads mapped. This means that the total number of DNA fragments sequenced for each sample can be different, so that possible differences detected by *Multiseq* can in fact be due to sequencing depth, rather than actual biological mechanisms. In this case, we model the sequencing reads as

$$\begin{aligned} Y_b^i &\sim \text{Bin}(s^i, \lambda_b^i) \\ \log(\lambda_b^i) &= \mu_b^o + \beta_b^o g^i + v_b^i \end{aligned} \tag{3.8}$$

where s^i is the sequencing depth for sample i. Correspondingly, we modify (3.2)-(3.7) to become

$$Y_{1:B}^i \sim \text{Bin}(s^i, \lambda_{1:B}^i) \tag{3.9}$$

$$Y_{1:(B/2)}^i | Y_{1:B}^i \sim \text{Bin}(Y_{1:B}, \text{logit}^{-1}(\alpha_2)) \tag{3.10}$$

$$... \tag{3.11}$$

$$Y_{B-1} | Y_{(B-1):B} \sim \text{Bin}(Y_{(B-1):B}, \text{logit}^{-1}(\alpha_B)) \tag{3.12}$$

$$\text{logit}(\lambda_{1:B}^i) = \mu_1 + \beta_1 g^i + u_1^i \tag{3.13}$$

$$\alpha_j^i = \mu_j + \beta_j g^i + u_j^i \qquad (j = 2, ..., B) \tag{3.14}$$

, using the same implementation that fits (3.7) for the corresponding logistic models.

In addition to producing posterior estimates of the effect β_j^o, we can also perform a significance test for any given region. Specifically, we wish to test H_0: $\beta_b^o = 0 \, \forall b$ against H_A: $\exists b$ s.t. $\beta_b^o \neq 0$. Since *Multiseq* models the effects using a multiscale transformation, we use the multiscale parameters β_j instead, where β_j is assumed to

be a mixture of a point mass and several Gaussians in *ash*. By defining π_j^0 to be the mixture proportion for the point mass component for β_j, the equivalent formulation in multiscale space is H_0: $\pi_j^0 = 1\ \forall j$, and H_A: $\exists j$ s.t. $\pi_j^0 \neq 1$. To test this null hypothesis, we use the likelihood ratio statistic

$$\Lambda(\boldsymbol{D}) = \frac{P(\boldsymbol{D}|\hat{\boldsymbol{\pi}}^0)}{P(\boldsymbol{D}|\boldsymbol{\pi}^0 = \mathbf{1})} \tag{3.15}$$

where the "data" $\boldsymbol{D}$ is represented in multiscale space as $\hat{\beta}_j$ and $se(\hat{\beta}_j)$ (with a Gaussian approximation for the distribution of $\hat{\beta}_j$), and $\hat{\boldsymbol{\pi}}^0$ are the maximum likelihood estimates for the null mixture proportions $\boldsymbol{\pi}^0$. As with *smash*, we apply *ash* to the $\hat{\beta}_j$'s and $se(\hat{\beta}_j)$'s in the same scale to obtain a likelihood ratio statistic for that scale, and then multiply the likelihood ratio statistics across scales to obtain $\Lambda(\boldsymbol{D})$, making use of the assumption that the β_j's are independent across scales.

In practice, we have made several approximations in computing the likelihood $\Lambda(\boldsymbol{D})$, and the estimates $\hat{\boldsymbol{\pi}}^0$ are computed by maximizing a pseudo-likelihood rather than the true likelihood (Stephens (2016)). Hence, it is inadvisable to approximate $-2\log(\Lambda)$ using a χ^2 distribution. Instead, we can assess the significance of the likelihood ratio statistic by using its empirical distribution under the null, as is done in Shim and Stephens (2015).

3.2 Simulation results

To assess the performance of *Multiseq*, we run it on both simulated and real sequencing data. For simulated data, we compare *Multiseq* mostly against methods

designed to detect differences at the gene level rather than at the base pair level, which we call "window-based" methods. In our simulation studies, window-based methods consist of *edgeR*, *DESeq* and *DESeq2* (Love et al. (2014)). To apply these window-based methods in practice, we partition the regions into bins and aggregate the counts in each bin, before using the aggregated counts as inputs. To make the simulations realistic, we make use of sequencing data from two different projects to set up two different schemes. For data generated from the DNase-seq experiment, we also compare *Multiseq* to *WaveQTL* (Shim and Stephens (2015)). *WaveQTL* is not included for data generated from the ChIP-seq experiment because the Gaussian approximations used makes it unsuitable for such small sample sizes.

1. For the first scheme, we use publicly available ChIP-seq data (hg19) available as part of the ENCODE project. The reads measure binding of transcription factor YY1 in two cell lines: GM12878, a lymphoblastoid cell line (LCL), and H1hesc, which are embryonic stem cells. There are two replicates for each cell line, resulting in a total of 4 samples. In our simulations, we also simulate data for 4 samples, with two samples in each of the two groups. To start off, we ran MACS on non-overlapping regions of 2^{15}bp. We then selected regions which had peak(s) present in either of the cell lines, as determined by MACS. The location of this peak will be where we simulate an "effect" β_b^o (difference between groups) for the corresponding region. In particular, we used a constant two-fold (relatively weak and thus hard to detect) difference at the peak location, with no difference in the mean profiles for the two groups elsewhere. Data were simulated for regions of varying lengths (2^{11}bp, 2^{13}bp and

2^{15}bp), where the smaller regions are simply truncated versions of the largest region (2^{15}bp), while still keeping the site of interest within the region. The baseline mean profile μ_b^o will be generated based on two different procedures, as outlined below. 117 regions were eventually chosen, with varying intensities for the original peaks detected by MACS.

2. For the second scheme, we use DNase-seq data from Pique-Regi et al. (2011), which was originally intended to identify dsQTLs. The data consist of reads from 70 HapMap Yoruba LCLs. In our simulations, we simulate data for varying sample sizes (4, 6, 10, 30 and 70) in two different groups, with an equal number of samples in each group. 578 regions of length 1024bp were chosen based on the analysis scheme in Shim and Stephens (2015), which targeted sites with the highest DNase I sensitivity. Shim and Stephens (2015) considered different region lengths in greater detail, so we do not consider the effect of region size for this simulation scheme. The "effect" β_b^o for each region was generated by using the effect size estimate from running *Multiseq* on all 70 samples, where the covariate g is the genotype at the most strongly associated genetic variant for that region. We also generate an additional β_b^o using the smoothing procedure *bayesthr* from the R package *wavethresh*, and include results below. As with the first scheme, the baseline mean profile μ_b^o will again be generated using the two procedures we will now describe.

For each of the two simulations schemes described (which are based on two different datasets with different shapes for the effect), we further explore two different data generation procedures, so as to test the robustness of *Multiseq* as well as the window-

based approaches:

1. A Negative Binomial generation process. This data generation process is based exactly on the model (3.2)-(3.7) that we start off from. Hence, this acts as more of a sanity check to make sure that *Multiseq* behaves as we expect. To generate reads, both a baseline signal μ_b^o and an effect size signal β_b^o are needed. In the simulations for the ChIP-seq data, the baseline signal μ_b^o for each of the 117 regions was generated by running *Multiseq* on the original data from that region, using the original cell lines as group indicators. We then generated a null case and an alternative case for each region. The 4 samples in the null case were all simulated according to

$$Y_b^i \sim \text{NB}(e^{\mu_b^o}, \nu) \tag{3.16}$$

for some pre-specified dispersion parameter ν, where ν is parametrized such that $\text{Var}(Y_b^i) = e^{\mu_b^o} + \nu e^{2\mu_b^o}$. In the alternative case, two of the samples were simulated according to (3.16), while the other two samples were simulated according to

$$Y_b^i \sim \text{NB}(e^{\mu_b^o + \beta_b^o}, \nu) \tag{3.17}$$

. As noted previously, β_b^o is everywhere 1 except for the site where a peak was detected by MACs, where it takes the value 2. For the DNase-seq data, we again generate a null case and an alternative case for each of the 578 regions. All samples (which can vary in number) in the null case were simulated according to (3.16), where the baseline signal μ_b^o is taken to be the baseline estimate from

running *Multiseq* on all 70 samples in the original data, and the covariate g is the genotype at the most strongly associated genetic variant for that region. For the alternative cases, half the samples were simulated according to (3.16), and the other half according to (3.17), where the effect β_b^o is generated in the procedure described above.

2. A Beta Binomial thinning process. For this procedure, our goal is to produce more realistic simulations that can test the robustness of *Multiseq*, where the data do not necessarily follow the generative process (3.1). In the simulations for the ChIP-seq data, we first pool the reads from all 4 samples in the original data. Given the pooled reads at each base b, denoted by $\tilde{Y}_b$, we generate the Y_b^i by sampling from $\text{Bin}(\tilde{Y}_b, p_b^i)$, where p_b^i is the Binomial "success" probability. To introduce sample-specific variation into the simulation scheme, we further sample p_b^i from a Beta distribution characterized by its mean and dispersion parameter. Again, a null case and an alternative case were generated for each of the 117 regions. For the null cases, we set $E(p_b^i) = 1/4$, so that each sample is the same on average. For the alternative cases, we set $E(p_b^i) = 1/4$ for $i = 1, ..., 4$ and $b \notin S$, where S is the site where we simulated an artificial effect as described above. We then set $E(p_b^i) = 1/3, b \in S$ for the two samples in one of the groups, and $E(p_b^i) = 1/6, b \in S$ for the other two samples. This is done to ensure that the two groups differ by a factor of two (on average) at the site of interest, and no difference (on average) everywhere else, while enforcing the constraint $\sum_{i=1}^4 p_b^i = 1 \, \forall b$. In practice, we switch the direction of the effect at random i.e. a $\log_2$ difference of either 1 or -1. This is done to

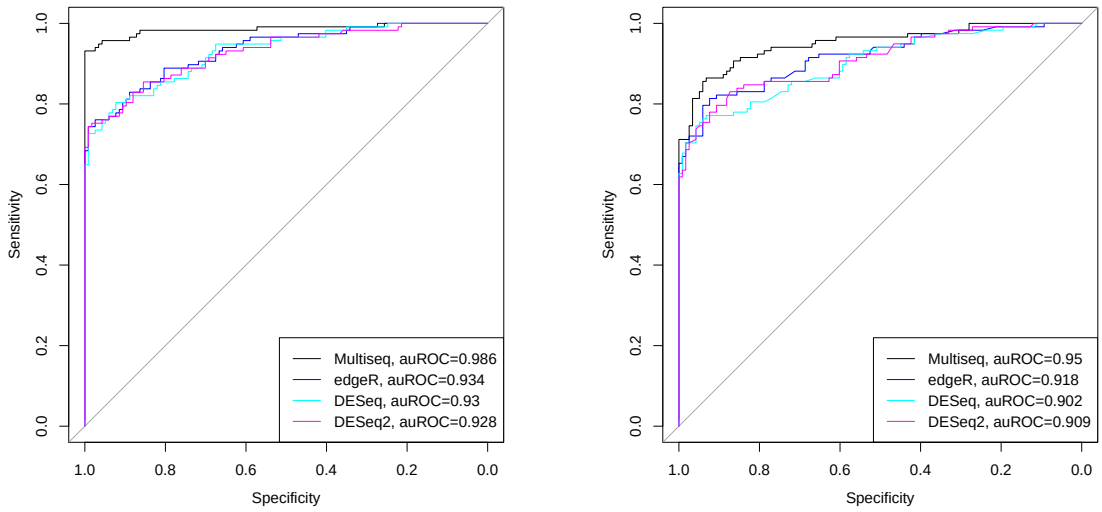
ensure that the overall read count is relatively constant across the 4 samples. The simulations for the DNase-seq data follows a similar procedure. We again pool the reads from all 70 samples in the original data to obtain $\tilde{Y}_b$, and then generate the reads for each sample in the simulated dataset by sampling from $\text{Bin}(\tilde{Y}_b, p_b^i)$, where p_b^i is itself generated from a Beta distribution. A null case and an alternative case are generated for each of the 578 regions. For the null cases, we set $E(p_b^i) = 1/70$. For the alternative cases, we set $E(p_b^i)$ so that the log difference (on average) between the two groups is exactly the effect β_b^o estimated using *Multiseq* or *bayesthr* (see above), subject to the constraint that $\sum_{i=1}^m p_b^i = \frac{m}{70}\ \forall b$, where $m \in \{4, 6, 10, 30, 70\}$ is the desired sample size in the simulations.

For convenience, we will refer pairings of the data sources and the generation processes as ChIP-NB (ChIP-seq data with the Negative Binomial generation scheme), ChIP-BB (ChIP-seq data with the Beta Binomial generation scheme), DNase-NB (DNase-seq data with the Negative Binomial generation scheme) and DNase-BB (DNase-seq data with the Beta Binomial generation scheme) respectively.

We now present results for all of the simulation schemes described above. Since the actual presence or absence of an effect is known for each region in a simulation scheme, we assess the performance of the methods based on area under the receiver operating characteristic (auROC) curve, with a higher auROC indicating better performance. To obtain the auROCs, what we need is a ranking based on some form of statistic for any given method. For *Multiseq*, the log-likelihood ratio statistic in (3.15) provides a ranking, with higher values indicating more evidence against the

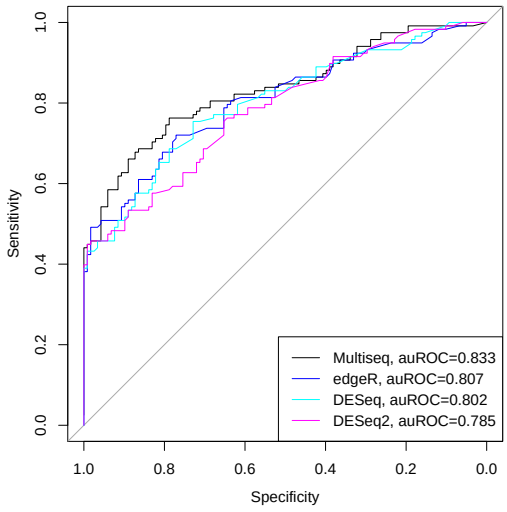
null. A similar log-likelihood ratio statistic can be used for *WaveQTL* (see Shim and Stephens (2015) for more details). For window-based approaches, a p-value testing for differences between groups is provided for each bin within a region. We then use the minimum p-value across all bins within each region as the statistic for that region, with a smaller p-value indicating more evidence against the null.

Figures 3.1 and 3.2 show the receiver operating characteristic (ROC) curves for ChIP-NB and ChIP-BB respectively, while Figures 3.3 and 3.4 show results for ChIP-NB and ChIP-BB using only *bayesthr* to estimate the effect sizes. Results using *Multiseq* to estimate the effect sizes are mostly similar, and can be found in Appendix A.5. Note that the window-based methods for ChIP-NB and ChIP-BB all use a constant window size of 300bp, as the sites where effects were generated were all around 300bp. On the other hand, we used window sizes of 100, 300 and 1024 bp for DNase-NB and DNase-BB, as different choices of window sizes can lead to different results due to the irregular nature of the effects generated under these simulation schemes. To avoid clutter, we show results only for *DESeq* for DNase-NB and DNase-BB. Results for *DESeq*2 and *edgeR* are quite similar.



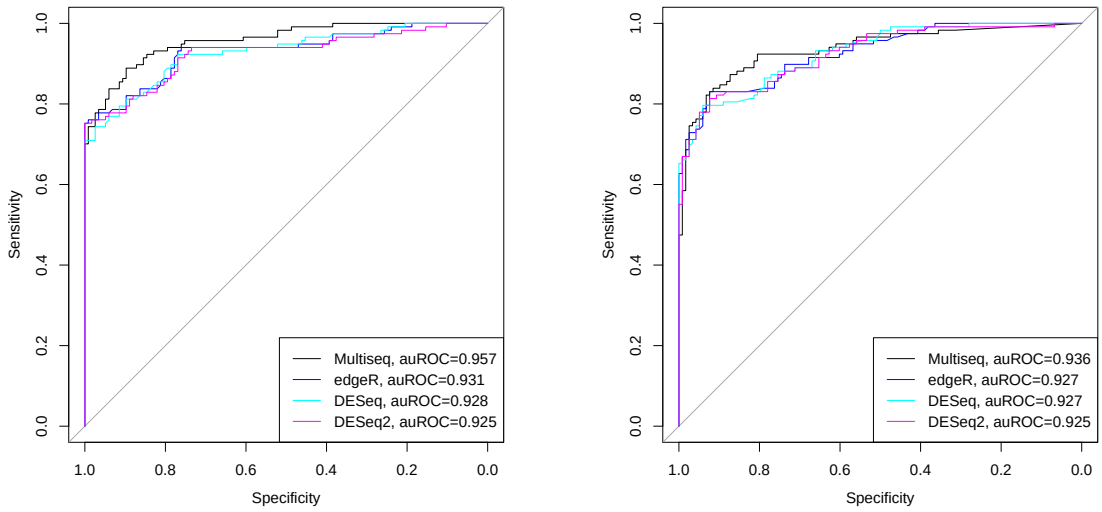
(a)

(b)



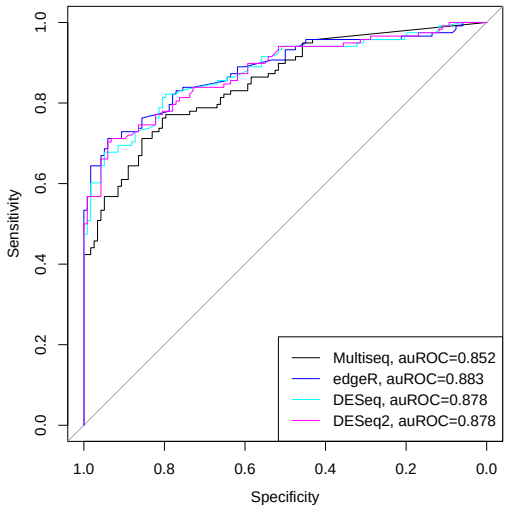
(c)

Figure 3.1: ROC curves and areas under the curves for ChIP-seq-based simulations under the Negative Binomial generation scheme. For each of the 117 regions, one null case was simulated in which an effect is absent in the 4 samples, and one non-null case was simulated in which an effect was present in the 4 samples. Methods being compared include *Multiseq*, *edgeR*, *DESeq* and *DESeq*2. All window-based approaches used 300bp windows, which is close to the width of the effect being simulated. Panels (a), (b) and (c) show results when the region sizes are 2^{11}bp, 2^{13}bp and 2^{15}bp respectively.



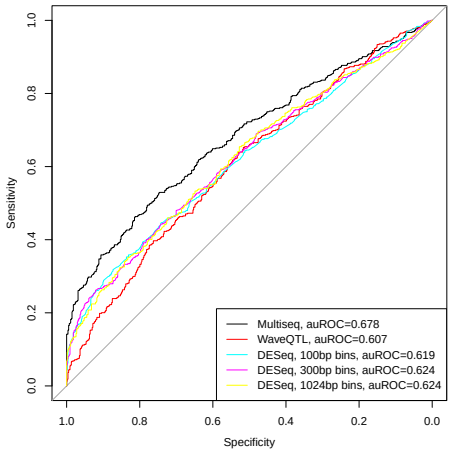
(a)

(b)

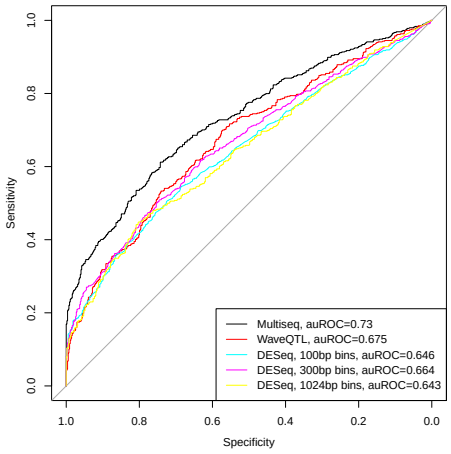


(c)

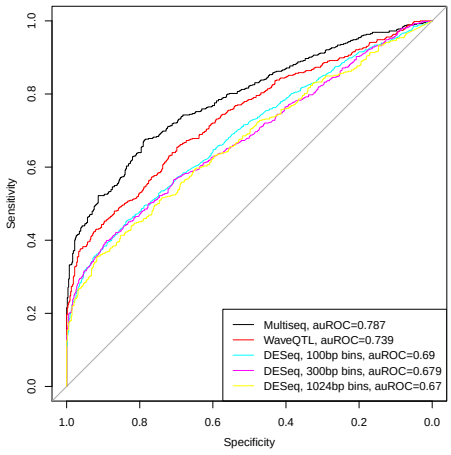
Figure 3.2: ROC curves and areas under the curves for ChIP-seq-based simulations under the Beta Binomial generation scheme. For each of the 117 regions, one null case was simulated in which an effect is absent in the 4 samples, and one non-null case was simulated in which an effect was present in the 4 samples. Methods being compared include *Multiseq*, *edgeR*, *DESeq* and *DESeq*2. All window-based approaches used 300bp windows, which close to the width of the effect being simulated. Panels (a), (b) and (c) show results when the region sizes are 2^{11}bp, 2^{13}bp and 2^{15}bp respectively.



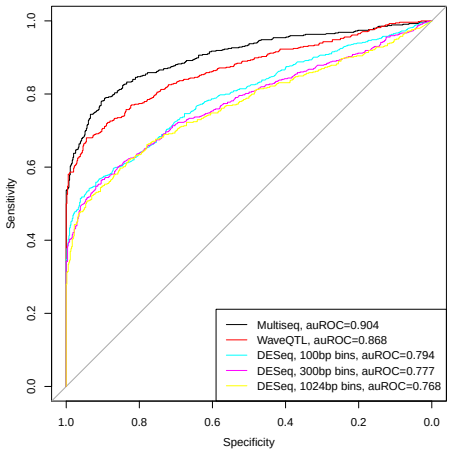
(a)



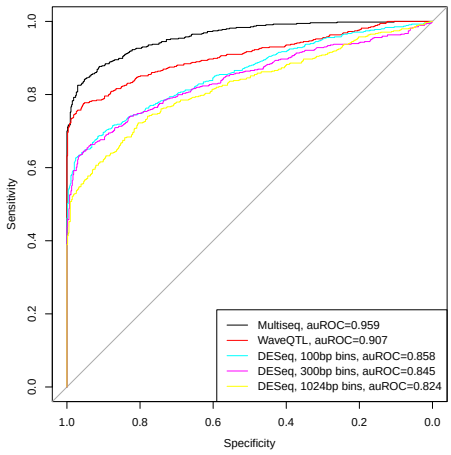
(b)



(c)

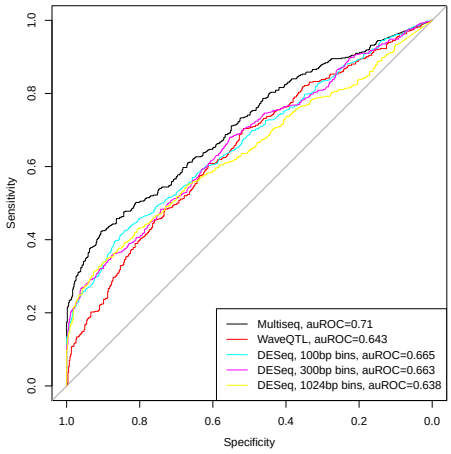


(d)

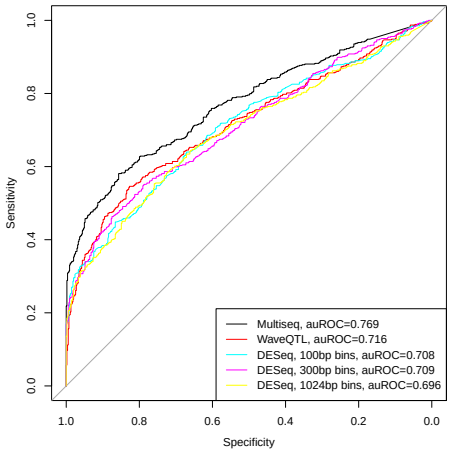


(e)

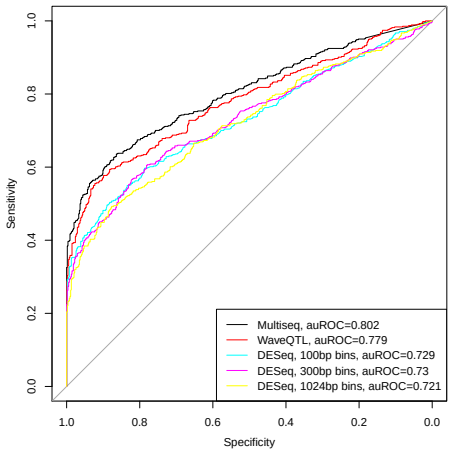
Figure 3.3: ROC curves and areas under the curves for DNase-seq-based simulations under the Negative Binomial generation scheme, where the simulated effect is estimated from raw data using *bayesthr*. For each of the 578 regions, one null case was simulated in which an effect is absent, and one non-null case was simulated in which an effect was present. Methods being compared include *Multiseq*, *WaveQTL*, and *DESeq* with various choices of window sizes. Panels (a), (b), (c), (d) and (e) show results when the sample sizes are 4, 6, 10, 30 and 70 respectively.



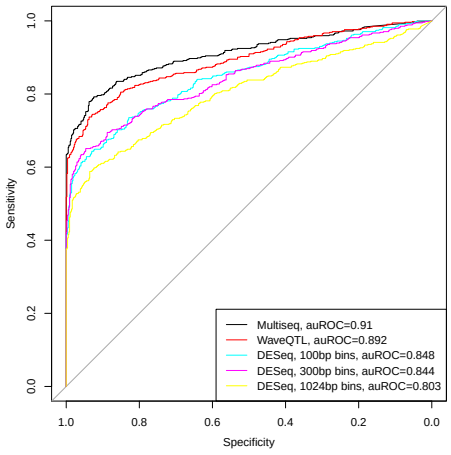
(a)



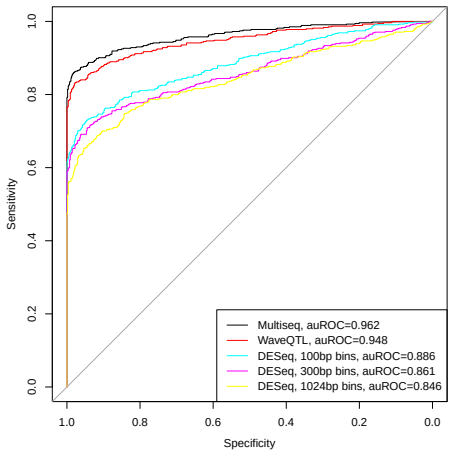
(b)



(c)



(d)



(e)

Figure 3.4: ROC curves and areas under the curves for DNase-seq-based simulations under the Beta Binomial generation scheme, where the simulated effect is estimated from raw data using *bayesthr*. For each of the 578 regions, one null case was simulated in which an effect is absent, and one non-null case was simulated in which an effect was present. Methods being compared include *Multiseq*, *WaveQTL*, and *DESeq* with various choices of window sizes. Panels (a), (b), (c), (d) and (e) show results when the sample sizes are 4, 6, 10, 30 and 70 respectively.

For ChIP-NB and ChIP-BB, we see that *Multiseq* solidly outperforms window-based methods for moderate region sizes of 2^{11}bp and 2^{13}bp, with 2^{11}bp being more favorable for *Multiseq*. When the region size is 2^{15}bp however, *Multiseq* outperforms window-based methods for ChIP-NB, but not ChIP-BB. The general pattern is that while the performance of all the methods decrease with increasing region size, *Multiseq* is the most severely impacted, while window-based methods are less influenced thus. We also note that the relative performance gain of *Multiseq* over window-based methods is greater for ChIP-NB than ChIP-BB in general. These behaviors will be explored more in Section 3.4, although we note here that the relatively poor performance of *Multiseq* for ChIP-BB when the region size is 2^{15}bp can be circumvented in practice. For one, this particular scenario rarely resembles what actual sequencing data looks like, and even if certain isolated regions do indeed look like this, we can avoid this issue by running the method on multiple regions across the entire genome to learn model parameters. More importantly, it is also worth remembering that the window size chosen for the window-based methods almost exactly matches the width of the effects being simulated; in practice, it is difficult to know the exact window size to choose, and the appropriate window size likely varies along the genome. As such, the excellent overall performance of *Multiseq* despite the simulation schemes being favorable to window-based methods is reassuring.

DNase-NB and DNase-BB, on the other hand, are inherently less favorable to window-based methods. Although we used a variety of different window sizes for window based methods, the nature of the estimated effects used to generate the reads renders any single window size ineffective, and this behavior likely holds for

actual sequencing data. The inability of window-based methods to adapt to the data is reflected in the auROCs, where *Multiseq* substantially outperforms window-based methods in all scenarios, even more so for larger sample sizes. We also note that *Multiseq* also outperforms *WaveQTL*, particularly for small sample sizes, as the latter method is not suited to small sample sizes in the first place. Overall, the relative performance of the methods are similar between DNase-NB and DNase-BB, with all methods performing better for larger sample sizes, as is expected.

These simulations studies serve to demonstrate the advantages of using multi-scale methods, particularly *Multiseq*, over window-based methods when used to find differences between two groups. In practice, real effect sizes are more like the ones in DNase-NB and DNase-BB than ChIP-NB or ChIP-BB, so that we expect *Multiseq* to have much more power when testing for non-zero effects, when compared with approaches that were originally designed to test for differences for units much larger than base pairs (e.g. genes).

3.3 Case study: Application to RNA-seq data for gene *OAS1*

While we have focused our applications and simulations on sequencing data that aim to understand gene regulation (ChIP-seq and DNase-seq) so far, RNA-seq is a powerful alternative to the traditional microarrays for measuring gene expression, possibly revealing genetic aspects of certain diseases that are not directly observable. In Chapter 4 we will attempt to cluster samples via RNA-seq measurements, potentially establishing complex relationships between different subgroups. Here we

set up a precursor to the clustering analysis by first running *Multiseq* on the same dataset that will be used in Section 4.3.

Specifically, we apply *Multiseq* to RNA-seq data from a well known study (Pickrell et al. (2010)). In that study, RNA-seq reads were obtained from lymphoblastoid cell lines (LCLs) derived from a population of 69 Yoruba individuals that were studied as part of the International HapMap project. In particular, one of the findings in Pickrell et al. (2010) was the mechanisms through which a splicing QTL (sQTL) rs10774671 affects the expression level in a portion of the *OAS1* gene (hg19 chr12:113344582-113371027), a gene that controls the synthesis of a family of proteins that are involved in innate antiviral response.

To run *Multiseq*, the counts matrix $\boldsymbol{Y}$ consists of RNA-seq reads from 66 of the 69 individuals which did not have missing genotype information. The covariate g is the genotype for each individual at SNP rs10774671, encoded as a quantitative variable with values 0, 1 or 2. Figure 3.5 shows the raw data as well as the effect estimated from *Multiseq*. For clarity of presentation, part of the intronic region between the third and fourth exon has been removed from the plot as indicated by the dotted blue vertical line, since they add no new information. The raw data shows obvious linear trends as we move from the major to the minor homozygote, which justifies our use of genotype information as a quantitative covariate. The estimated effect size agrees well with the results reported in Pickrell et al. (2010), where rs10774671 affects expression level in the intronic region between the fifth and the sixth exon, as well as the sixth exon itself. In particular, we can also observe the splicing junction that was absent from public databases at the time which the study in Pickrell et al.

(2010) was conducted.

While this is a simple illustration of the potential uses of *Multiseq*, a formal analysis of differential chromatin accessibility (measured by ATAC-seq data in this case) between control and treatment conditions using *Multiseq* can be found in Shim et al. (in preparation). This illustration also serves to show that *Multiseq* can be used as an initial exploratory tool in clustering analysis to determine the regions that are interesting enough to run clustering methods on, although the usefulness of *Multiseq* in this context would be limited by its inclusion of a single covariate in the model.

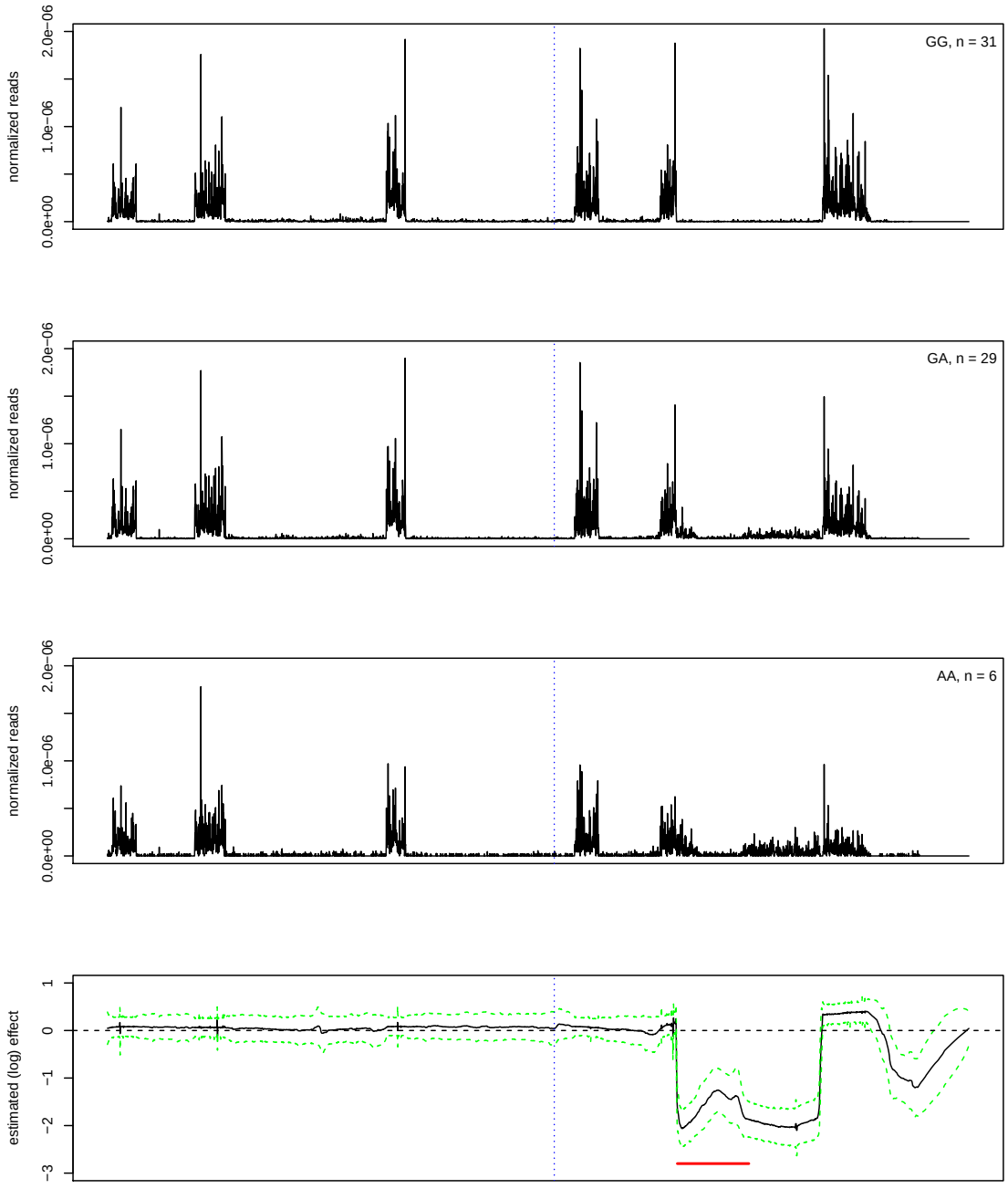


Figure 3.5: An illustration where *Multiseq* is applied to the RNA-seq data for the *OAS1* gene in 69 Yoruba individuals from the International HapMap project. 3 individuals had missing genotype information for the SNP of interest (rs10774671) and were removed. Top three plots show the average sequencing profiles for the three different genotype classes, normalized by their sequencing depth. The bottom plot shows the estimated effect size using *Multiseq* (solid black), as well as the posterior standard deviations (dotted green). The red bar in the bottom plot indicates the region where the splicing junction was discovered by Pickrell et al. (2010). In all four plots, the vertical dotted blue lines mark the boundary where a segment of the gene was excluded (see main text) for easier visualization.

3.4 An alternative model to estimate effect sizes

While the simulations conducted in Section 3.2 show that *Multiseq* performs reasonably well on a wide range of simulated data, one caveat we noted is that *Multiseq* tends to perform worse than window-based approaches when the data is simulated from a Beta Binomial scheme from the ChIP-seq experiment, when the region size is large (e.g. 2^{15}bp). Figure 3.6 shows an example of how the *Multiseq* estimates change when we increase the region size for a particular set of simulations on a single region from the Beta-Binomial ChIP-seq simulations described in Section 3.2. Since we only focus on one non-null region where an effect is actually present, 100 independent runs were simulated to obtain an average effect estimate.

While the reason for this behavior is not immediately obvious, we believe that there are two factors contributing to this:

1. The data simulated under the Beta Binomial procedure tend to exhibit larger extra-Poisson variation than the data simulated under the Negative Binomial scheme (and in fact, larger than the values estimated from the raw data as well). For example, the dispersion parameter (according to a Negative Binomial model) estimated by *edgeR* for simulated data from ChIP-BB is about twice the value estimated for data simulated from ChIP-NB. At the same time, because it is not immediately clear what the true baseline under the Beta Binomial scheme is, it could be highly irregular and thus not really "spatially structured". These two factors combined makes it much harder to estimate both the baseline as well as the effect, which is not very strong in the first place (only a two-fold difference). On the other hand, the data generated under the Negative

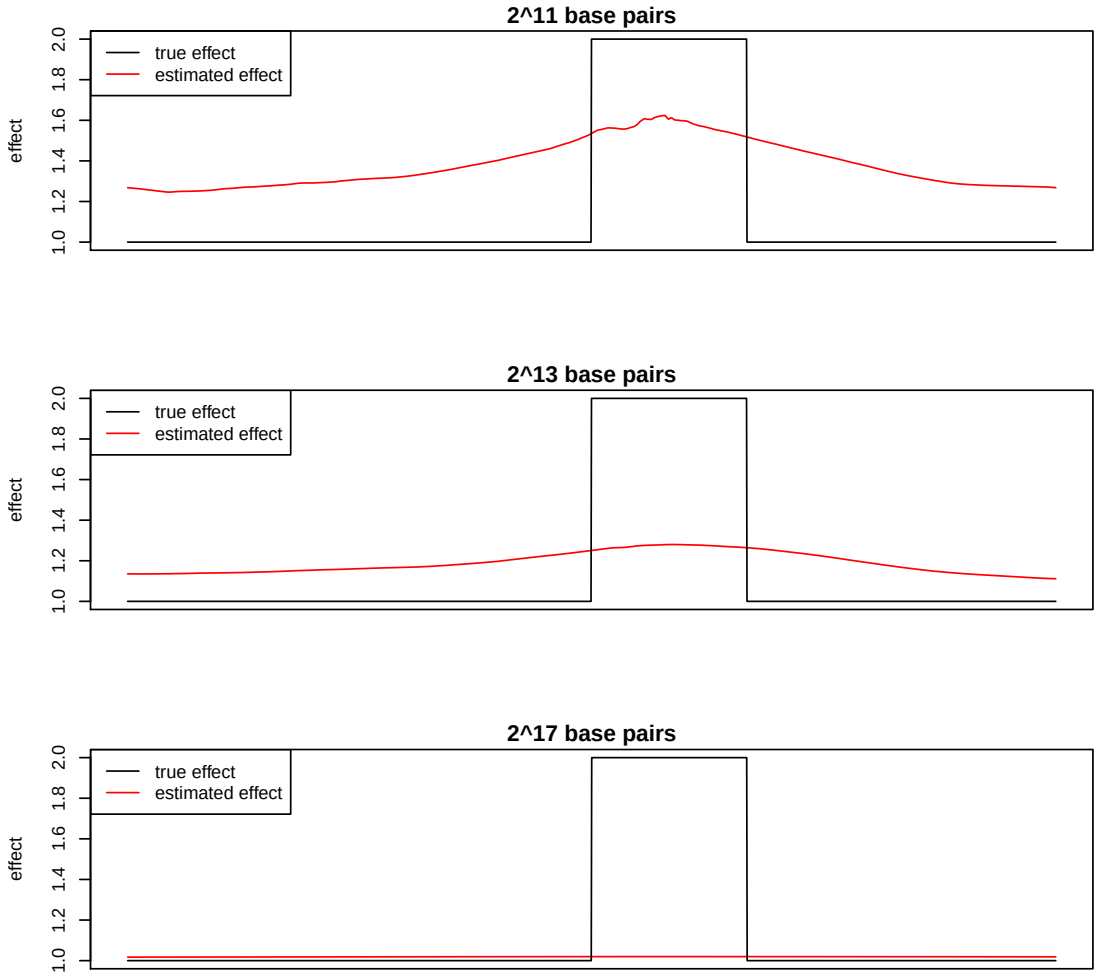


Figure 3.6: An example where *Multiseq* was applied to ChIP-seq-based simulations under the Beta Binomial simulation scheme corresponding to one of the MACS peaks. The three plots show results for region sizes of 2^{11}bp, 2^{13}bp and 2^{15}bp respectively. For each region size, 100 independent datasets were generated. In each of the plots, the black line corresponds to the true simulated effect size, while the red line corresponds to the average *Multiseq* estimate. Note that the results for all region sizes are zoomed in to the same resolution for easier visual comparison.

Binomial model follows the generative model for *Multiseq* and has comparatively smaller extra-Poisson variation, hence *Multiseq* performs substantially better than window-based approaches in this case.

2. Not only is the effect weak, it is also "sparse" in the sense that the effect is everywhere zero (on the log scale) except for a small site, which is tiny ($\approx$ 300bp) compared to the length of the entire region ($\approx$ 33000bp). Since the shrinkage procedure in *Multiseq* works independently for each scale, information from other scales are not taken into account. One can intuitively think of the sparse and weak effect being distributed across all the scales, so that each scale will contain very little information about the presence of the effect by itself. This translates to the *ash* hyperparameters π being estimated as having nearly all the mass on the null component, resulting in posterior estimates that often fail to pick up the effect.

Since window-based approaches use windows of the same size as the site where the effect is generated, they have a much easier time picking up the true effect, hence outperforming *Multiseq* in this particular scenario. While a combination of the two above-mentioned factors hinders the performance of *Multiseq*, we speculate that it would rarely be a problem in practice. First, actual sequencing data do not often exhibit such sparse and weak effects in large regions, and typically posses smaller extra-Poisson variations than the Beta Binomial scheme outlined above, making it significantly easier for *Multiseq* to pick up the effect. Second, the way *Multiseq* is intended to be used when applying it genome-wide is to estimate the *ash* hyperparameters from a number of regions, and use the estimated hyperparameters when

applying the method to other regions in the genome. This would greatly reduce the time *Multiseq* takes to estimate the hyperparameters, compared to estimating these hyperparameteres separately for each region. Furthermore, by estimating the hyperparameters from a multitude of regions, we expect the posterior estimates to be much less affected by regions that possess only a single weak effect. Of course, the downside to this practice would be that hyperparameters for any single region would not be estimated as accurately, so that the final call for how *Multiseq* is used should be made by the user.

Even though we speculate that this issue is mostly conceptual and of little import in practice, we seek to build upon *Multiseq* and explore alternative methods in the event that one might indeed be interested in finding a single weak effect in an isolated region of the genome. In particular, we wish to capture the "sparsity" effect using a chain-like structure. A natural model that arises often in practice is a Markov Chain, which is indeed what we propose. To be explicit, suppose we again have sequencing reads Y_b^i from multiple samples on the same region, as well as an associated covariate g. We formulate the model initially as in (3.1):

$$\begin{aligned} Y_b^i &\sim \text{Poi}(\lambda_b^i) \\ \log(\lambda_b^i) &= \mu_b^o + \beta_b^o g^i + v_b^i \end{aligned} \tag{3.18}$$

where the parameters have the same meaning as they do in (3.1). However, instead of assuming both μ_b^o and β_b^o to be spatially structured, we assume only μ_b^o to be spatially structured, but that β_b^o is "sparse", meaning that the effect of g is present

in only small portions of the region. This gives rise to a "step-like" structure to β_b^o, which we model by a Markov Chain. In particular, given the type of effects we assume, we use a special form of Markov Chain where we place constraints on the transition probabilities of β_b^o. Intuitively, if a certain base shows no effect from g, we would expect neighboring base pairs to show no effect as well. Similarly, base pairs in a region with an effect will tend to have similar effect sizes. Motivated by this idea, we consider a discretized version of a continuous-time Markov chain. For states $x_1, ..., x_K$, the transition probabilities are given by:

$$P(\beta_{b+1}^o = x_k | \beta_b^o = x_j) = \begin{cases} 1 - q + q\pi_k & k = j \\ q\pi_k & k \neq j \end{cases} \tag{3.19}$$

where the parameter $q \in [0, 1]$ represents a "jump probability", and the parameter vector $\pi = (\pi_1, \ldots, \pi_K)$ is a probability vector representing the stationary distribution of the Markov chain. One can immediately see the connection between the above formulation and that of a continuous-time Markov chain: instead of an exponential holding time in the continuous version, we have a discrete holding time with a Geometric(p) distribution. Furthermore, once the chain jumps, it jumps according to a transition probability matrix independent of the jump time. In this case, the embedded transition probability matrix A is given by

$$A_{jk} = \pi_k \quad j, k = x_1, ..., x_K \tag{3.20}$$

In words, at each step b the chain either stays in the current state with probability

$1-q$, or jumps with probability q. If a jump occurs, then β^o_{b+1} is drawn according to the distribution π. In the context of a "sparse" effect discussed earlier, we expect the vector π to have most of its mass on the state corresponding to 0 (i.e. no effect), so that $\beta^o_b = 0$ for most b. Not only does this formulation have a naturally appealing interpretation in our application, but it also has a computational advantage, namely that the forwards-backwards algorithm becomes $O(K)$ rather than $O(K^2)$.

While it would be ideal to jointly estimate μ^o_b and β^o_b, in practice it is far simpler and likely more efficient to first estimate μ^o_b, and then estimate β^o_b treating μ^o_b as known. As such, our method, which we call *HMM-seq*, first estimates the posterior means of μ^o_b using *Multiseq*. Given the posterior means $\hat{\mu}^o_b$, the emission probabilities are given by

$$P(Y^i_b = y|\hat{\mu}^o_b, \beta^o_b) = \frac{e^{-e^{\hat{\mu}^o_b+\beta^o_b}} e^{(\hat{\mu}^o_b+\beta^o_b)y}}{y!} \tag{3.21}$$

ie. $Y_b|\hat{\mu}^o_b, \beta^o_b \sim Pois(e^{\hat{\mu}^o_b+\beta^o_b})$. Hence, the emission probabilities can be computed explicitly, and need not be estimated. In practice, we use the Negative Binomial distribution as an overdispersed Poisson distribution, and include an addition dispersion parameter ν, which we estimate using *edgeR*.

Our goal is thus to estimate the unknown parameters π and q, given the emission probabilities (3.21) and the observations Y^i_b. We estimate these unknown quantities via maximum likelihood, as is common practice. Since the effects we are interested in can be really sparse, we include the option to add a prior component to the expected complete-data log-likelihood, and maximize the pseudo-likelihood instead. Various

details for implementing *HMM-seq* can be found in Appendices A.4.1-A.4.2.

Given the MLEs of q and π, denoted by $\hat{q}$ and $\hat{\pi}$, we can compute the marginal posterior distribution of β_b^o, $P(\beta_b^o = x_k|Y, \hat{q}, \hat{\pi})$. An effect size can then be estimated using any of the following three commonly-seen quantities, depending on the actual size and sparsity of the effect(s):

1. The posterior mean $E(\beta_b^o|Y, \hat{q}, \hat{\pi}) = \sum_k x_k P(\beta_b^o = x_k|Y, \hat{q}, \hat{\pi})$. This minimizes the L_2 distance between any estimate and the true effect, and is probably the best in general, as effects can vary in strength and shape.

2. The posterior mode $\underset{x_k}{\text{argmax}} P(\beta_b^o = x_k|Y, \hat{q}, \hat{\pi}), b = 1, ..., B$. This maximizes the expected number of correct individual states, and is suitable when the actual effect appears discrete.

3. The most likely state sequence $\underset{x_{1,k},...,x_{B,k}}{\text{argmax}} P(\beta_1^o = x_{1,k}, ..., \beta_1^o = x_{B,k}|Y, \hat{q}, \hat{\pi})$ among all sequences. While this a popular optimality criterion in many applications and might be a possible candidate with which we can test for the presence of an effect, it probably is not the most suitable quantity for actually estimating effect sizes. The Viterbi algorithm for computing the most likely state sequence is described in e.g. Rabiner (1989).

As with *Multiseq*, we can use *HMM-seq* to perform significance testing in addition to estimating effect sizes. Specifically, we wish to test H_0: $\beta_b^o = 0\ \forall b$ vs H_A: $\exists b$ s.t. $\beta_b^o \neq 0$. We use the log-likelihood ratio

$$\Lambda_{\text{HMM}}(\boldsymbol{Y}) = \frac{P(\boldsymbol{Y}|\hat{\boldsymbol{\mu}}^o, \boldsymbol{\beta}^o = \mathbf{0}, \hat{\nu})}{P(\boldsymbol{Y}|\hat{\boldsymbol{\mu}}^o, \hat{\boldsymbol{\beta}}^o, \hat{\nu})} \tag{3.22}$$

where $\boldsymbol{Y}$ is the matrix of sequencing counts, $\hat{\boldsymbol{\mu}}^o$ is the estimated baseline, $\hat{\boldsymbol{\beta}}^o$ is the estimated effect using one of the quantities above, and $\hat{\nu}$ is the estimated dispersion parameter. Note that

$$P(\boldsymbol{Y}|\hat{\boldsymbol{\mu}}^o, \boldsymbol{\beta}^o = \mathbf{0}, \hat{\nu}) = \prod_i \prod_b f(Y_b^i; \hat{\mu}_b^o, \beta_b^o = 0, \hat{\nu}) \tag{3.23}$$

$$P(\boldsymbol{Y}|\hat{\boldsymbol{\mu}}^o, \hat{\boldsymbol{\beta}}^o, \hat{\nu}) = \prod_i \prod_b f(Y_b^i; \hat{\mu}_b^o, \hat{\beta}_b^o, \hat{\nu}) \tag{3.24}$$

where f is the probability mass function for the Negative Binomial distribution.

Since the parameters are not estimated by maximum likelihood, we cannot guarantee an asymptotic χ^2 distribution for $-2\log(\Lambda_{\text{HMM}})$, if the asymptotics even hold in the first place. Instead, the empirical distribution for the statistics will have to be computed via numerical procedures such as permutation, similar to the procedures used for *Multiseq*. Quantities such as p-values can then be obtained to quantify the amount of evidence against H_0.

3.4.1 Simulation results

To assess the performance of *HMM-seq*, we run the method on the exact same sets of simulations presented in Section 3.2. To obtain auROCs for *HMM-seq*, we use the likelihood ratio statistics from (3.22) to rank the significance of the regions, with a higher likelihood ratio indicating more evidence of an effect being present in the associated region. Figures 3.7-3.10 show the same ROCs as in Figures 3.1-3.4, but with two extra ROCs for the results from *HMM-seq*, both with and without the "sparse" prior.

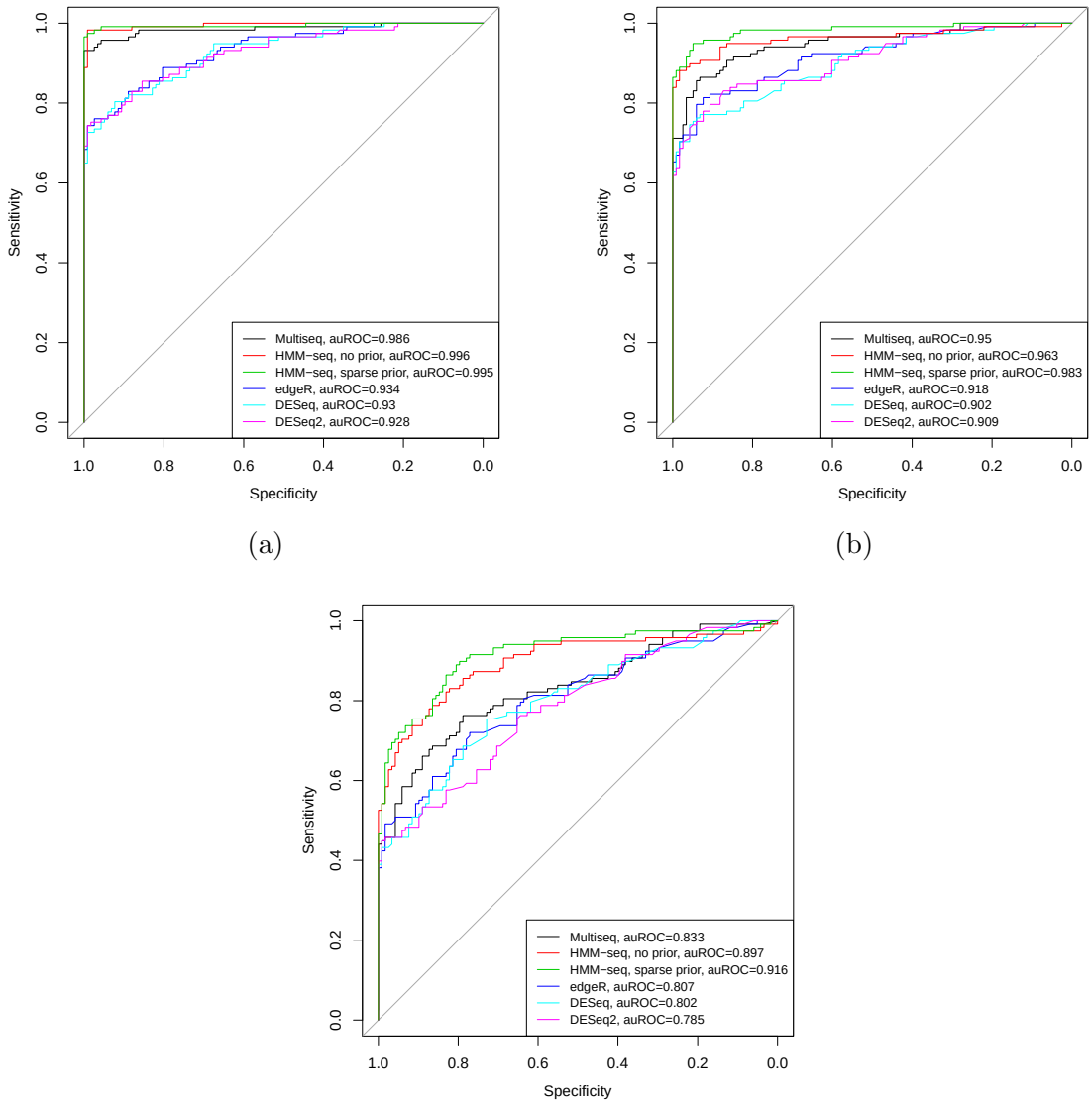
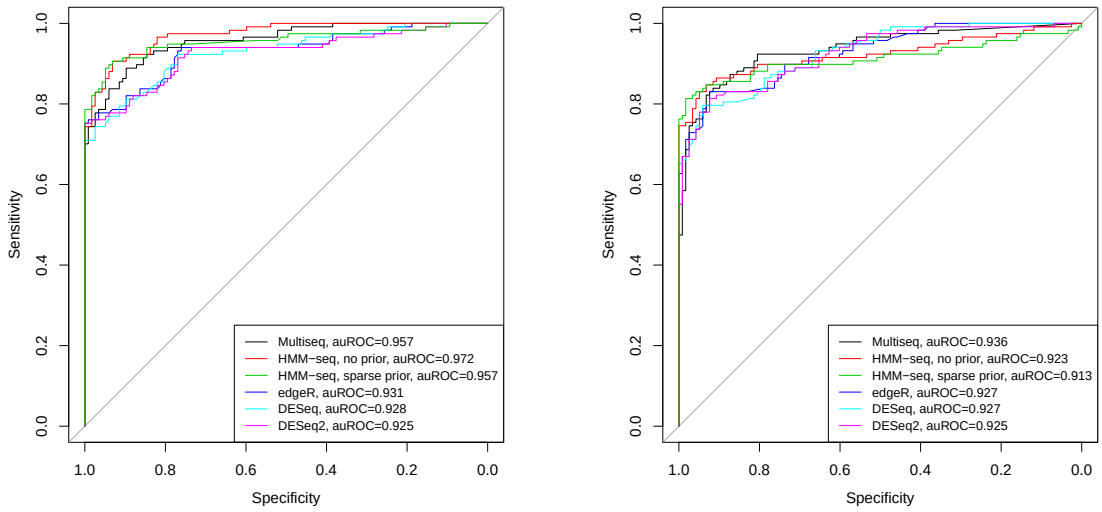
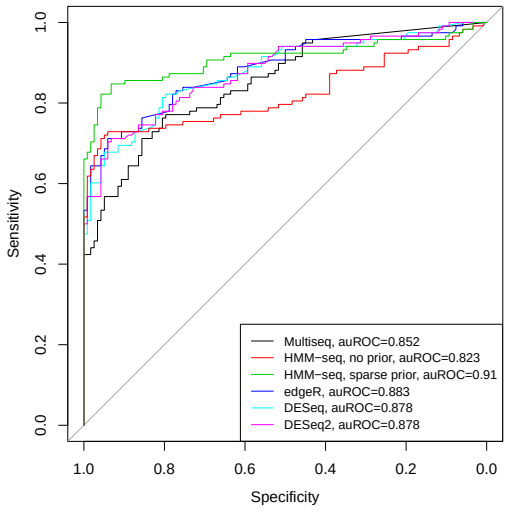


Figure 3.7: ROC curves and areas under the curves for ChIP-seq-based simulations under the Negative Binomial generation scheme. For each of the 117 regions, one null case was simulated in which an effect is absent in the 4 samples, and one non-null case was simulated in which an effect was present in the 4 samples. Methods being compared include *Multiseq*, *HMM-seq* with and without the sparse prior, *edgeR*, *DESeq* and *DESeq*2. All window-based approaches used 300bp windows, which close to the width of the effect being simulated. Panels (a), (b) and (c) show results when the region sizes are 2^{11}bp, 2^{13}bp and 2^{15}bp respectively.



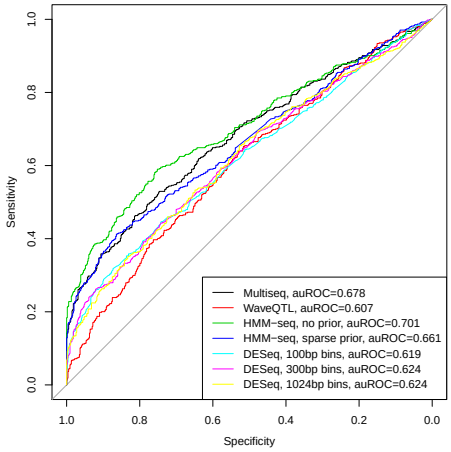
(a)

(b)

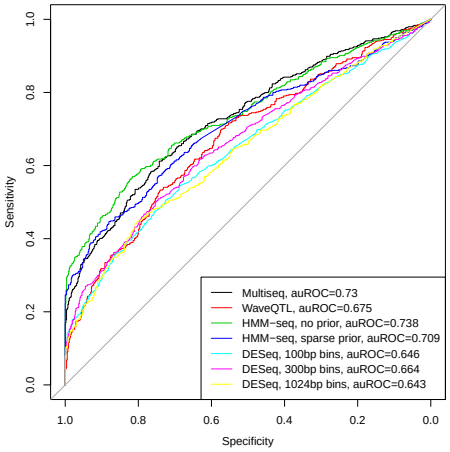


(c)

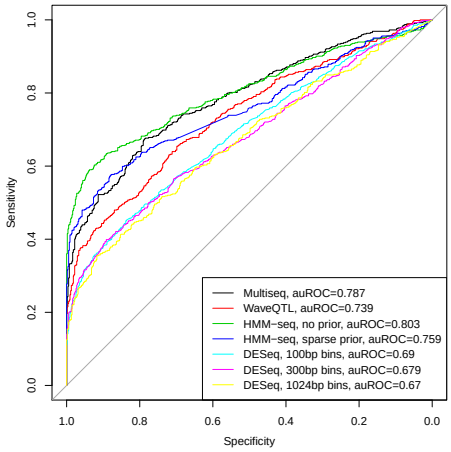
Figure 3.8: ROC curves and areas under the curves for ChIP-seq-based simulations under the Beta Binomial generation scheme. For each of the 117 regions, one null case was simulated in which an effect is absent in the 4 samples, and one non-null case was simulated in which an effect was present in the 4 samples. Methods being compared include *Multiseq*, *HMM-seq* with and without the sparse prior, *edgeR*, *DESeq* and *DESeq*2. All window-based approaches used 300bp windows, which close to the width of the effect being simulated. Panels (a), (b) and (c) show results when the region sizes are 2^{11}bp, 2^{13}bp and 2^{15}bp respectively.



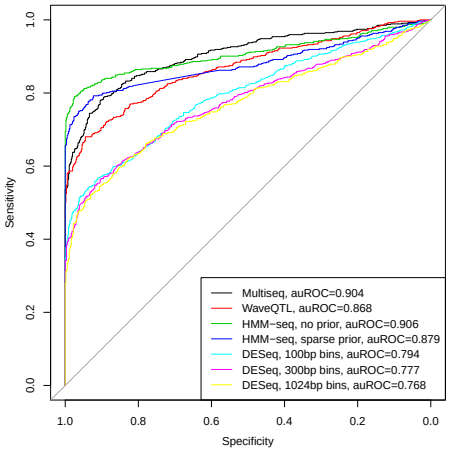
(a)



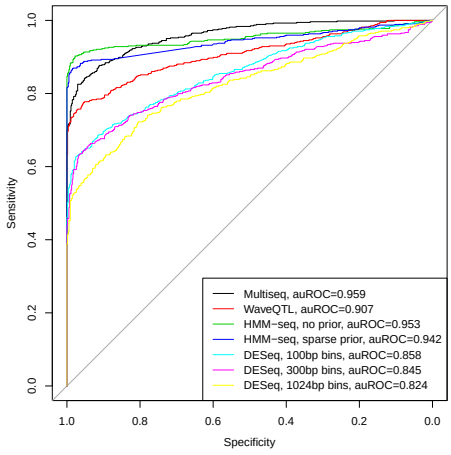
(b)



(c)

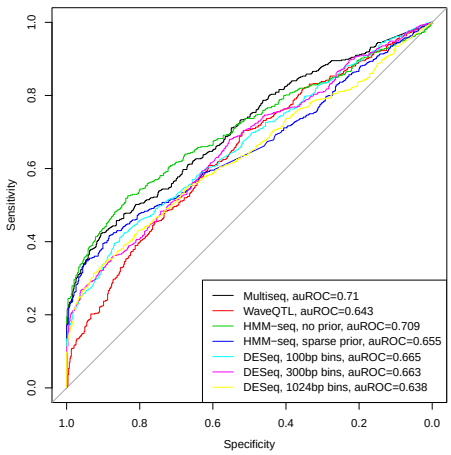


(d)

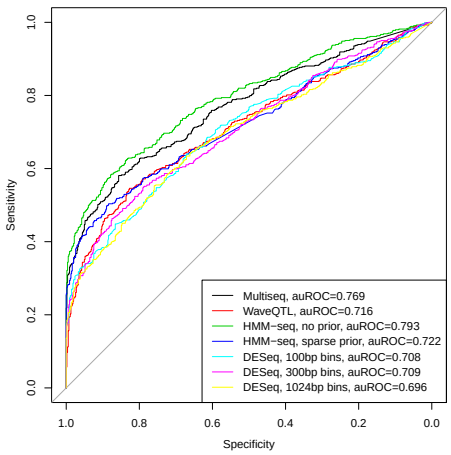


(e)

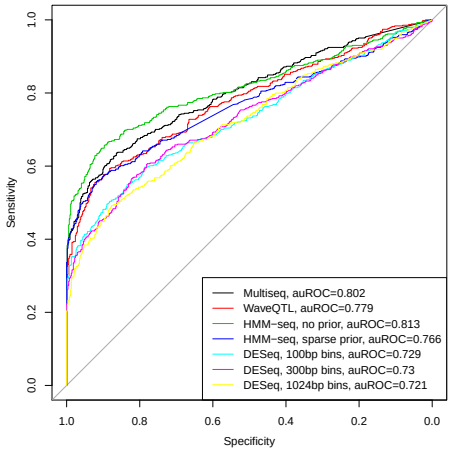
Figure 3.9: ROC curves and areas under the curves for DNase-seq-based simulations under the Negative Binomial generation scheme, where the simulated effect is estimated from raw data using *bayesthr*. For each of the 578 regions, one null case was simulated in which an effect is absent, and one non-null case was simulated in which an effect was present. Methods being compared include *Multiseq*, *WaveQTL*, *HMM-seq* with and without the sparse prior, and *DESeq* with various choices of window sizes. Panels (a), (b), (c), (d) and (e) show results when the sample sizes are 4, 6, 10, 30 and 70 respectively.



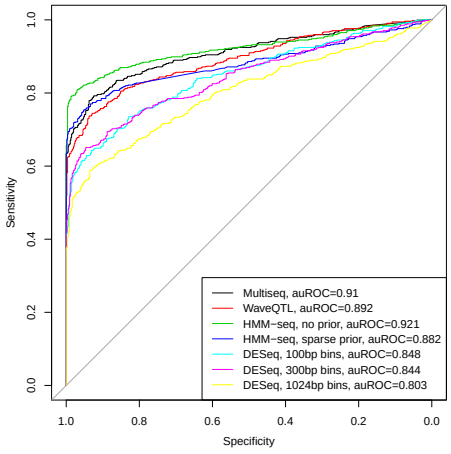
(a)



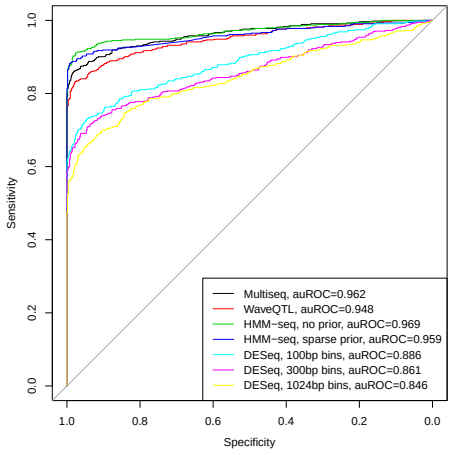
(b)



(c)



(d)



(e)

Figure 3.10: ROC curves and areas under the curves for DNase-seq-based simulations under the Beta Binomial generation scheme, where the simulated effect is estimated from raw data using *bayesthr*. For each of the 578 regions, one null case was simulated in which an effect is absent, and one non-null case was simulated in which an effect was present. Methods being compared include *Multiseq*, *WaveQTL*, *HMM-seq* with and without the sparse prior, and *DESeq* with various choices of window sizes. Panels (a), (b), (c), (d) and (e) show results when the sample sizes are 4, 6, 10, 30 and 70 respectively.

Results for ChIP-NB indicate that *HMM-seq* performs remarkably well, outperforming *Multiseq* as well as window-based methods for all region sizes, with the gain in performance greatest for 2^{15}bp. In ChIP-BB however, *HMM-seq* is on par with *Multiseq* for region sizes 2^{11}bp and 2^{13}bp, whilst underperforming substantially for 2^{15}bp when no prior is added. When a "sparse" prior is added however, *HMM-seq* outperforms all other methods by a significant margin for all three sample sizes.

For both sets of DNase-seq simulations, *HMM-seq* with no prior slightly outperforms *Multiseq* for small sample sizes, and on par with the latter for larger sample sizes. If a "sparse" prior is added however, *HMM-seq* performs slightly worse than *Multiseq* in general, but still substantially better than the window-based approaches.

From these results we conclude that *HMM-seq* does indeed achieve superior performance compared to *Multiseq* for the ChIP-seq simulations (where the effect is weak and sparse), which is what we set out to achieve. Its overall performance is also satisfactory, almost always outperforming window-based approaches and being on par with *Multiseq* on average, albeit with considerable variation in performance. Nevertheless, we note that the performance of *HMM-seq* is heavily influenced by two factors: 1) the accuracy for which the baseline μ_b^o is estimated, and 2) the shape of the effect β_b^o.

The first point is illustrated by results from ChIP-BB, where we see that *HMM-seq* does not actually perform as well as either *Multiseq* or the window-based approaches when the region size is 2^{15}bp. As explained previously, this particular scenario poses considerable challenges for *Multiseq* when estimating both the baseline and the effect, and errors in estimating μ_b^o are exacerbated in the estimation of β_b^o via *HMM-seq*.

While adding a "sparse" prior as described 3.4 alleviates this issue substantially, introducing such a prior in general can be detrimental when the effect size is not actually "sparse", as the DNase-seq simulations show. On the other hand, for smaller region sizes in ChIP-BB and all simulations in ChIP-NB, *Multiseq* is able to estimate μ_b^o pretty accurately, resulting in the excellent performance of *HMM-seq* in these scenarios.

The second factor as mentioned above is demonstrated by results from DNase-NB and DNase-BB. Because the true effects used in the simulations are actually estimated from the raw data, they can be highly irregular in nature, making a Markov Chain less of a suitable model for the effect size β_b^o. In fact, we see that *Multiseq* performs substantially better than *HMM-seq* when the true effects were actually estimated using *Multiseq* (Appendix A.5), and performs nearly as well as the latter when the true effects were estimated using *bayesthr*.

As such, *Multiseq* is overall a more robust method than *HMM-seq* in terms of estimating effect sizes, being worse than the latter on occasions where the true effect is weak and sparse. Efficiency-wise, *Multiseq* is computationally faster than *HMM-seq* (excluding the *Multiseq* step, which has to be run in the first place), especially for large regions. Hence, we will use the multiscale methods *Multiseq* and *smash* in tackling other interesting problems (3.5 and 4), keeping *HMM-seq* as a complementary tool.

3.5 Application: inferring transcription factor binding from chromatin accessibility data

One primary goal in genomics is a better understanding of the human regulatory system, in particular non-coding genomic sequences that regulates gene expression. Transcription factors (TFs) are of particular interest because they directly control transcription (as the name implies) by binding to specific locations on the genome. As mentioned in Section 1.1, ChIP-seq experiments have been widely used to locate TF binding sites, and can be regarded as the gold standard sequencing experiment for this task. However, each ChIP-seq experiment profiles only one TF, so the costs of conducting these experiments can be prohibitive when one wishes to find TF binding sites in general. On the other hand, sequencing experiments such as DNase-seq or FAIRE-seq (formaldehyde-assisted isolation of regulatory elements) measure chromatin accessibility and thus correlate with TF binding. Based on this line of reasoning, methods such as CENTIPEDE (Pique-Regi et al. (2011)) and msCentipede (Raj et al. (2015)) make use of data from assays measuring chromatin accessibility - as well as external sources of information - to infer TF binding.

At its core, this is essentially a classification problem, where we wish to classify a site as being bound or not based on a variety of features (of which chromatin accessibility data is a prominent one). Both CENTIPEDE and msCentipede are unsupervised approaches to the problem, where actual ChIP-seq measurements are not given. Here, we take a supervised approach to the problem, making use of available ChIP-seq data in addition to chromatin accessibility data to infer binding at

new locations. Using this supervised approach, our main goal is to better understand the relationship between ChIP-seq and DNase-seq data. A simple Bayesian model for computing the posterior probability odds at a new site is

$$\frac{P(B=1|D,T)}{P(B=0|D,T)} = \frac{P(D|B=1,T)}{P(D|B=0,T)} \frac{P(B=1|T)}{P(B=0|T)} \tag{3.25}$$

where T is the training data (both DNase-seq and ChIP-seq measurements), $B = 0$ or 1 indicates no binding and binding respectively at the new site, and D represents chromatin accessibility data at the new site. Here $P(B = 1)$ and its complement can be computed empirically from the available ChIP-seq data. In practice however, various peak calling algorithms produce different results, and it is difficult to decide on a single best algorithm for deciding if binding actually occurs. Instead, we use the total number of ChIP-seq reads at any particular site as a surrogate for determining TF binding. This particular measure is also much more informative than a binary "bind" or "no bind" condition.

We thus modify (3.25) to become

$$P(C=c|D,T) = \frac{P(D|C=c,T)P(C|T)}{P(D|T)} \tag{3.26}$$

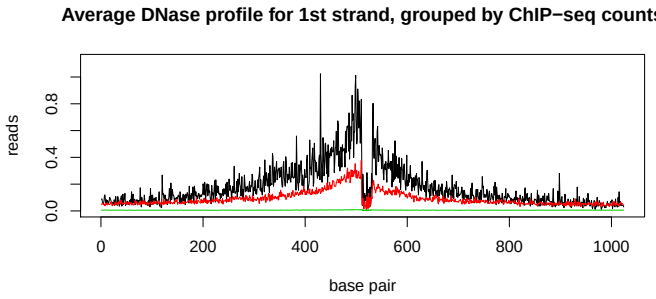
where C represents total ChIP-seq reads at the new site. Here our goal is to determine the posterior distribution for the number of total ChIP-seq reads at a new site. We estimate the prior $P(C|T)$ using its empirical distribution from the ChIP-seq data in T. For the likelihood component, we make the standard assumption of a Poisson

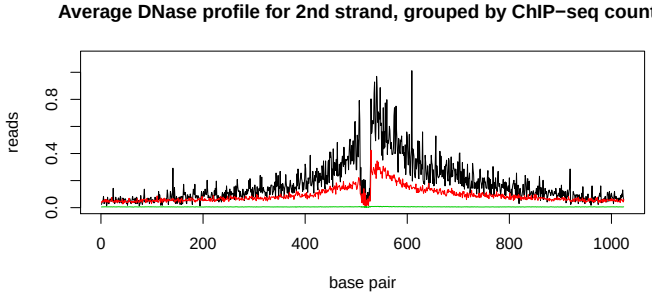
distribution on the DNase-seq reads D:

$$P(D|C=c,T)=\prod_b P(D_b|C=c,T)=\prod_b \frac{e^{\hat{\lambda}_{b,c}}\hat{\lambda}_{b,c}^{D_b}}{D_b!} \tag{3.27}$$

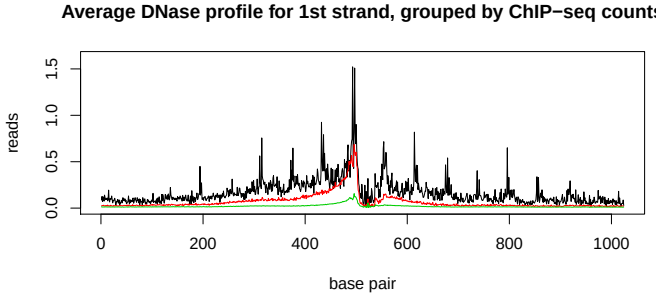
where $\hat{\lambda}_{b,c}$ is the estimated Poisson intensity for DNase-seq measurements at base b when the total ChIP-seq reads is c. To bring *Multiseq* into the framework and estimate the rate $\lambda_{b,c}$, we first visualize what $\lambda_{b,c}$ looks like for different values of c. Figure 3.11 shows the average DNase-seq read profiles for two example TFs, BATF and CTCF, when total ChIP-seq reads is low, medium or high. Note that DNase-seq data is strand-specific, so there are two plots for each TF, one for each strand. Here sites marked as "low" is defined to have total ChIP-seq reads less than the 90-th percentile; sites marked as "medium" falls in the 90-th to 99-th percentile; sites marked as "high" have total reads above the 99-th percentile.

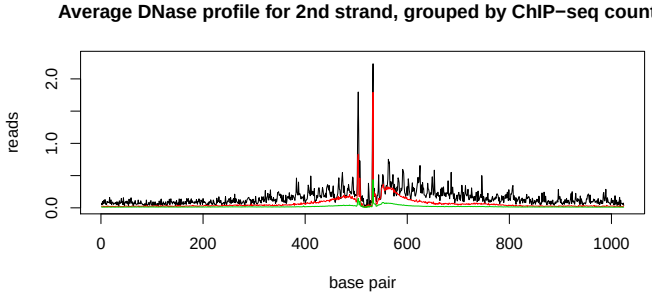
We can clearly see that there is a monotonic positive relationship between average DNase-seq reads and total ChIP-seq reads: the higher the total ChIP-seq reads, the higher the intensity for the corresponding DNase-seq profile. This relationship can be captured well by the *Multiseq* model as described in (3.2). Specifically, we run *Multiseq* on the DNase-seq data in T, with each site being a sample. The covariate in the *Multiseq* model will be the vector of total ChIP-seq reads C that are in the training set T. *Multiseq* will output the posterior means of $\hat{\mu}_b^o$ and $\hat{\beta}_b^o$. Using these quantities, we can compute the desired quantity of interest $\hat{\lambda}_{b,c} \equiv \hat{\mu}_b^o + \hat{\beta}_b^o c$, which is the posterior mean of the Poisson intensity at base b when total ChIP-seq reads is c. Putting this in (3.27) allows us to compute the posterior probabilities $P(C=c|D,T)$.





(a)





(b)

Figure 3.11: Average DNase-seq profiles for two examples TFs, grouped according to different total ChIP-seq reads. Panel (a) shows the average profiles for BATF, where the first plot corresponds to the forward strand, and the second plot corresponds to the reverse strand. Panel (b) shows the average profiles for CTCF, where the first plot corresponds to the forward strand, and the second plot corresponds to the reverse strand. In all four plots, the black lines correspond to the average DNase-seq profile for samples with total ChIP-seq reads above the 99-th percentile; the red lines correspond to the average DNase-seq profile for samples with total ChIP-seq reads between the 90-th and the 99-th percentile; the green lines correspond to the average DNase-seq profile for samples with total ChIP-seq reads below the 90-th percentile.

(3.26) gives us the posterior distribution for the total ChIP-seq reads C at any new site, which is very informative of binding, possibly more so than a binary outcome as determined by a peak calling algorithm (since none is guaranteed to be perfect). If, however, a binary outcome of binding vs no binding is indeed desired, an appropriate threshold H on C can be computed using any or several of the available peak calling algorithms. Finding the posterior probability of binding $P(B = 1|D, T)$ given H as well as the posterior distribution for C is then a trivial task.

3.5.1 Case study: inferring binding for the GM12878 cell line

To assess the performance of our method, we compare it against CENTIPDE and msCentipede on DNase-seq and ChIP-seq data obtained from the GM12878 lymphoblastoid cell line. We follow closely the analysis routine described in Raj et al. (2015), using data kindly provided by A. Raj. Total ChIP-seq reads in 400bp windows centered at the 42 motif instances were derived from experiments conducted as part of the ENCODE project, while DNase-seq reads in a 2^{14}bp window centered at the same motif instances were obtained from a study conducted at the University of Washington (Hesselberth et al. (2009)). As an initial filtering step, we removed sites which had mappability issues, as well as sites for which the position-weight matrix (PWM) scores were less than 10, following the procedure outlined in Raj et al. (2015). For our method, we split the data into a 80% training set and and a 20% test set, where the training set (including both DNase-seq and ChIP-seq measurements) is used to infer the parameters $\hat{\lambda}_b^c$, and the test set is used to assess accuracy of our method, as described below. For CENTIPEDE and msCentipede,

we run them on the DNase-seq data from all sites with PWM scores greater than 13 to learn model parameters, instead of from just the training set, as they are unsupervised approaches. For the results to be consistent, we evaluate the accuracy of these two methods on just the 20% test set. We also provide PWM scores to both CENTIPEDE and msCentipede as additional input features.

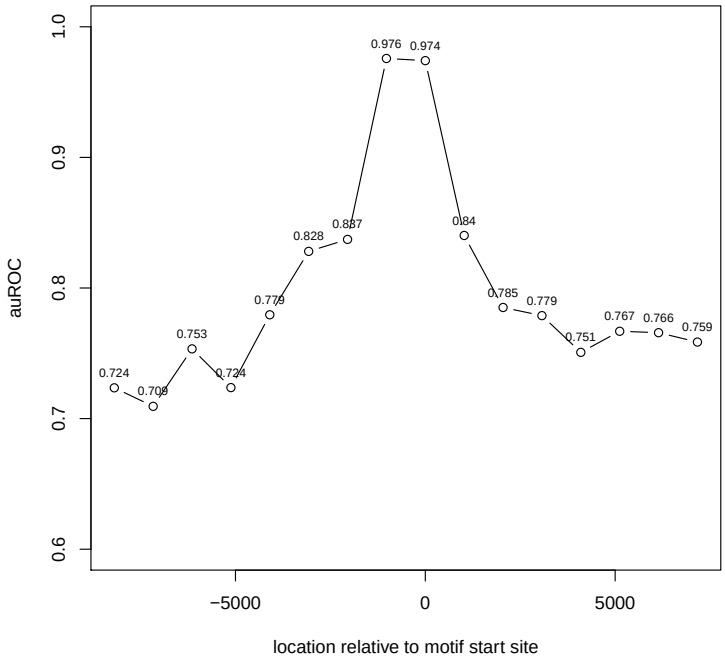
To evaluate the accuracy of all three methods, we use the auROC as the performance metric. To compute the auROC, a gold standard set of bound and unbound sites for a given TF were generated using the peak calling algorithm MACS at a 1% FDR cutoff. Specifically, bound sites were ones that lay within a ChIP-seq peak as defined by MACS, while unbound sites were ones that did not lie within any ChIP-seq peaks, and had fewer total reads than the median amount over all the sites for that TF. Given the "true" binding labels on each of the sites in the 20% test set, an auROC can be computed based on the rankings induced by the posterior probability of binding estimated by all three methods.

Since there is a huge number of putative binding sites for any given TF-PWM pair, we first determine the ideal region for DNase-seq data on which to train *Multiseq*. Because different PWM models can be associated with the same TF, we will use the naming convention (name of TF)-(ID of PWM model) for each TF-PWM pair. Figure 3.12a plots the auROCs for BATF-S356 for sliding windows of 1024bp across a region of length 16384bp, centered on the start of the motif instance. Figure 3.12b plots the auROCs for BATF-S356 with varying region sizes all centered on the start of the motif instance. From these two figures we can conclude that 1) the region that provides the strongest signal for TF binding comprises of bases immediately

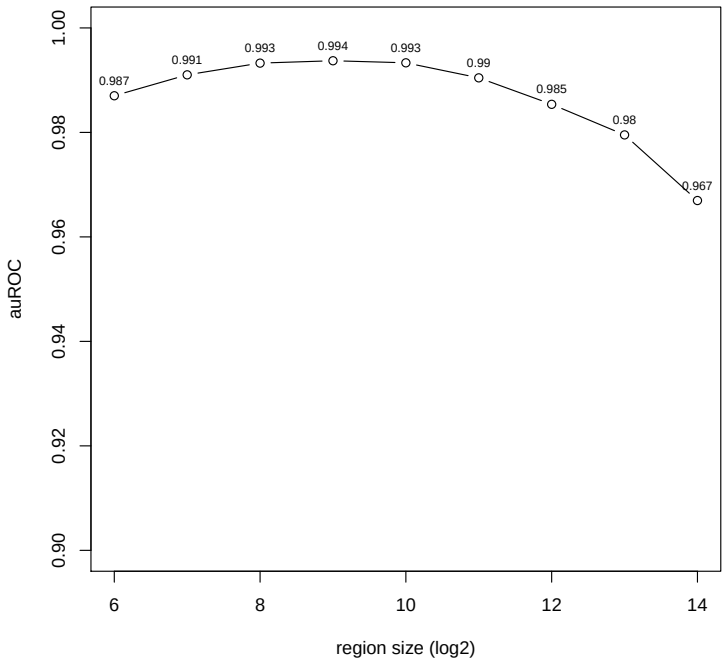
surrounding the first base of the motif, with decaying strength further out, and 2) the region size that correlates best with TF binding is less than 1024bp. Other TF-PWM pairs have the same trends, so these observations hold in general. As such, we run *Multiseq* on DNase-seq data consisting of x base pairs surrounding the start of the motif for each TF-PWM pair, where $x \in \{64, 128, 256, 512, 1024\}$.

While there are many TF-PWM pairs, many of them result in the same relative performance of *Multiseq*, CENTIPEDE and msCentipede. Hence, we show figures that can succinctly summarize the main patterns we found, and record the auROCs for each TF-PWM pair as a table in Appendix A.6. Figures 3.13-3.17 outline the main patterns in decreasing frequency of occurrence.

Overall, we conclude that using *Multiseq* as a supervised approach to inferring binding is superior to both of the unsupervised approaches CENTIPEDE and msCentipede for this dataset. It consistently outperforms CENTIPEDE, and underperforms in relation to msCentipede only for a couple of TF-PWM pairs. However, we recognize the shortcoming of our approach in that it still requires some ChIP-seq data to train on, which limits its use in practice. On the other hand, through this data analysis we have discovered interesting relationships between total ChIP-seq counts and DNase-seq data. In particular, the optimal window size for DNase-seq data to learn from is typically between 512-1024bp, but both CENTIPEDE and msCentipede often has lower auROCs when trained on DNase-seq data with these window sizes, compared to when they are trained on smaller windows. This suggests that both CENTIPEDE and msCentipede are not making optimal use of all available DNase-seq data, and that there is still much room for improvement.



(a)



(b)

Figure 3.12: AuROCs from the *Multiseq*-based inference procedure for BATF-S356. Panel (a) shows the auROCs when *Multiseq* is trained on DNase-seq data from contiguous 1024bp windows within a region of length 16384bp around the start of the motif instance. Panel (b) shows the auROCs when *Multiseq* is trained on DNase-seq data from different regions sizes (2^6bp-2^{14}bp).

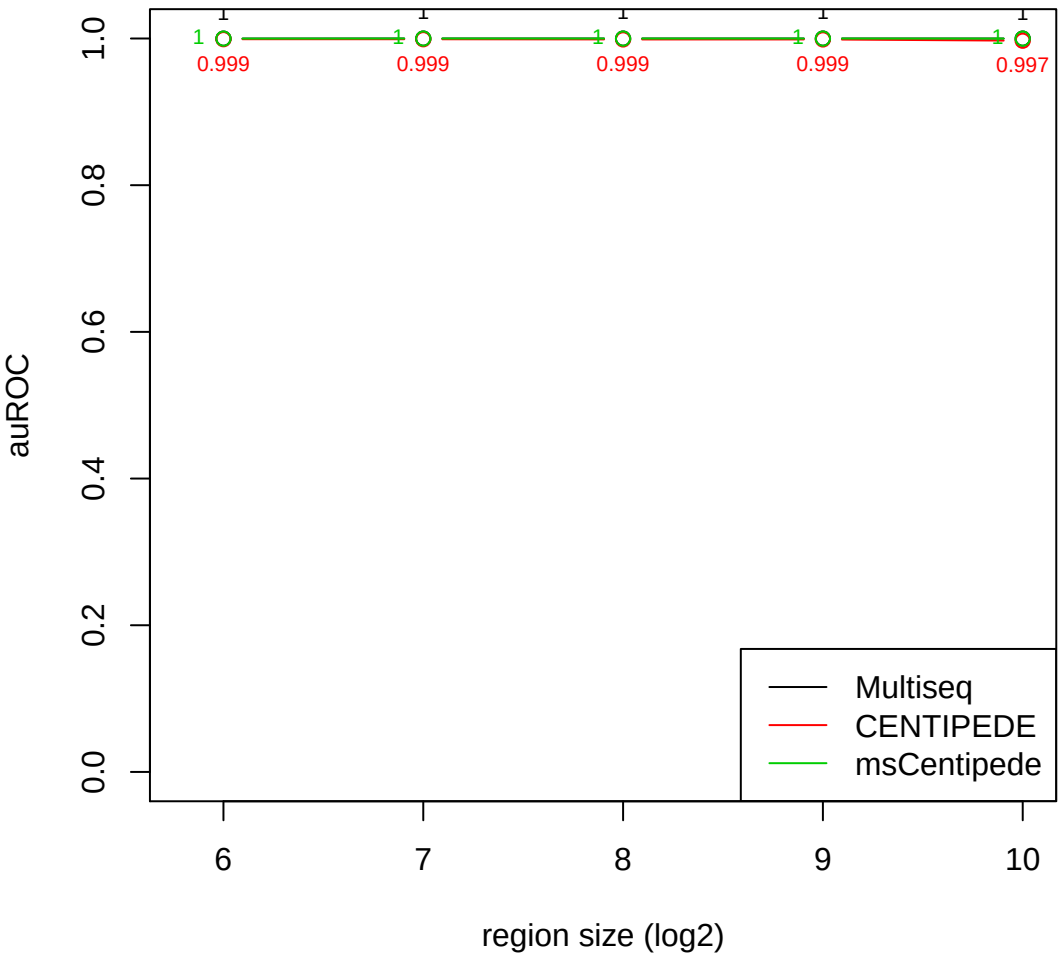


Figure 3.13: AuROCs corresponding to the *Multiseq*-based inference procedure (black), CENTIPEDE (red) and msCentipede (green) for NFE2-S92. Here all three methods achieve nearly "perfect" auROCs, where the largest auROC across all region sizes is greater than 0.995. Other TF-PWM pairs showing this kind of pattern (possibly excluding CENTIPEDE) include E2F4-S753, E2F4-S754, ELK1-S88, ELK1-S89, ELK1-S90, ELK1-S841, ERRA-S661, ERRA-S662, ETS1-S99, ETS1-S100, ETS1-S101, IRF3-S147, IRF4-S148, MAX-S325, NRF1-S194 and SREBP2-S342.

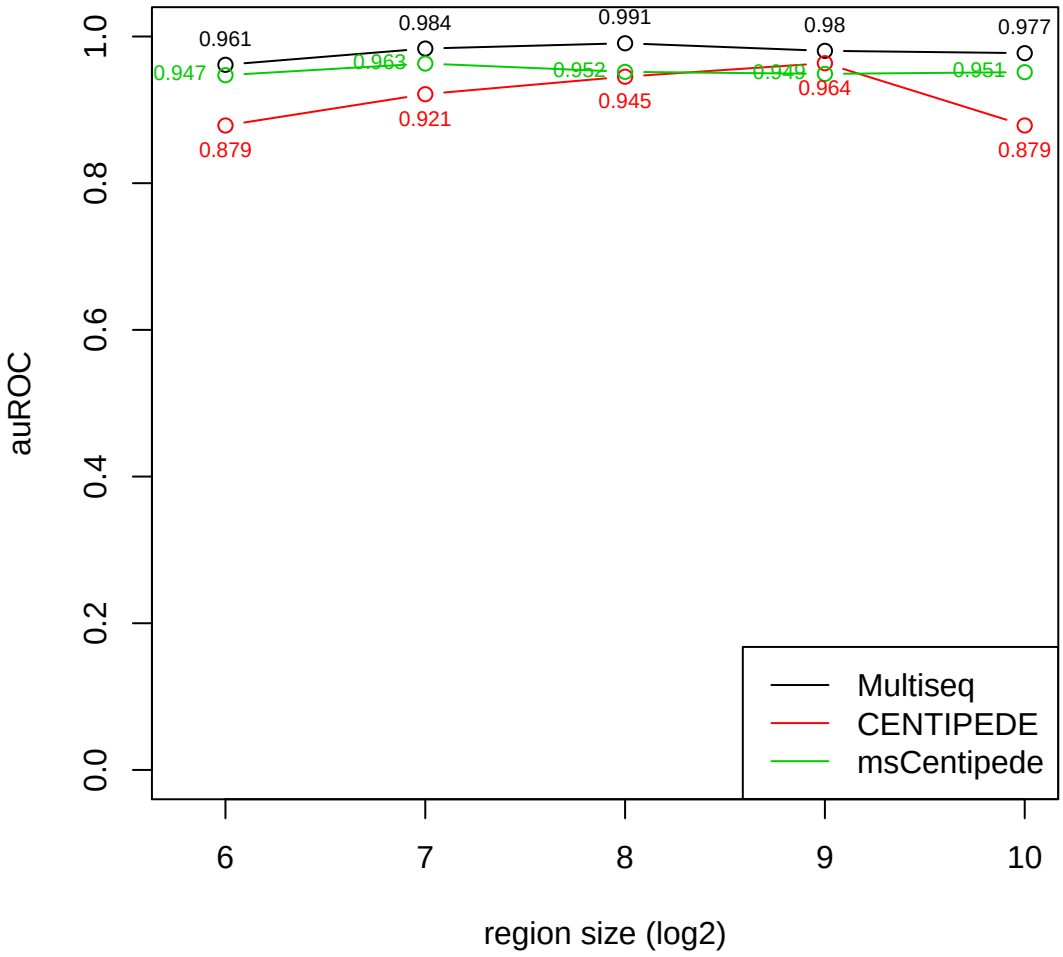


Figure 3.14: AuROCs corresponding to the *Multiseq*-based inference procedure (black), CENTIPEDE (red) and msCentipede (green) for BATF-S356. Here the *Multiseq*-based inference procedure outperforms both CENTIPEDE and msCentipede. Other TF-PWM pairs showing this kind of pattern include BHLHE40-S311, CEBPB-S357, EBF-S79, EGR1-S4, ERRA-S838, NFATC1-S186, NFATC1-S187, NFATC1-S188, NFKB-S189, NFKB-S190, RFX5-S240, RFX5-S241 and RFX5-S825 and ZNF143-S43.

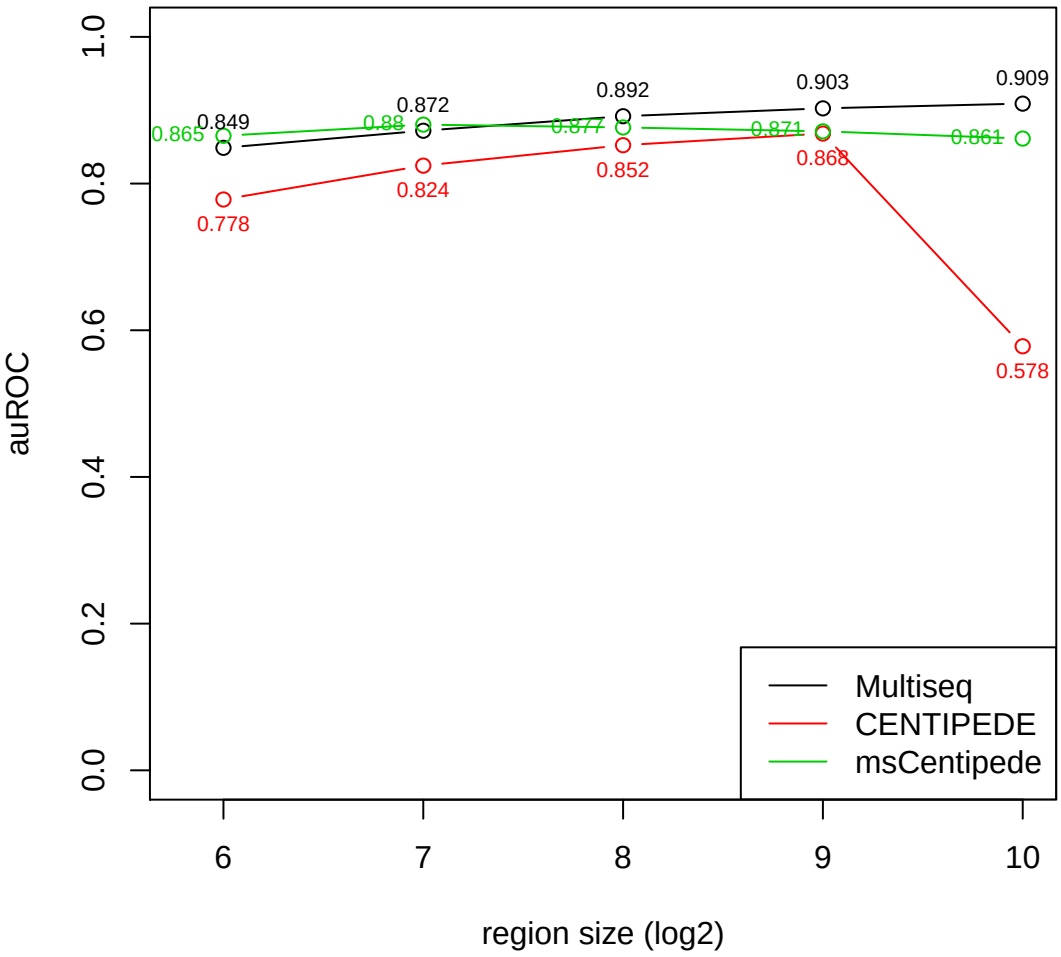


Figure 3.15: AuROCs corresponding to the *Multiseq*-based inference procedure (black), CENTIPEDE (red) and msCentipede (green) for RUNX3-S250. Here msCentipede outperforms the *Multiseq*-based procedure for smaller region sizes, but the latter outperforms the former overall (using maximum auROC across all region sizes) when larger region sizes are taken into account. The *Multiseq*-based procedure outperforms CENTIPEDE for all window sizes. Other TF-PWM pairs showing this kind of pattern include ELF1-S81, ELF1-S82, PU1-S123, SRFV0416101-S159 and SRFV0416101-S160.

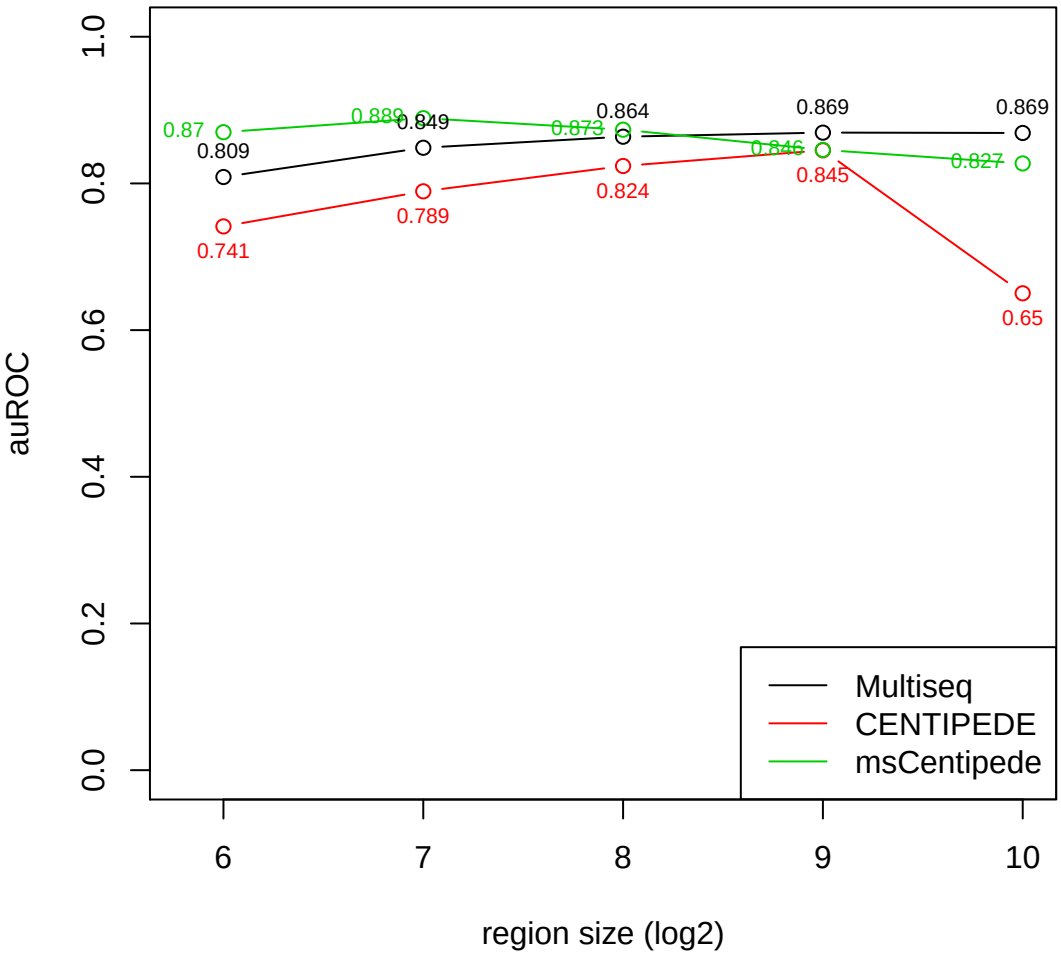


Figure 3.16: AuROCs corresponding to the *Multiseq*-based inference procedure (black), CENTIPEDE (red) and msCentipede (green) for PAX5C20-S200. Here msCentipede performs the *Multiseq*-based inference procedure. Other TF-PWM pairs showing this kind of pattern include MAFK-S383 and MAFK-S384.

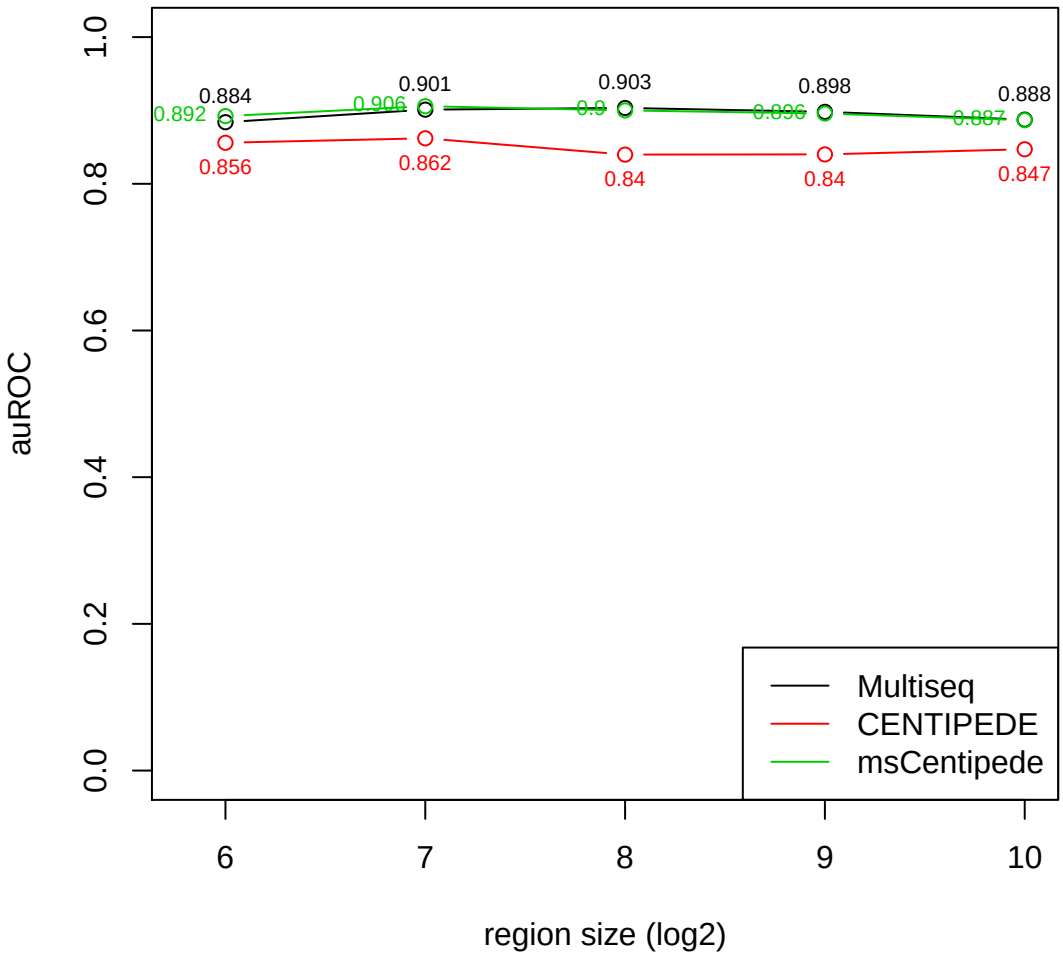


Figure 3.17: AuROCs corresponding to the *Multiseq*-based inference procedure (black), CENTIPEDE (red) and msCentipede (green) for CTCF-S2. Here the maximum auROC across all regions for both the *Multiseq*-based inference procedure and msCentipede are nearly identical (< 0.003 difference). Both methods outperform CENTIPEDE. Other TF-PWM pairs showing this kind of pattern include GABP-S116.

CHAPTER 4

CLUSTERING SEQUENCING DATA USING A SMOOTHED GRADE OF MEMBERSHIP MODEL

4.1 Overview

Clustering techniques have often been used in the analysis of genetic data as a powerful exploratory tool. In particular, the STRUCTURE model in Pritchard et al. (2000) (also known as the grade-of-membership/admixture/latent Dirichlet allocation/topic model) was proposed as a "soft" clustering method for genotype data to infer population structure. In this type of clustering, each sample can have partial membership in each cluster, potentially revealing complex relationships between the samples of interest. Other clustering tools such as k-means and hierarchical clustering have also been used in the analysis of micro-array data, to cluster both genes and samples, as well as sequencing data. A review of these methods can be found in Jiang et al. (2004). More recently a similar grade-of-membership (GoM) model was proposed by Dey et al. (2016) to cluster samples from RNA-seq data, motivated by the argument that each sample could exhibit behavior from a variety of different cell types, potentially revealing similarities and dissimilarities between different samples.

Motivated by the results from the GoM models in both Pritchard et al. (2000) and Dey et al. (2016), we propose an extension of the latter model to analyze sequencing data at the base pair resolution (RNA-seq reads were summed up within genes, or other "units" e.g. transcripts, exons, in Dey et al. (2016)). The goal of such a model is to capture clusters at a finer scale compared to gene-level analysis. For

example, certain samples might share similar expression patterns for a portion of a gene, but reveal different patterns in another portion of the same gene. One possible explanation for this might be variations in splicing patterns, and new clusters could potentially reveal transcripts not previously found.

One main issue with applying the GoM model directly is the low number of reads at each base in typical sequencing data. For a substantial fraction of the samples and genes, there could be as few as zero counts in more than 70% of the bases. The low signal-to-noise ratio means that the standard GoM model might return estimates of distinct cluster centers that are, in fact, very similar to each other. Indeed, results in Sections 4.2 and 4.3 will illustrate this point, as we compare the standard GoM model to ours.

To model the low read count when analyzing sequencing data at the base pair resolution, we make the assumption that the cluster centers are "spatially-structured", and hence induce some form of "smoothing" when estimating the cluster centers by making use of the multiscale framework described in Chapter 2 (also see Shim and Stephens (2015), Xing and Stephens (2016)). To be precise, we add an additional smoothing step during each update of the EM algorithm for the GoM model by running *smash* on the estimated cluster centers. As we will see in Sections 4.2 and 4.3, this additional step greatly improves the accuracy of the model when compared to the standard GoM model.

For easier conceptual understanding, we first describe the basic GoM model (without smoothing) in terms of a generative model, which (given the cluster profiles) can be specified as

1. For the j-th DNA fragment (for which we only consider the first nucleotide) in sample i, first choose cluster Z_{ij} according to $P(Z_{ij} = k|\boldsymbol{\pi}) = \pi_{ik}$, where $\sum_k \pi_{ik} = 1$, for $k = 1, ..., K$.

2. Given Z_{ij}, choose the base R_{ij} it maps to according to $P(R_{ij} = b|Z_{ij}, \boldsymbol{\phi}) = \phi_{Z_{ij},b}$, where $\sum_b \phi_{kb} = 1$.

Given this generative model, the likelihood $P(D|\boldsymbol{\phi}, \boldsymbol{\pi})$ is given by

$$\begin{aligned} P(R|\boldsymbol{\phi}, \boldsymbol{\pi}) &= \prod_i \prod_j \sum_k P(R_{ij}|Z_{ij} = k, \boldsymbol{\phi}) P(Z_{ij} = k|\boldsymbol{\pi}) \\ &\propto \prod_i \prod_b \left(\sum_k P(R_{ij} = b|Z_{ij} = k, \boldsymbol{\phi}) P(Z_{ij} = k|\boldsymbol{\pi}) \right)^{Y_{ib}} \\ &\qquad \text{where } Y_{ib} = \textstyle\sum_j I_{\{R_{ij}=b\}} \\ &= \prod_i \prod_b \left(\sum_k \pi_{ik} \phi_{kb} \right)^{Y_{ib}} \end{aligned} \tag{4.1}$$

For notational convenience, we have moved the sample indicator i to the subscript here instead of the superscript as in the multiscale models. Note that (4.1) re-expresses the likelihood in terms of the sum of the reads at a given base pair, Y_{ib}, which is sufficient for the information contained in the actual data R_{ij}. Alternatively, this model can be expressed as

$$\begin{aligned} &(Y_{i1}, ..., Y_{iB}) \sim \text{Multinomial}(Y_{i\cdot}, p_{i1}, ..., p_{iB}) \\ &\text{where } \; Y_{i\cdot} := \sum_b Y_{ib}, \;\; p_{ib} := \sum_k \pi_{ik} \phi_{kb} \end{aligned} \tag{4.2}$$

which is exactly the model (2) presented in Dey et al. (2016). Fitting the model (4.2) requires us to estimate the quantities $\boldsymbol{\pi} = \{\pi_{ik}\}_{i\in\{1,\ldots,m\},k\in\{1,\ldots,K\}}$ and $\boldsymbol{\phi} = \{\phi_{kb}\}_{k\in\{1,\ldots,K\},b\in\{1,\ldots,B\}}$. We choose estimates for $\boldsymbol{\pi}$ and $\boldsymbol{\phi}$ that maximize the likelihood (4.1), using an EM algorithm described in Appendix A.7. Following the derivations in Appendix A.7, the parameter updates in the M step for a standard GoM model are given by

$$\pi_{ik}^{(t+1)} = \frac{\sum_b Y_{ib}\left(\frac{\pi_{ik}^{(t)}\phi_{kb}^{(t)}}{\sum_{k'}\pi_{ik'}^{(t)}\phi_{k'b}^{(t)}}\right)}{\sum_{b'} Y_{ib'}} \tag{4.3}$$

$$\phi_{kb}^{(t+1)} = \frac{\sum_i Y_{ib}\left(\frac{\pi_{ik}^{(t)}\phi_{kb}^{(t)}}{\sum_{k'}\pi_{ik'}^{(t)}\phi_{k'b}^{(t)}}\right)}{\sum_i \sum_{b'} Y_{ib'}\left(\frac{\pi_{ik}^{(t)}\phi_{kb'}^{(t)}}{\sum_{k'}\pi_{ik'}^{(t)}\phi_{k'b'}^{(t)}}\right)} \tag{4.4}$$

As noted above, the standard GoM model does not perform well when applied directly to sequencing data, so we add a "smoothing" step at each iteration of the EM algorithm. Specifically, we modify the M step as follows:

$$\pi_{ik}^{(t+1)} = \frac{\sum_b Y_{ib}\left(\frac{\pi_{ik}^{(t)}\phi_{kb}^{(t)}}{\sum_{k'}\pi_{ik'}^{(t)}\phi_{k'b}^{(t)}}\right)}{\sum_{b'} Y_{ib'}} \tag{4.5}$$

$$\phi_{kb}^{(t+1)} = f\left(\frac{\sum_i Y_{ib}\left(\frac{\pi_{ik}^{(t)}\phi_{kb}^{(t)}}{\sum_{k'}\pi_{ik'}^{(t)}\phi_{k'b}^{(t)}}\right)}{\sum_i \sum_{b'} Y_{ib'}\left(\frac{\pi_{ik}^{(t)}\phi_{kb'}^{(t)}}{\sum_{k'}\pi_{ik'}^{(t)}\phi_{k'b'}^{(t)}}\right)}\right) \tag{4.6}$$

Note that the $\boldsymbol{\pi}$ updates remain unchanged. For the $\boldsymbol{\phi}$ updates, the function f

consists of the following steps:

1. Rescale the term inside f, which we call ϕ^o_{kb}. Note that the ϕ^o_{kb}'s were defined originally to be multinomial probabilities, and hence lie between 0 and 1. Since the smoothing procedure *smash* was originally designed to smooth Poisson counts, the magnitude of ϕ^o_{kb} should be close to those of actual sequencing read counts. To that end, we multiply $\{\phi^o_{kb}\}_{b=1,\ldots,B}$ by $c_k := \sum_i \sum_b Y_{ib} \left(\frac{\pi^{(t)}_{ik} \phi^{(t)}_{kb}}{\sum_{k'} \pi^{(t)}_{ik'} \phi^{(t)}_{k'b}} \right) / \sum_i \pi_{ik}$ for each k, giving us ϕ^m_{kb}.

2. Apply *smash* to ϕ^m_{kb} for each k, resulting in the smoothed cluster means ϕ^s_{kb}.

3. Normalize ϕ^s_{kb} so that $\sum_b \phi^s_{kb} = 1$. This gives us the actual updated parameters $\phi^{(t+1)}_{kb}$.

We iterate this procedure till "convergence" to obtain estimates $\hat{\boldsymbol{\pi}}$ and $\hat{\boldsymbol{\phi}}$. Note that these are not exact maximum likeliood estimates, as the smoothing step changes the likelihood function we are trying to maximize at each iteration. As such, there is no guarantee that the algorithm actually converges, especially not in terms of the likelihood function. Nevertheless, we define approximate numerical convergence in terms of the parameters $\boldsymbol{\pi}$ and $\boldsymbol{\phi}$, so that the algorithm terminates when the difference between successive parameter updates is smaller than some pre-specified tolerance. Through simulations and data analyses in the following sections, we will show that despite the heuristic nature of our approach, which we call *Cluster-seq*, we substantially improve upon the estimates from the basic GoM model, concluding that *Cluster-seq* has considerable potential when used to perform exploratory clustering for sequencing data at the base pair resolution.

4.2 Simulation results

We compare the basic GoM model with *Cluster-seq* via a very simple simulation study, which is intended to be a proof-of-concept. More interesting results can be found in Sections 4.3 and 4.4.

For this simulation study, we simulate data according to the generative model corresponding to the likelihood (4.1). Specifically, we set $K = 4$, $n = 20$ and $B = 1024$. For each sample i, the membership proportions are given by $\pi_{ik^*} = 0.7$ where $P(k^* = k) = 1/4, k = 1, 2, 3, 4$ and $\pi_{ik} = 0.1, k \neq k^*$. For each cluster k, the (unnormalized) cluster means are given by $\lambda_{1b} = 1, b \in S_1$, $\lambda_{2b} = 10, b \in S_2$, $\lambda_{3b} = 3, b \in S_3$, $\lambda_{4b} = 5, b \in S_4$, and $\lambda_{kb} = 0.1, b \notin S_k \forall k = 1, ..., 4$, where $S_1 = \{100, 101, ..., 200\}$, $S_1 = \{300, 301, ..., 400\}$, $S_1 = \{500, 501, ..., 600\}$, and $S_1 = \{700, 701, ..., 800\}$. The λ_{kb}'s as defined thus are indeed "spatially-structured". The ϕ_{kb}'s can then be obtained by normalizing the λ_{kb}'s, and are plotted in Figure 4.1.

To compare the performance of the two methods, we compute MSE($\boldsymbol{\pi}, \hat{\boldsymbol{\pi}}$) and MSE($\boldsymbol{\phi}, \hat{\boldsymbol{\phi}}$) averaged over 100 independently simulated datasets as the two assessment metrics. The MSE values, together with their associated standard deviations in parenthesis, are shown in Table 4.1 below. Here we see that the MSEs for the cluster means $\boldsymbol{\phi}$ are much smaller for *Cluster-seq* than their GoM counterparts without smoothing. The improved accuracy for *Cluster-seq* is illustrated in Figure 4.1, where we also plot the estimated ϕ_{kb} values on top of the true profiles for the two methods for one simulated run. In turn, this improves the accuracy with which the membership proportions $\boldsymbol{\pi}$ are estimated, as reflected by the MSEs for $\boldsymbol{\pi}$ in Table 4.1.

	$\boldsymbol{\phi}$	$\boldsymbol{\pi}$
Cluster-seq	1.652×10^{-7} (1.898×10^{-7})	0.0138 (0.0054)
Basic GoM model	1.610×10^{-6} (2.418×10^{-7})	0.0462 (0.0313)

Table 4.1: Comparison of accuracy (MSE) of basic GoM model and *Cluster-seq* on a simple simulation scheme over 100 independent runs. Numbers in parantheses indicate the standard deviations for the MSEs in each case. *Cluster-seq* outperforms the basic GoM model in estimating both $\boldsymbol{\pi}$ and $\boldsymbol{\phi}$.

While this is only a simple simulation study, it nevertheless demonstrates the immediate advantages of *Cluster-seq* over a standard GoM model when used to cluster sequencing data at the base pair resolution. Next, we will run *Cluster-seq* on sequencing data from two well known studies to highlight its potential for revealing complex structure amongst different biological samples.

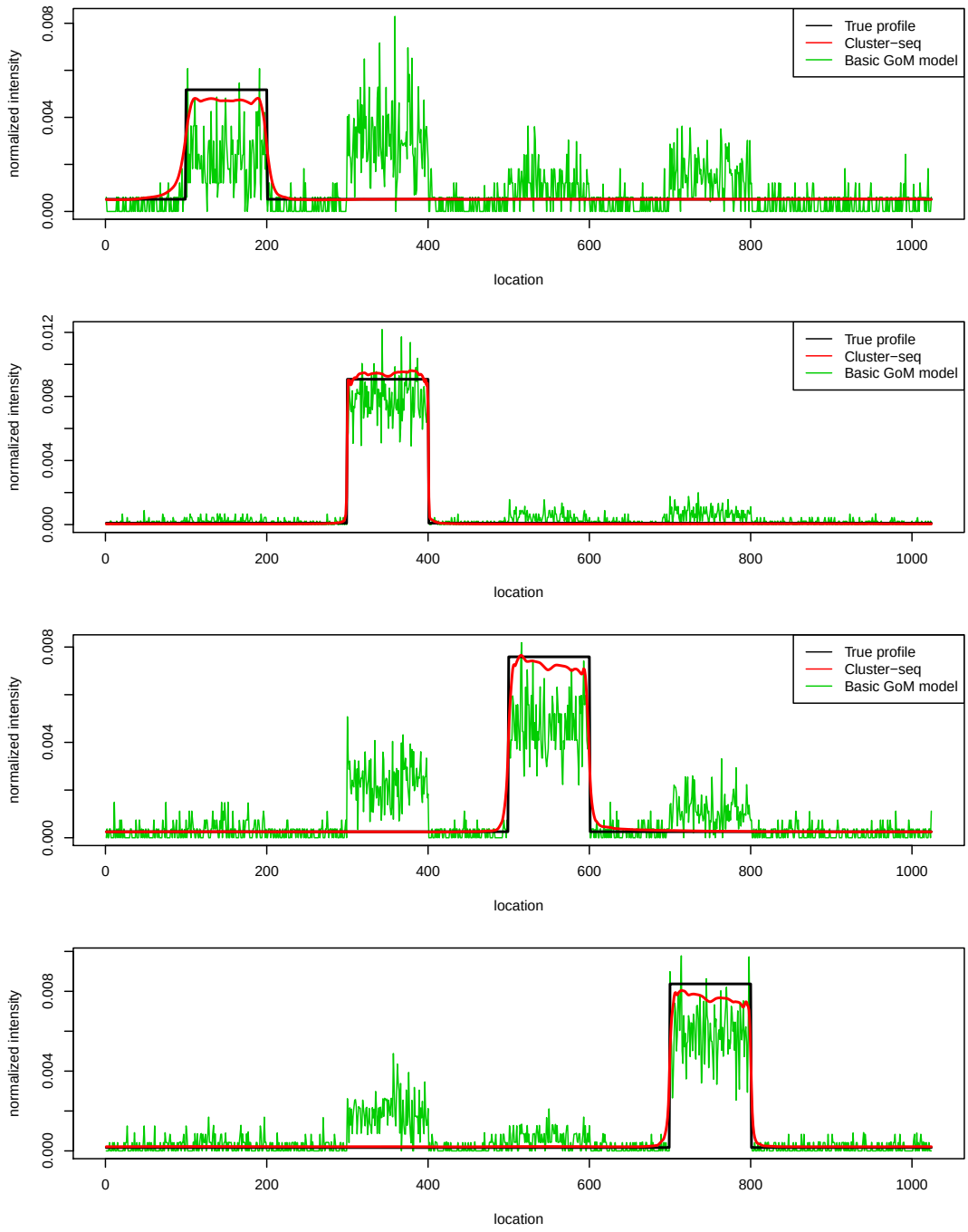
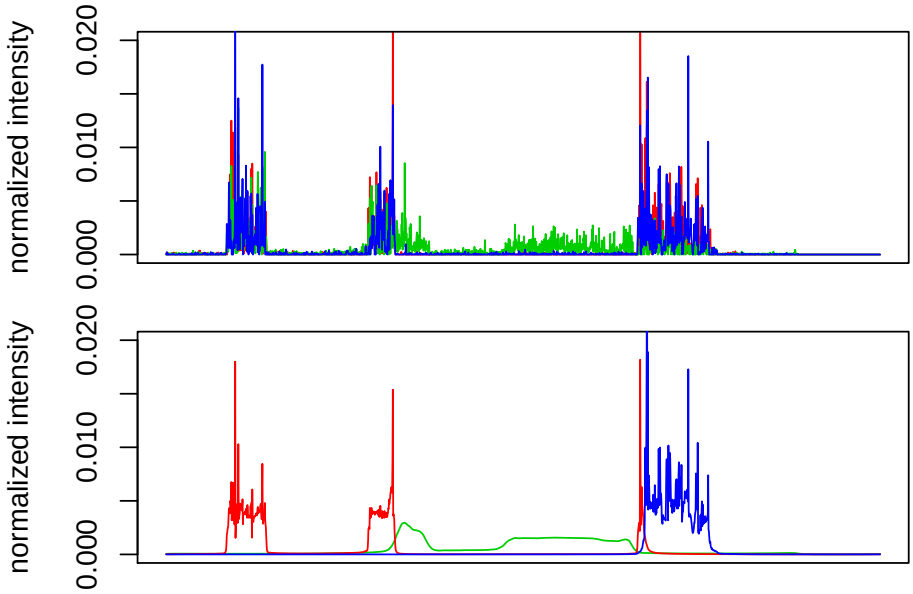


Figure 4.1: Estimates of $\boldsymbol{\phi}$ for one particular run of the simulation. The four plots correspond to the four distinct cluster means being simulated from. Black lines indicate the true cluster means; red lines indicate the estimates from *Cluster-seq*; green lines indicate the estimates from the basic GoM model. *Cluster-seq* is able to estimate the true $\boldsymbol{\phi}$ much more accurately than the basic GoM model.

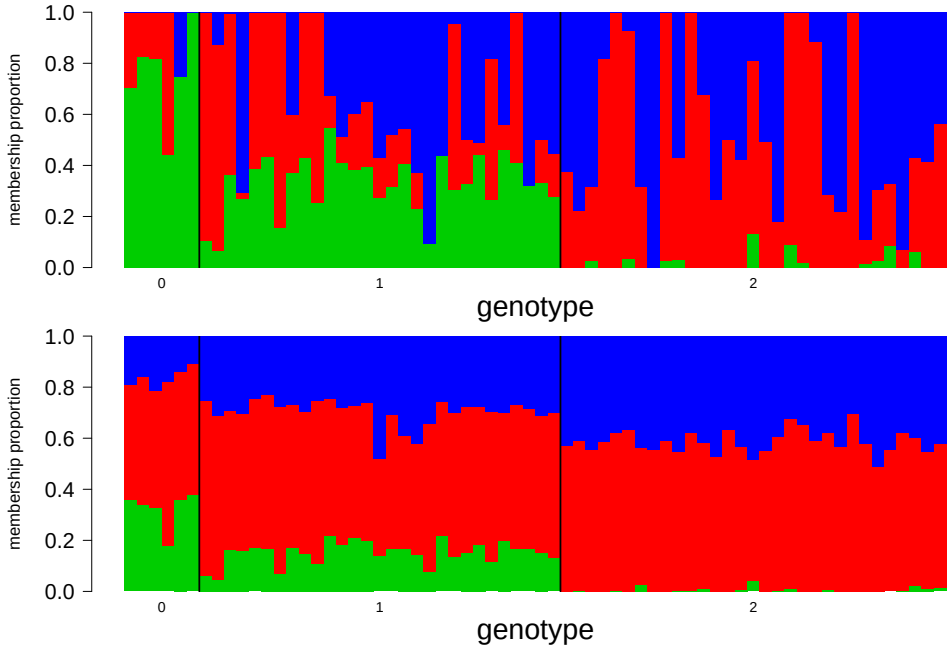
4.3 Case study: analysis of RNA-seq data for gene *OAS1*

In Section 3.3 we applied *Multiseq* to RNA-seq counts for gene *OAS1* and located differences between genotype classes for rs10774671, matching the findings from Pickrell et al. (2010). Here, we remove the genotype labels on the samples and attempt to discover the same structural differences by running *Cluster-seq* on the raw RNA-seq data.

Depending on the the type of generic variant in question, choosing the number of clusters K requires some contextual knowledge. Since many genetic variants are SNPs however, $K = 3$ is a reasonable first choice if we are indeed looking for differences induced by a genetic variant. Figure 4.2a shows the cluster means estimated by both *Cluster-seq* and the basic GoM model. Note that we estimated the cluster means using reads spanning the last three exons to better highlight the structures revealed by our method, although we expect similar results had we used data from the entire gene. Figure 4.2b shows the corresponding admixture plots for the two methods, where the genotype labels are added to better visualize the cluster structures. Admixture plots are commonly used in the field of population genetics to reveal population structure (e.g. Pritchard et al. (2000)) by using barplot representations of the membership proportions $\boldsymbol{\pi}$ across samples i, and are an extremely useful tool for summarizing the results from GoM models.



(a)



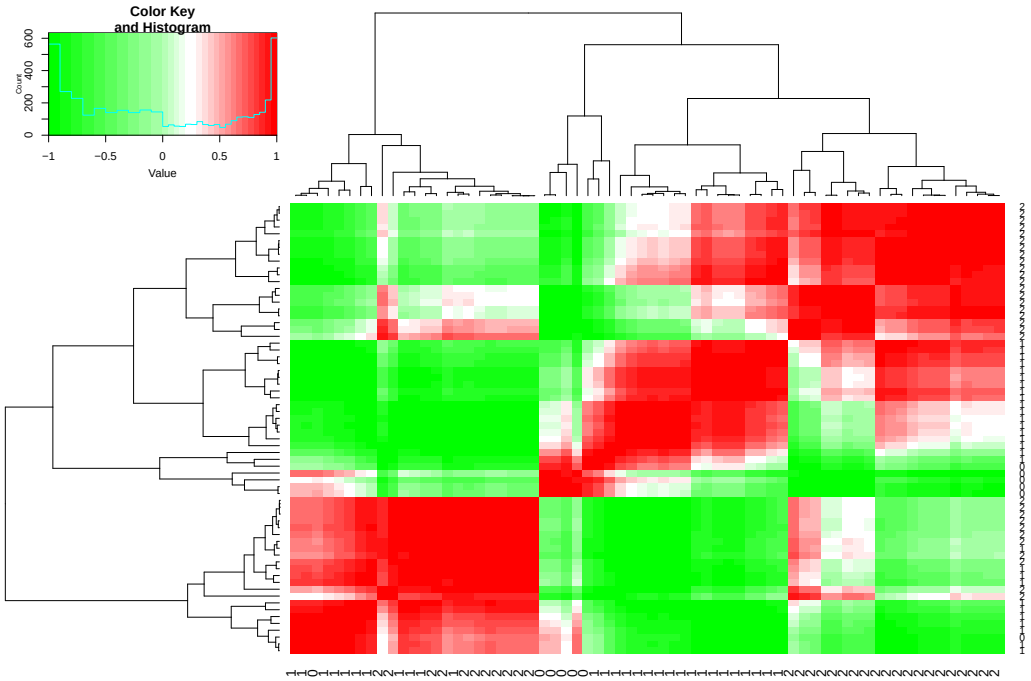
(b)

Figure 4.2: Results from running the basic GoM model and *Cluster-seq* on RNA-seq data from gene *OAS1* with $K = 3$. Panel (a) shows the estimated cluster means; top plot in panel (a) corresponds to estimates from the basic GoM model, and bottom plot in panel (a) corresponds to estimates from *Cluster-seq*. Panel (b) shows the admixture plots; top plot in panel (b) corresponds to cluster memberships for the basic GoM model, and bottom plot in panel (b) corresponds to cluster memberships for *Cluster-seq*. We see that *Cluster-seq* not only gives a better visualization of the cluster means, but is also able to separate the samples into the three genotype classes much more accurately than the basoc GoM model.

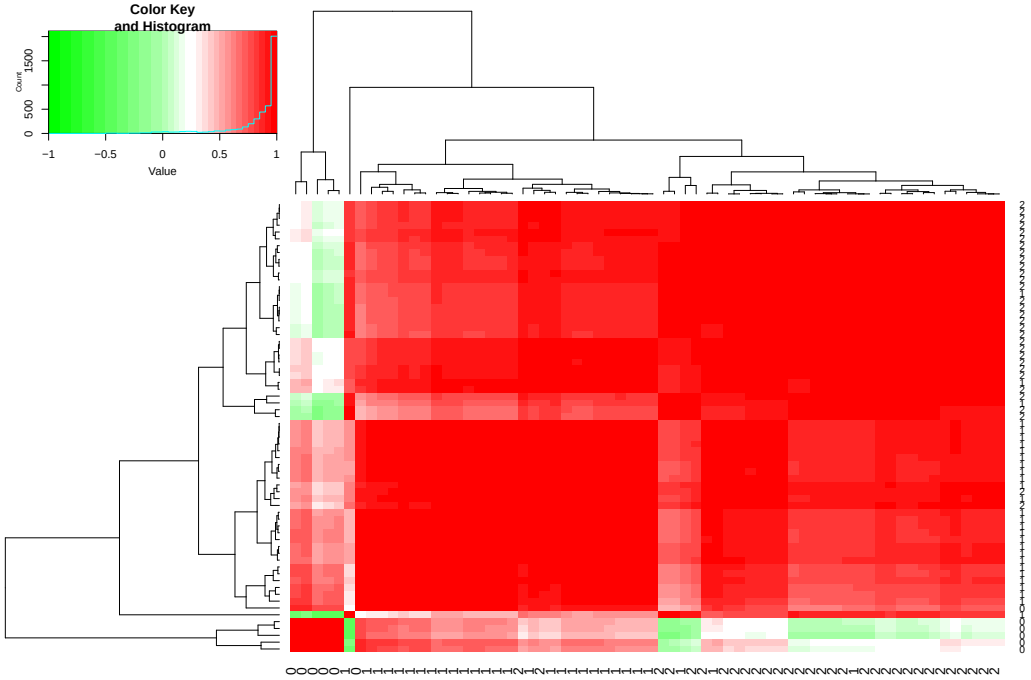
From Figure 4.2a it is clear that *Cluster-seq* does a better job at finding the cluster centers. While the "green" clusters in both approaches pick up the splice junction proposed in Pickrell et al. (2010), the one in the basic GoM model overlaps with the "red" and "blue" clusters at all three exons, making biological inference less tractable. Furthermore, the "red" and "blue" clusters in the basic GoM model are nearly identical and non-identifiable. This is reflected in the admixture plot for the basic GoM model as shown Figure 4.2b. Although there is decreasing membership in the "green" cluster as we move from the minor homozygotes to the heterozygotes to the major homozygotes, the plot also reveals considerable variation in membership proportions among samples within the same genotype class, making it difficult to observe similarities between samples. In contrast, the three clusters estimated by *Cluster-seq* are immediately distinguishable from one another, with the green and blue clusters capturing the variation in expression between genotype classes resulting from the sQTL. Figure 4.2b shows that the three genotype classes are clearly separated via the membership proportions estimated by *Cluster-seq*, which is highly encouraging given that we did not use the genotype labels in running *Cluster-seq*.

To further explore how well the samples group by genotype class given the estimates from both clustering approaches, Figure 4.3 plots the correlation heatmaps for each pair of samples, where the correlations are computed based on the membership proportions $\boldsymbol{\pi}$. Dendrograms derived from hierarchical clustering of the membership proportions are also plotted along the left and top of the heatmaps for better visualization. We observe that *Cluster-seq* achieves impressive categorization of the samples by genotype class, while such separation is absent using estimates derived

from the basic GoM model.



(a)



(b)

Figure 4.3: Pairwise correlation heatmaps for both the basic GoM model (panel (a)) and *Cluster-seq* (panel (b)), with $K = 3$. Each grid on the heatmap denotes the between sample correlation of the cluster memberships π, color-coded according to the amount of correlation. A hierarchical clustering of the correlations further reveals (dis)similarities between samples. The samples for *Cluster-seq* are categorized according to their genotype classes much more accurately than using the basic GoM model.

In practice, such clustering approaches are extremely helpful when the true genetic variant (sQTL in this case) associated with the estimated clusters is unknown. In this case, we perform a simple association analysis between SNPs and the estimated clusters from *Cluster-seq*; Figures 4.4 and 4.5 plot the correlation between cluster memberships for each estimated cluster mean and genoptype labels for each SNP within gene *OAS1*, for the basic GoM model and *Cluster-seq* respectively. In the correlation plots, the vertical red dashed line denotes rs10774671, and the blue dot represents the most highly correlated SNP with that cluster. We observe that the most associated SNP for each cluster center estimated using *Cluster-seq* is indeed rs10774671, confirming the effect of the sQTL on gene expression at this portion of the gene. On the other hand, the SNPs associated with the "red" and "blue" clusters estimated using the basic GoM model are not rs10774671, and indeed, quite far from rs10774671. These results suggest overall that *Cluster-seq* has the potential to be a more effective dimension clustering tool than the basic GoM model.

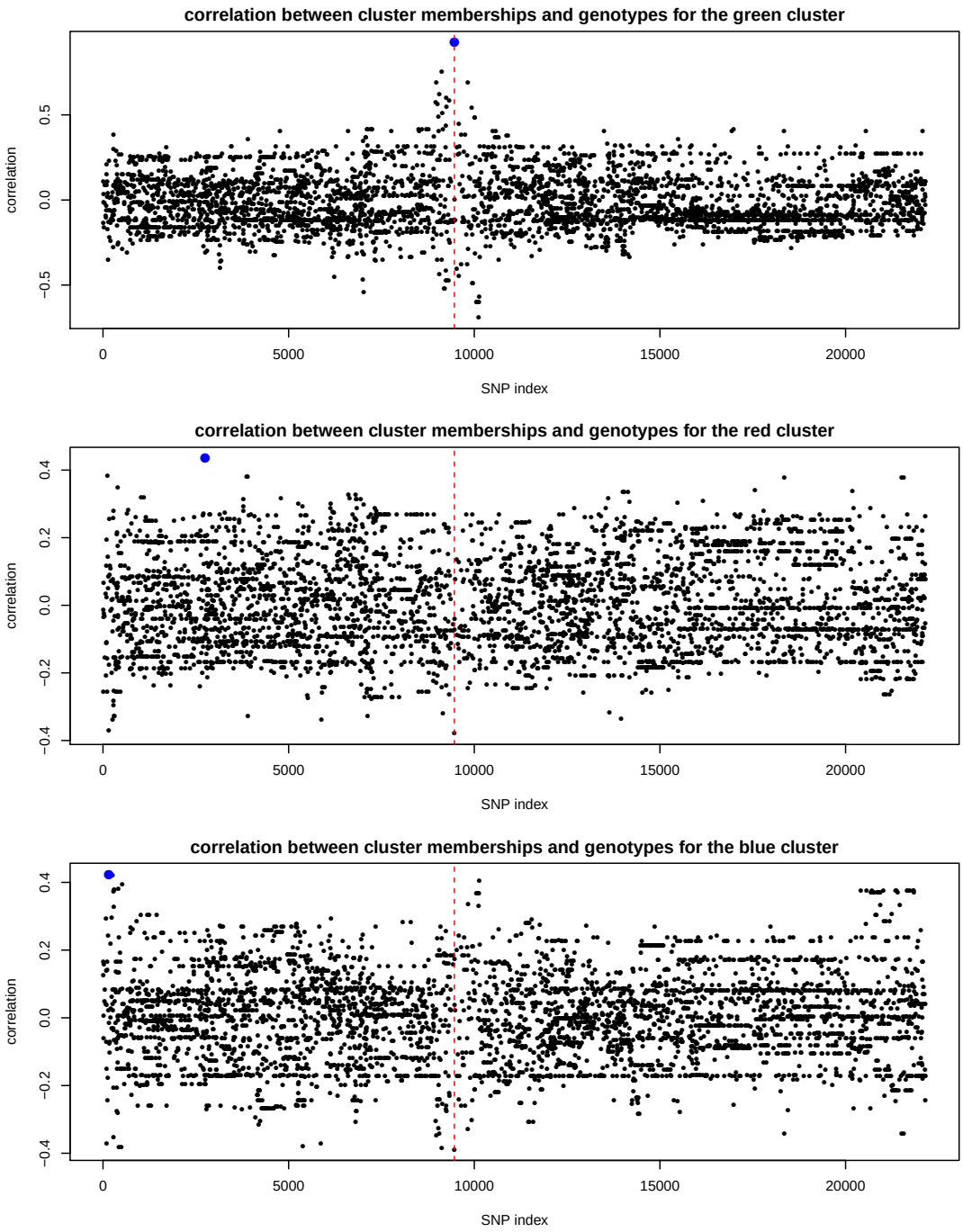


Figure 4.4: Correlation plots between cluster memberships estimated using the basic GoM model ($K = 3$) and genotype classes for each SNP located within and near gene *OAS1*. The three plots correspond to the three clusters. For each cluster, the SNP most significantly associated with the cluster memberships is colored in blue, while the location of rs10774671 is marked by the vertical dashed red line. We observe that the most significantly associated SNP is rs10774671 only for the green cluster, but not for the other two, and is in fact no where near to the physical location of rs10774671.

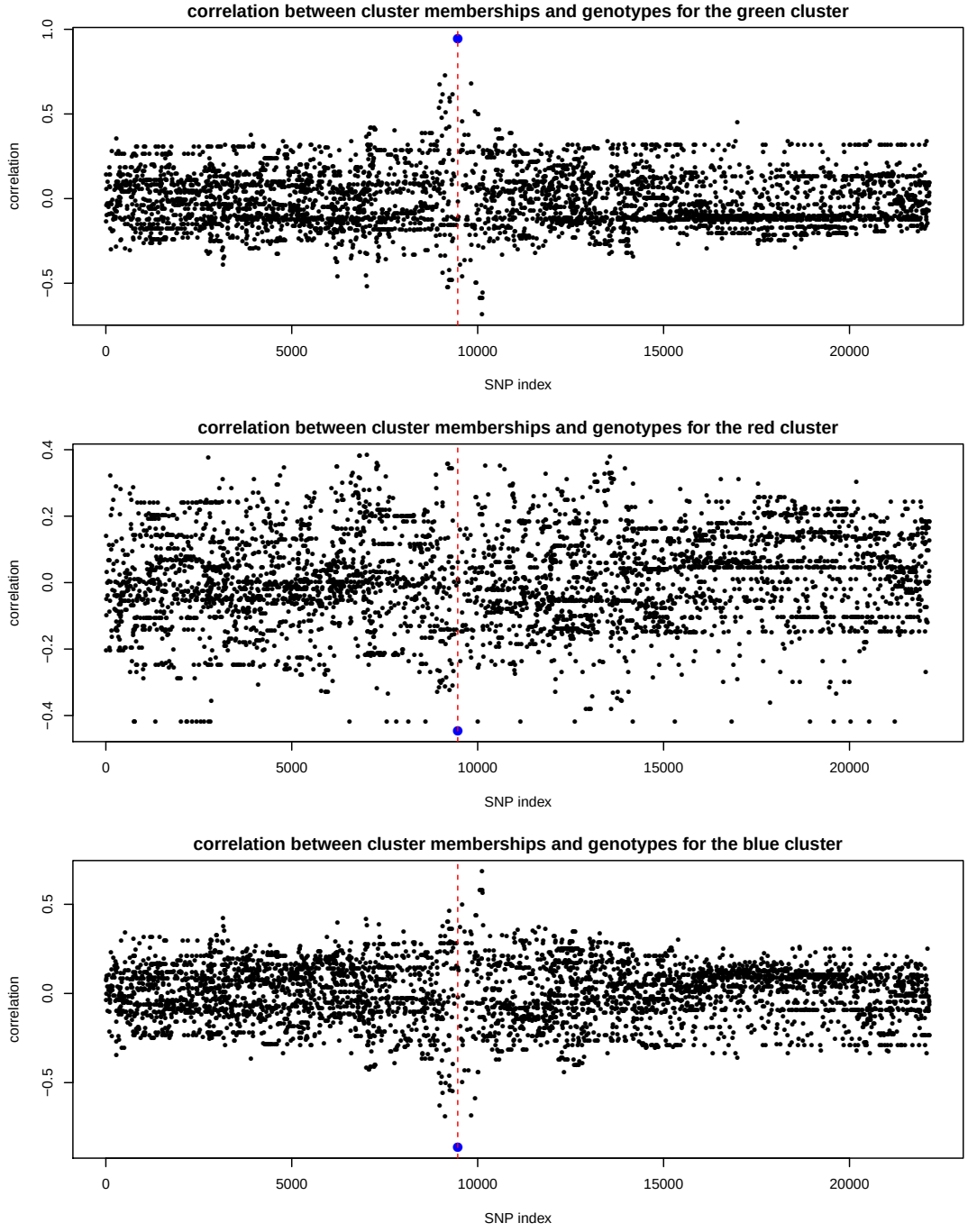


Figure 4.5: Correlation plots between cluster memberships estimated using *Cluster-seq* ($K = 3$) and genotype classes for each SNP located within and near gene *OAS1*. The three plots correspond to the three clusters. For each cluster, the SNP most significantly associated with the cluster memberships is colored in blue, while the location of rs10774671 is marked by the vertical dashed red line. We observe that the most significantly associated SNP is rs10774671 for all three clusters.

Of course, one might also wish try out different values of K to see if other underlying patterns can be revealed. For $K = 2$ and $K = 4$, figures of cluster means, admixture proportions, correlation heatmaps as well as association analyses can be found in Appendix A.8. The qualitative conclusions are mostly similar to $K = 3$, with *Cluster-seq* being more reliable than the basic GoM model when looking for underlying patterns in this dataset. One additional finding for $K = 4$ is that the most associated SNPs with the "red" and "cyan" clusters estimated by *Cluster-seq* are no longer rs10774671. On the other hand, the membership proportion in the "cyan" cluster is virtually identical for all the samples, making it somewhat of a "redundant" cluster, resulting in rs10774671 not being the most associated SNP. This in turn weakens the association between certain SNPs and the other clusters such as the "red" cluster. As such, it is not yet clear if this finding has much biological significance when $K = 4$. Choosing the number of clusters K in *Cluster-seq* is therefore an interesting but challenging direction for future work.

4.4 Case study: analysis of RNA-seq data from GTEX project

Having established the potential of *Cluster-seq* in Sections 4.3, we now apply it to RNA-seq data from the GTEx (**G**enotype-**T**issue **Ex**pression) project. Specifically, version 6 of the GTEx project collected RNA-seq data from tissue cells in 551 individuals, with up to 54 different tissue types for each individual. Dey et al. (2016) explored the relationship between samples from different tissues types by running a GoM model on gene-level expression (as measured by RNA-seq counts), and sum-

marized their findings elegantly using an admixture plot. This is shown in Figure 1 of Dey et al. (2016).

Due to storage limit however, we are not able to obtain raw RNA-seq reads for the entire GTEx project, and have to choose a subset of tissues and genes to focus on. As such, we downloaded data for 8 of the 9 tissues (Adipose, Artery, Heart, Lung, Muscle, Nerve, Skin and Thyroid) from the GTEx pilot study GTEx Consortium (2015), which appear to be the most interesting tissues for an initial analysis. Whole Blood was excluded from the analysis because it was a partial outlier (see GTEx Consortium (2015)), showing much fewer transcribed regions and much more heterogeneity in gene expression as compared to the other tissues. Furthermore, in an attempt to capture structural variations between expression profiles, we focus our attention on the 21 genes that have the highest average expression levels.

Having already demonstrated the advantages of *Cluster-seq* over the basic GoM model in Sections 4.2 and 4.3, we will show results only for *Cluster-seq* here. To explore different possible structural relationships between samples, we run *Cluster-seq* with $K = 2, 3, 4, 5$. Figures 4.6-4.8 show the estimated cluster means and corresponding admixture plots for three representative genes *RPS13* (hg19 chr11:17095938-17099220), *PSAP* (hg19 chr10:73576054-73611082) and *GPX3* (hg19 chr5:150399998-150408554), using $K = 3, 4, 3$ respectively. Note that we only show plots corresponding to the smallest values of K which reveal sufficient structure in the data. Plots for all the genes using all values of K are shown in Appendix A.9.

While these figures reveal many interesting structures and intricate relationships between samples and tissues, we highlight some of the more important observations

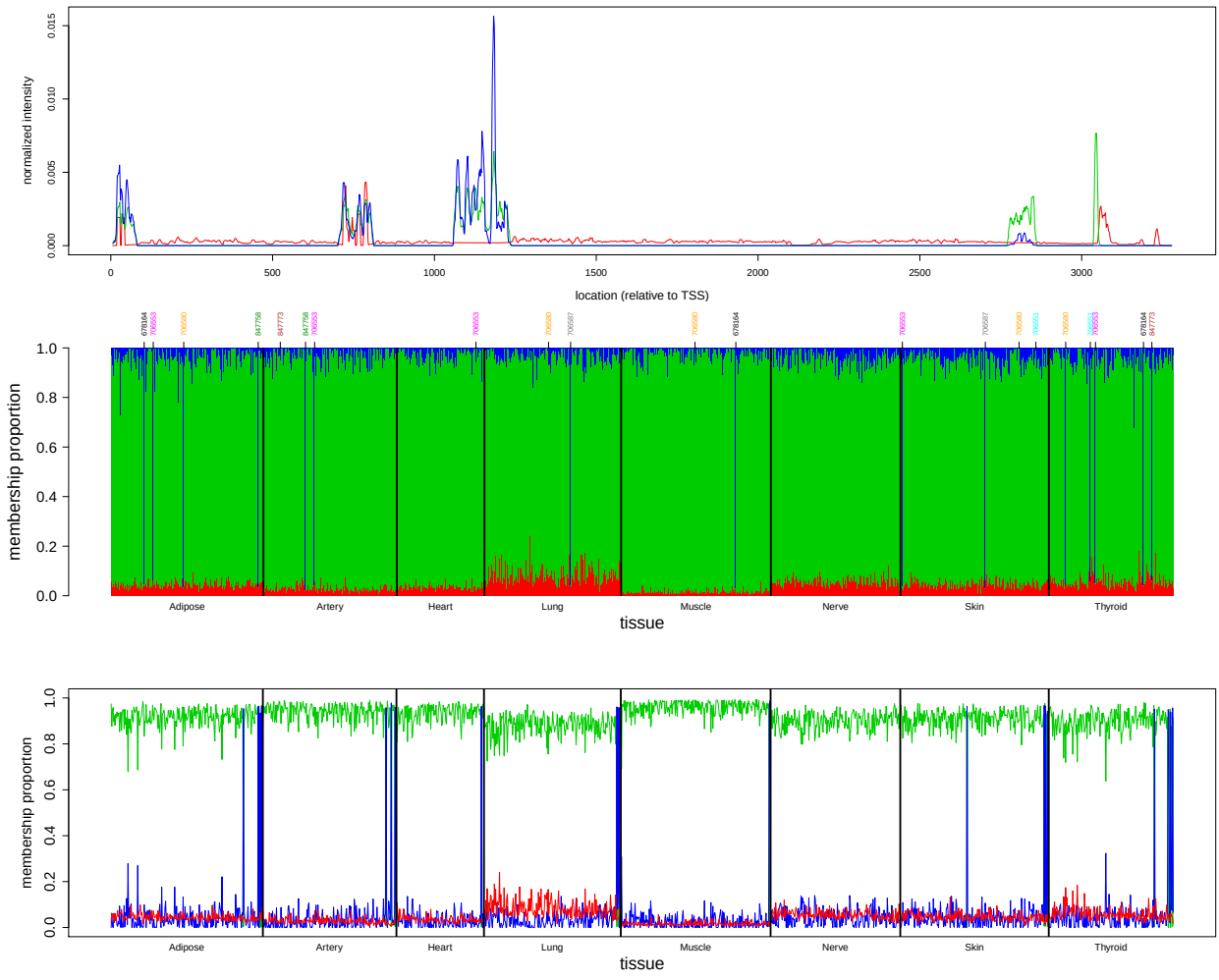


Figure 4.6: Results from running *Cluster-seq* on RNA-seq data from gene *RPS13* with $K = 3$. Top plot shows the estimated cluster means, while middle plot shows the corresponding admixture plot. For this gene, we also label the samples which belong almost exclusively to the blue cluster using their individual IDs (middle plot). Same individuals are labeled using the same color. Bottom plot shows the membership proportions represented as line graphs, where the samples within each tissue are ordered ascendingly by their normalized total expression. We note that the red cluster corresponds to intronic reads, while the blue cluster has no expression at the last two exons, revealing possible alternative splicing patterns. Furthermore, the individuals belonging almost entirely to the blue cluster may have certain genetic variants associated with this cluster.

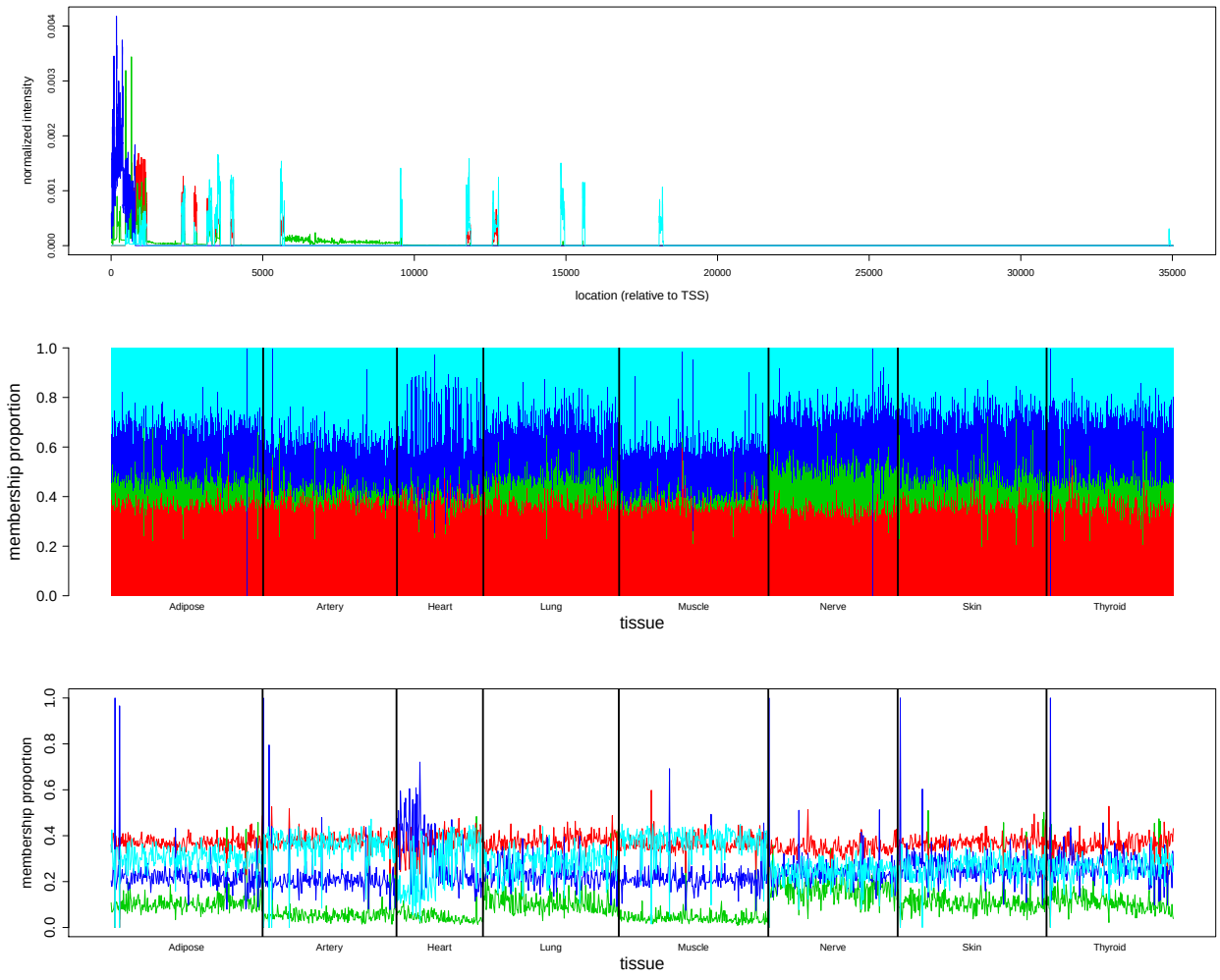


Figure 4.7: Results from running *Cluster-seq* on RNA-seq data from gene *PSAP* with $K = 4$. Top plot shows the estimated cluster means, while middle plot shows the corresponding admixture plot. Bottom plot shows the membership proportions represented as line graphs, where the samples within each tissue are ordered ascendingly by their normalized total expression. We note that the green cluster corresponds to intronic reads, while the other clusters show no expression in the last few exons of the gene, revealing possible alternative splicing patterns. The presence of intronic reads between the two exons located about 5500bp-9500bp away from the transcription start site captures intronic retention. Certain individuals belong almost entirely to the blue cluster, while others show remarkably high membership in the green cluster.

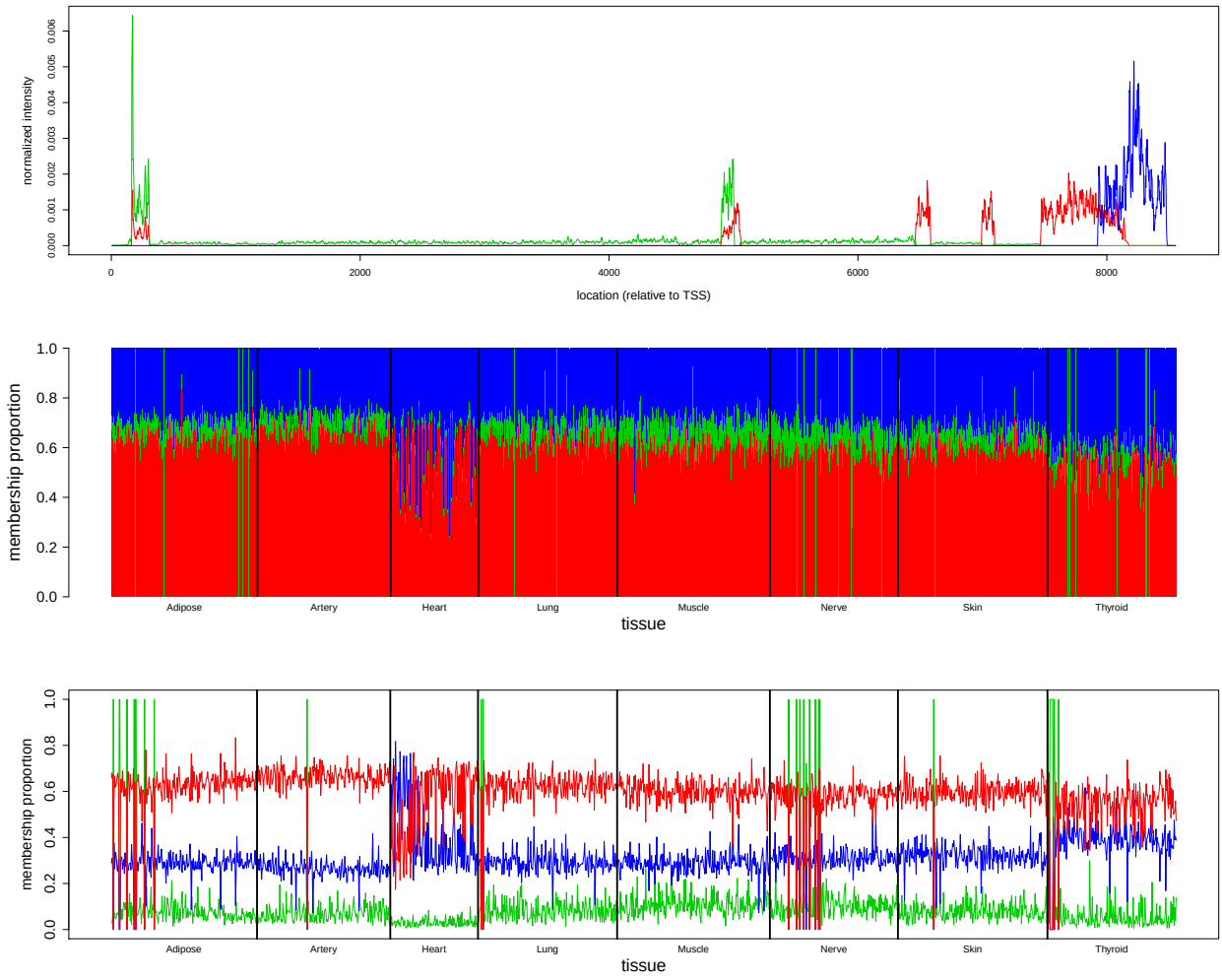


Figure 4.8: Results from running *Cluster-seq* on RNA-seq data from gene *GPX3* with $K = 3$. Top plot shows the estimated cluster means, while middle plot shows the corresponding admixture plot. Bottom plot shows the membership proportions represented as line graphs, where the samples within each tissue are ordered ascendingly by their normalized total expression. We note that the green cluster corresponds to intronic reads, while the blue cluster shows expression only in the last exon. Certain individuals belong almost entirely to the green cluster. Samples from Heart shows interesting variation in the proportion of membership to the blue cluster.

here.

- As we might expect, cluster membership proportions tend to show heterogeneity between tissues (although the difference can be subtle between some tissues). This is true for 14 out of the 21 genes analyzed, including genes *RPS13*, *PSAP* and *GPX3* shown in Figures 4.6-4.8, demonstrating that different tissues can vary at the transcriptional level despite the (nearly) identical DNA between cells from different tissues.

- Although different tissues tend to have varying membership proportions, expression profiles for some tissues are more similar at certain genes compared to others. At gene *GPX3* (Figure 4.8) for example, Lung and Muscle show similar expression profiles, but Heart is quite different from these two tissues. At gene *RPS13* (Figure 4.6) however, Heart and Muscle are much more similar to each other, with Lung being different from both these tissues. This goes to show that seemingly unrelated tissues can be similar to one another at one gene, yet completely different at another. More careful analysis of such patterns may reveal more about the underlying regulatory mechanisms for the various tissue types.

- Different values of K also reveal different types of structure. In particular, we note that all tissues appear very similar for gene *RPS13* when run with $K = 2$, but show obvious differences when run with $K = 3, 4$ or 5 (see Figure **??**). The reason is because *Cluster-seq* with $K = 2$ has too few clusters to account for reads in the intronic regions, focusing instead on the (possibly more obvious)

structural differences in the exonic regions. As such, it is always useful in such types of analysis to explore different resolutions with which to analyze the data, which could reveal structures that might otherwise be hidden. A more systematic approach to choosing the value of K is desirable, and is a possible direction for future work.

- Looking at the expression profiles rather than gene-level expression allows us to better understand the patterns arising from different biological mechanisms. For example, differences between tissues for genes *GPX3* and *RPS13* are mainly due to variations in read intensity across (almost) all the intronic regions, possibly implying the presence of tissue-specific splicing mechanisms. On the other hand, differences between tissues for gene *PSAP* are mostly at the exonic regions,.

- Individual specific effects are quite apparent for many of the tissues in this analysis. Samples from the same individual may share similar membership proportions across tissues, and belong almost exclusively to one or two clusters. As an example, subject IDs are color-labeled in Figure 4.6. We observe in this particular example that the last two exons in gene *RPS13* are not transcribed for the labeled individuals. However, these individual specific effects may not apply to all tissues. In *RPS13* for example, no sample in Nerve belongs almost entirely to the blue cluster, even though samples from the same individual in other tissues exhibit such behavior. Furthermore, these particular samples also tend to be either among the most or among the least expressed samples within each tissue, as shown by the bottom plots in Figures 4.6 and 4.8. As an

additional step, combining genotype information for these individuals with the estimated membership proportions may reveal a great deal about the genetic variants associated with the splicing mechanisms revealed by the clusters, such as the sQTL described in Section 4.3, although a quick analysis of *cis*-SNPs for *RPS13* failed to capture any single SNP responsible for such behavior.

- Membership proportions can also vary in a systematic manner within each tissue. In Heart for example, Figures 4.6 and 4.7 both reveal that membership proportions for one cluster can have a positive correlation with normalized expression and thus grow with increasing expression. Correspondingly the membership proportions for another cluster would have a negative correlation with normalized expression and thus decrease with increasing expression.

While the preliminary results we presented here have already revealed a great deal about the structure underlying the data and their biological implications, there is potential for future work to uncover much more. Most notably, having the raw sequencing data for all available tissues and genes will be the first step in answering the many questions that remain. For example, we speculate that the structural similarities between related tissues (eg. different brain tissues) will be more evident for a much larger proportion of genes when compared with unrelated tissues. Running *Cluster-seq* on all 54 tissues with different choices of K will likely reveal more fine level structures, particularly among related tissues. Including all genes will also add another dimension to the picture by possibly revealing structural relationships between genes, in addition to tissues. Other possibilities for future exploration includes biological interpretation of the results we have presented here by using gene anno-

tation information, or finding associated genetic variants by making use of genotype labels for SNPs near/within each gene.

CHAPTER 5

DENOISING SIGNALS WITH HETEROSKEDASTIC GAUSSIAN ERRORS[1]

5.1 Overview

In Chapter 2 we have seen the shrinkage procedure *ash* being applied to the task of denoising Poisson signals. In general, shrinkage and sparsity play a key role in many areas of modern statistics, including, for example, high-dimensional regression (Tibshirani (1996)), covariance or precision matrix estimation (Bickel and Levina (2008)), multiple testing (Efron (2004)) and signal denoising (Donoho and Johnstone (1994, 1995)). One attractive way to achieve shrinkage and sparsity is via Bayesian or Empirical Bayes (EB) methods (e.g.Efron and Tibshirani (2002); Johnstone and Silverman (2005); Clyde and George (2000); Daniels and Kass (2001)). However, such Bayesian methods are usually perceived to require context-specific implementations, and this overhead can limit their use in practice. In this chapter our goal is to demonstrate the flexibility, efficacy, and convenience of *ash* by applying it to another signal denoising problem, specifically for Gaussian observations with heteroskedastic errors.

As with the well studied problem of signal denoising under a homoskedastic Gaussian assumption, we consider estimating a spatially-structured mean $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_T)$ and also the corresponding spatially-structured variance $\boldsymbol{\sigma} = (\sigma_1^2, \ldots, \sigma_T^2))$, from

1. Parts of this chapter are included in a preprint https://arxiv.org/abs/1605.07787.

observations $\boldsymbol{Y} = (Y_1, \ldots, Y_T)$, where $t = 1, \ldots, T$ indexes location in a one-dimensional space, such as time. Specifically, suppose we have

$$\boldsymbol{Y} = \boldsymbol{\mu} + \boldsymbol{\epsilon} \tag{5.1}$$

where $\boldsymbol{\epsilon} \sim N_T(\mathbf{0}, D)$ with D the diagonal matrix with diagonal entries $(\sigma_1^2, \ldots, \sigma_T^2)$. We consider, in turn, the problems of estimating $\boldsymbol{\mu}$ when $\boldsymbol{\sigma}$ is known; estimating $\boldsymbol{\sigma}$ when $\boldsymbol{\mu}$ is known; and finally the typical case of estimating $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ when both are unknown.

Estimating $\boldsymbol{\mu}$ with $\boldsymbol{\sigma}$ known

We first transform the data using a multi-scale transformation, specifically a discrete wavelet transform. This involves pre-multiplying $\boldsymbol{Y}$ by an orthogonal $T \times T$ matrix W that depends on the wavelet basis chosen. Pre-multiplying (5.1) by W yields

$$W\boldsymbol{Y} = W\boldsymbol{\mu} + W\boldsymbol{\epsilon} \tag{5.2}$$

which we write

$$\tilde{\boldsymbol{Y}} = \tilde{\boldsymbol{\mu}} + \tilde{\boldsymbol{\epsilon}}, \tag{5.3}$$

where $\tilde{\boldsymbol{\epsilon}} \sim N_T(0, WDW')$.

Next we use *ash* to obtain (posterior mean) shrinkage estimates $\hat{\tilde{\boldsymbol{\mu}}}$ for $\tilde{\boldsymbol{\mu}}$, which

we then reverse transform to estimates for μ:

$$\hat{\mu} := W^{-1}\hat{\tilde{\boldsymbol{\mu}}}. \tag{5.4}$$

Applying *ash* requires only point estimates for the elements of $\tilde{\boldsymbol{\mu}}$, and corresponding standard errors, which we obtain from the marginals of (5.3):

$$\tilde{Y}_j|\tilde{\mu}_j, s_j^2 \sim N(\tilde{\mu}_j, \omega_j^2), \tag{5.5}$$

where

$$\omega_j^2 := \sum_{t=1}^{T} \sigma_t^2 W_{jt}^2. \tag{5.6}$$

Thus to obtain $\hat{\tilde{\boldsymbol{\mu}}}$ we apply *ash* to the estimates $\hat{\beta}_j := \tilde{Y}_j$ with standard errors $\hat{s}_j = \omega_j$. (In practice it is important to group the wavelet-transformed observations $\tilde{Y}_j$ by their resolution level before shrinking; see note below.)

In focusing on the marginals (5.5) we are ignoring correlations among the $\tilde{Y}_j$, and this is the primary simplification here. We are not alone in making this simplification; see Silverman (1999) for example.

The above outlines the basic strategy, but there are some important additional details:

1. Rather than use a single wavelet transform, we use the "non-decimated" (translation invariant) wavelet transform (Appendix A.2), which averages results over all T possible rotations of the data (effectively treating the observations as coming from a circle, rather than a line). Although not always necessary, this is

a standard trick to reduce artifacts that can occur near discontinuities in the underlying signal (eg. Coifman and Donoho (1995)).

2. The non-decimated wavelet transform yields T wavelet coefficients (transformed values of $\boldsymbol{Y}$) at each of $J = \log_2(T)$ resolution levels. We apply *ash* separately to the T wavelet coefficients at each resolution level, so that a different distribution g for the $(\tilde{\mu}_j)$ is estimated for each resolution. This is the usual way that EB approaches are applied in this context (e.g. Johnstone and Silverman (2005)) and indeed is crucial because the underlying distribution g will vary with resolution (because smoothness of $\boldsymbol{\mu}$ will vary with resolution).

3. Although we have presented the wavelet transform as a matrix multiplication, which is an $o(T^2)$ operation, in practice both the wavelet transform and the inverse transform are implemented using efficient algorithms (Beylkin (1992); Coifman and Donoho (1995)), implemented in the `wavethresh` package (Nason (2013)), taking only $O(T \log_2 T)$ operations.

Estimating $\boldsymbol{\sigma}$ with $\boldsymbol{\mu}$ known

To estimate the variance $\boldsymbol{\sigma}$ we apply wavelet shrinkage methods to the squared deviations from the mean, similar to the strategies of Delouille et al. (2004) and Cai and Wang (2008). Specifically, define

$$Z_t^2 = (Y_t - \mu_t)^2 \tag{5.7}$$

and note that $\mathbb{E}(Z_t^2) = \sigma_t^2$, so estimating $\boldsymbol{\sigma}$ is now a mean estimation problem. We tackle this using the mean estimation procedure above, effectively treating the wavelet-transformed values $(W\boldsymbol{Z}^2)_t$ (where $\boldsymbol{Z}^2 \equiv (Z_1^2, ..., Z_T^2)$) as Gaussian when really they are linear combinations of χ_1^2 random variables. This requires an estimate of the variance of Z_t^2: we use $\frac{2}{3}Z_t^4$, which is an unbiased estimator of the variance. (If $Z^2 \sim \sigma^2\chi_1^2$, then $\mathbb{E}(Z^4) = 3\sigma^4$, and $\mathbb{V}(Z^2) = 2\sigma^4$.)

Despite the approximations made here, we have found this procedure to work well in practice in most cases, perhaps with a tendency to oversmooth quickly-varying variance functions.

Estimating $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ jointly

We iterate the above procedures to deal with the (typical) case where both mean and variance are unknown. To initialize we estimate the variance $\boldsymbol{\sigma}^2$ using

$$\hat{\sigma}_t^2 = \frac{1}{2}\left((Y_t - Y_{t-1})^2 + (Y_t - Y_{t+1})^2\right) \tag{5.8}$$

where $Y_0 \equiv Y_n$ and $Y_{T+1} \equiv Y_1$ (equivalent to putting the observations on a circle).

Then we iterate:

1. Estimate $\boldsymbol{\mu}$ as if $\boldsymbol{\sigma}^2$ is known (with the value obtained from the previous step).
2. Estimate $\boldsymbol{\sigma}^2$ as if $\boldsymbol{\mu}$ is known (with the value obtained by the previous step); return to 1.

We cannot guarantee that this procedure will converge, but in our simulations we

found that two iterations of steps 1-2 reliably yielded accurate results (so the full procedure consists of initialize + Steps 1-2-1-2).

5.2 Simulation results

As with the Poisson case, we conducted extensive simulation studies to compare the performance of *smash* with existing approaches. In particular, this set of simulations were conducted within a Dynamical Statistical Comparison
(DSC; https://github.com/stephens999/dscr) framework, which was designed to facilitate large-scale comparisons between statistical methods. The DSC framework for the set of simulations conducted in this section is contained in the GitHub repository dscr-smash (https://github.com/zrxing/dscr-smash). For more detailed descriptions of what the repository does, instructions on reproducing the results in this section, as well as guidelines for extending the simulation study under the DSC framework, refer to README.md within the repository.

We focus initially on the homoskedastic case, modelling our simulation study after Antoniadis et al. (2001). Specifically we used many of the same test functions, a variety of signal lengths (T), and two different signal to noise ratios (SNRs). We compare *smash* with Translation Invariant (TI) thresholding (Coifman and Donoho (1995)), which was one of the best-performing methods in Antoniadis et al. (2001), and with the Empirical Bayes shrinkage procedure Ebayesthresh (Johnstone and Silverman (2005)). All methods were applied using the Symmlet8 wavelet basis (Daubechies (1992)).

Figure 5.1 compares the mean integrated squared errors (MISEs) of the meth-

ods for the "Spikes" mean function, with SNR=3 and T=1024. We applied *smash* in three ways, the first (*smash*) estimating the variance function allowing for heteroskedasticity, the second (*smash*-homo) estimating the variance assuming homoskedasticity and the third(*smash* true variance) using the true variance function , which could be viewed as a "gold standard". All three versions of *smash* outperform both EbayesThresh and TI-thresholding. The different *smash* versions perform similarly, demonstrating that in this case there is little cost in allowing for heteroskedasticity when the truth is homoskedastic. We obtained similar results for other mean functions, SNRs and sample sizes (see Appendix A.10.2).

Turning to heteroskedastic errors, we again compare *smash* with EbayesThresh (which assumes homoskedastic variance) and TI-thresh. For TI-thresh we considered three different ways of estimating the heteroskedastic variance: RMAD (Gao (1997)), the *smash* estimated variance, and the true variance. (TI-thresh with homoskedastic variance performed very poorly; see Appendix A.10.2.) Figure 5.2 shows results for two sets of test functions: the "Spikes" mean function with the "Clipped Blocks" variance function and the "Corner" mean function with "Doppler" variance function, both with SNR=3 and $T = 1024$.

To summarize the main patterns in Figure 5.2:

1. *smash* outperforms all TI-thresh variants (including TI with the true variance).

2. *smash* performs almost as well when estimating the variance as when given the true variance.

3. Allowing for heteroskedasticity within *smash* can substantially improve accu-

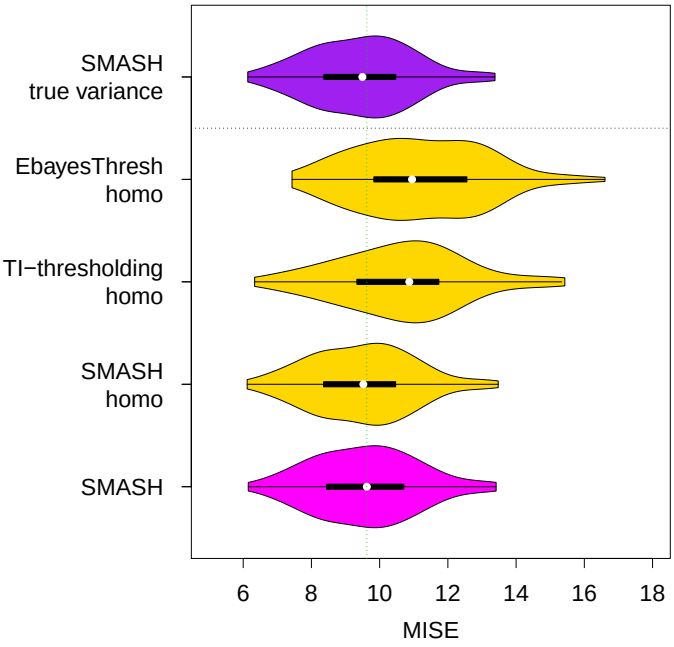


Figure 5.1: Comparison of accuracy (MISE) of estimated mean curves for data simulated with homoskedastic Gaussian errors. The figure shows violin plots of MISEs for (from bottom to top) *smash*, *smash* with homoskedastic assumption, TI-thresholding with homoskedastic assumption, Ebayesthresh with homoskedastic assumption, and *smash* with known true variance. Colors highlight certain features of a method: yellow for methods that assume homoskedastic variance; dark purple for methods that are provided the true variance; magenta for the default *smash* method that estimates both mean and variance. Smaller MISE implies better performance; dashed green line indicates the median MISE for *smash*. *smash* outperforms both TI-thresholding and Ebayesthresh, and *smash* with estimated variance performs nearly as well as with true variance.

racy of mean estimation (compare *smash* with *smash*-homo and EbayesThresh).

4. TI-thresh performs considerably better when used with the *smash* variance estimate than with the RMAD variance estimate.

These main patterns hold for a variety of different mean and variance functions, SNRs, and sample sizes (see Appendix A.10.2). Some variance functions are harder to estimate than others (e.g. the "Bumps" function), and in these cases providing methods the true variance can greatly increase accuracy compared with estimating the variance. As might be expected, the gain in allowing for heteroskedastic variance tends to be greatest when the variance functions are more volatile.

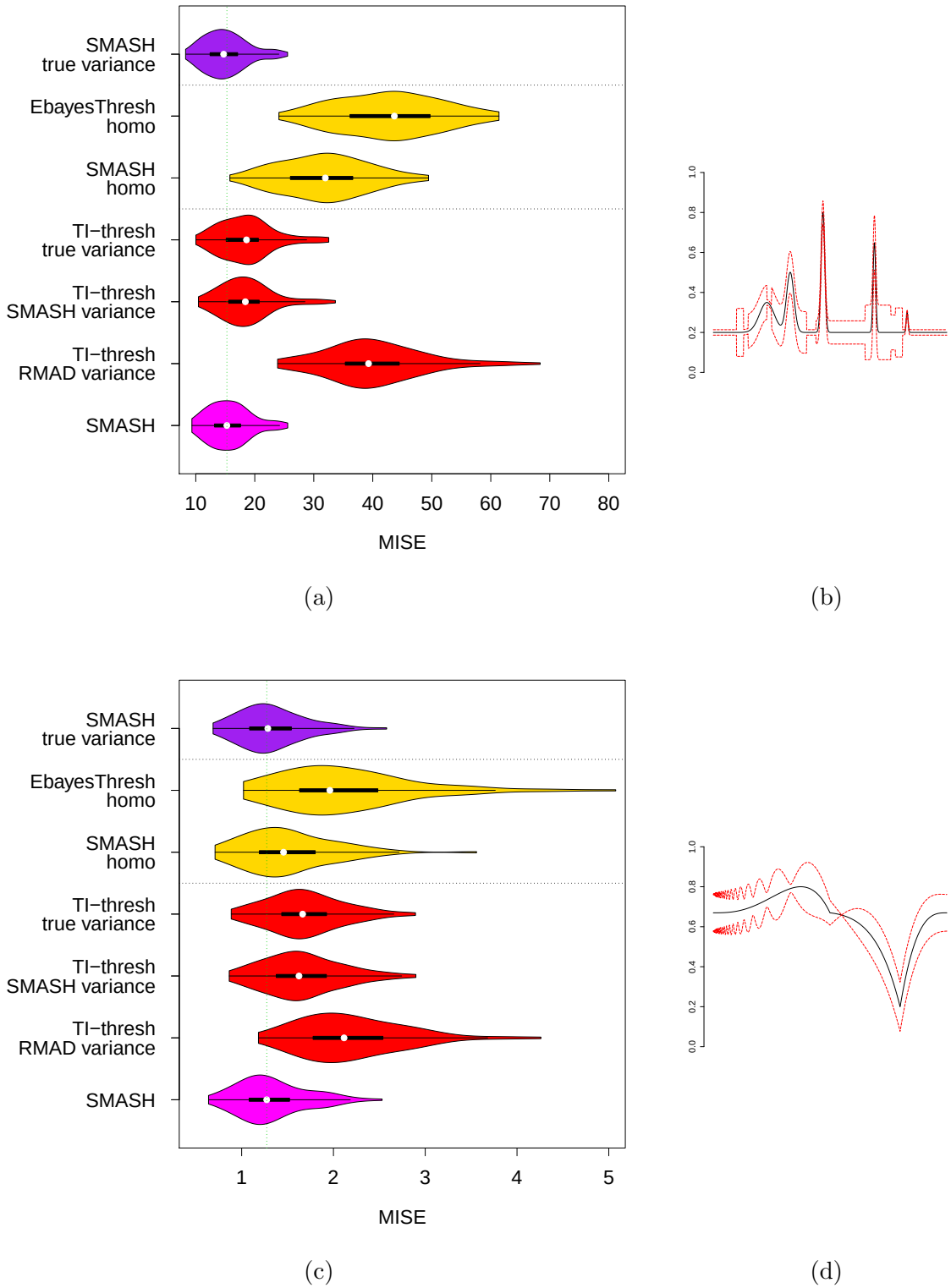


Figure 5.2: Comparison of accuracy (MISE) of estimated mean curves, for data simulated with heteroskedastic Gaussian errors. Panels (a) and (c) show violin plots of MISEs for various methods on two sets of mean-variance functions, shown as mean ± 2 standard deviations in panels (b) and (d). In (a) and (c) TI-based methods are shown in red, while other colors are as in Figure 5.1. Dashed green line indicates *smash* median MISE.

One unusual feature of *smash* is that it performs joint mean and variance estimation. Indeed, we found no existing R packages for doing this. The only work we have found on wavelet-based variance estimation is Cai and Wang (2008), which applied a wavelet thresholding approach to first order differences. Previous related non-wavelet-based work includes Fan and Yao (1998), which estimates the variance by smoothing the squared residuals using local polynomial smoothing, Brown and Levine (2007), which uses difference-based kernel estimators, and Menictas and Wand (2015), which introduces a Mean Field Variational Bayes (MFVB) method for both mean and variance estimation. In no case could we find publicly-available software implementations of these methods. However, we did receive code implementing MFVB by email from M. Menictas, which we use in our comparisons here.

The MFVB method is based on penalized splines, and so is not well suited to many standard test functions in the wavelet literature, which often contain "spiky" local features not well captured by splines. Hence, we compared *smash* and MFVB on some smoother mean and variance (standard deviation) functions, specifically scenario A in Figure 5 from Menictas and Wand (2015) (Figure 5.3), using scripts kindly provided by M. Menictas.

We simulated data under two different scenarios:

1. We generated $T = 500$ independent (X_t, Y_t) pairs, with $X_t \sim$ Uniform(0,1), and $Y_t|X_t = x_t \sim N(m(x_t), s(x_t)^2)$ where $m(\cdot)$ and $s(\cdot)$ denote the mean and standard deviation functions (Figure 5.3). We measured performance by the MSE evaluated at 201 equally spaced points on (X_{min}, X_{max}) for both the mean and the standard deviation.

2. We generated $T = 1024$ independent (X_t, Y_t) pairs, with the X_t's (deterministically) equally spaced on (0,1), and $Y_t|X_t$ as above. Performance is measured by MSE evaluated at the 1024 X_t's for both the mean and the standard deviation.

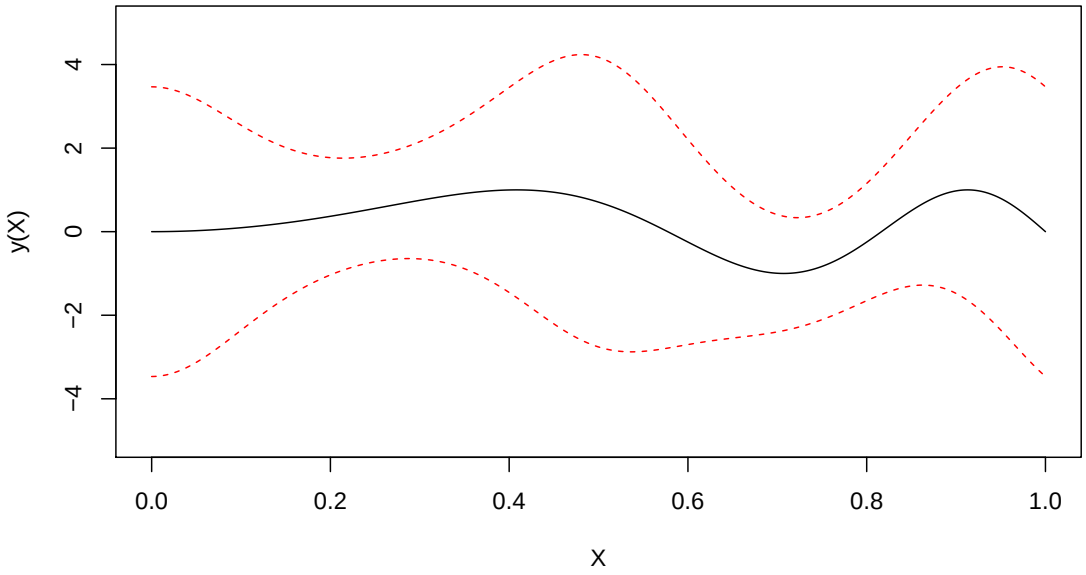


Figure 5.3: The mean function $m(x)$ ± 2 standard deviations $s(x)$ used in simulations comparing *smash* and MFVB. These functions correspond to mean and standard deviation functions (A) in Figure 5 from Menictas and Wand (2015).

The first scenario presents some issues for *smash* because the number of data points is not a power of two, nor are the points equally-spaced. To deal with the first issue, following standard ideas in the wavelet literature, we first mirrored the data about the right edge and extract the first $2^{\lfloor \log_2(2T) \rfloor}$ sample points, so that the number of data points in the new "dataset" is a power of two, and the mean curve is continuous at the right edge of the original data. To further ensure that the input

to *smash* is periodic, we reflected the new dataset about its right edge and used this as the final input. To deal with the second issue we follow the common practice of treating the observations as if they were evenly spaced (see Sardy et al. (1999) for discussion). To estimate the original mean and variance functions, we extract the first T points from the *smash* estimated mean and variance. To evaluate MSE at the 201 equally spaced points (Scenario 1) we use simple linear interpolation between the estimated points.

Table 5.1 shows mean MSEs over 100 independent runs for each scenario. Despite the fact that these simulation scenarios – particularly Scenario 1 – seem better suited to MFVB than *smash*, *smash* performs comparably or better than MFVB for both mean and variance estimation in both Scenarios.

	Scenario 1		Scenario 2	
	MSE (for mean)	MSE (for sd)	MSE (for mean)	MSE (for sd)
MFVB	0.0330	0.0199	0.0172	0.0085
smash	0.0334	0.0187	0.0158	0.0065

Table 5.1: Comparison of accuracy (MSE) of *smash* and MFVB for two simulation scenarios. True mean and sd functions are shown in Figure 5.3. In Scenario 1 the data are not equally spaced and not a power of 2; here *smash* is comparable to MFVB in mean estimation and more accurate for sd estimation. In Scenario 2 the data are equally spaced and a power of 2; here *smash* outperforms MFVB in both mean and sd estimation.

5.3 Case study: motorcycle acceleration data

To further illustrate the heteroskedastic Gaussian version of *smash*, we apply it to the motorcycle acceleration dataset from Silverman (1985). The data consist of 133 observations measuring head acceleration in a simulated motorcycle accident that is used to test crash helmets. The dependent variable is *acceleration* (in g), and

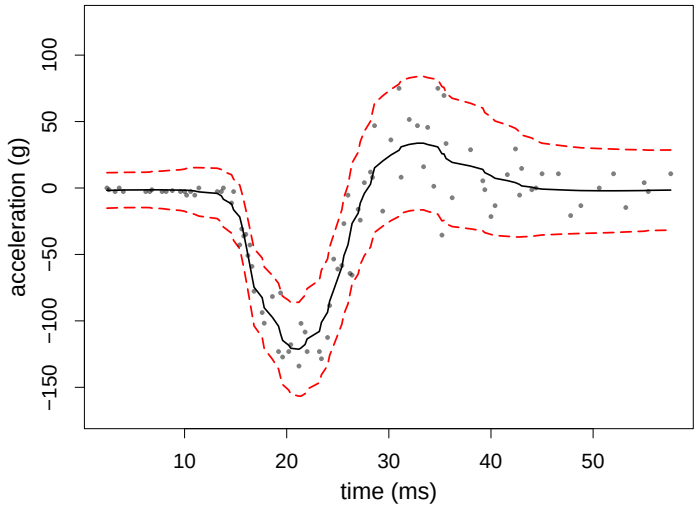


Figure 5.4: Results from fitting *smash* to the motorcycle acceleration data discussed in Silverman (1985). The figure shows the estimate mean curve (solid black line) with ±2 the estimated standard deviation curve (dashed red line).

the independent variable is *time* (in *ms*). To deal with repeated measurements, we take the median of the measurements for acceleration for any given *time* value. As with the procedure used when comparing *smash* against MFVB, we treat the data as if they are equally spaced although they are not. The fitted mean and variance curves (Figure 5.4) provide a visually appealing fit to the data, and were achieved without hand tuning of any parameters. This contrasts with results in Delouille et al. (2004) – which also uses a wavelet-based approach for heteroskedastic variance, but accounts for the unequal spacing of the data – which required the *ad hoc* removal of high-resolution wavelet coefficients to produce a visual appealing fit.

CHAPTER 6

CONCLUSION

In this dissertation we have proposed a fully model-based approach to deal with NGS data. While such high throughput sequencing technology has been around for more than a decade, statistical methods that actually tackle such massive data at high resolutions have only emerged in the very recent years. Even so, these methods are mostly developed with specific questions in mind, or make approximations that hold only under certain conditions. On the other hand, our approach circumvents these issues by first modeling the sequencing counts for a single sample, exploiting the spatial structure of sequencing data. Chapter 2 describes the multiscale framework and an adaptive shrinkage scheme designed for data with such spatial structure, as well as the potential biological uses for the multiscale framework even with just one sample.

With the basic model at hand, it can be adapted to answer more challenging and intriguing biological questions. In Chapter 3, we considered the question of estimating differences between groups of samples at the base pair resolution. While there exists several methods designed for this task, the assumptions underlying these methods often render them unsuitable for small sample sizes with extremely low counts. By incorporating the multiscale model into a GLM framework, our approach is able to handle this task satisfactorily. Furthermore, the same method can also be used to test for association with a quantitative covariate, and we have shown how this can be used to perform supervised learning of TF binding sites.

Chapter 4 extends the multiscale model in a different direction. By integrating it

into a grade-of-membership clustering scheme, our approach can be used as a powerful exploratory tool for learning about the intricate structures underlying massive biological datasets without having to incorporate any external information. This is particularly useful not only for finding patterns where specific biological inquiries might miss, but also for framing biological hypotheses in the first place. While we have applied our method to data from just the GTEx project, it has tremendous potential for uncovering patterns in massive amounts of NGS data that is available in many other existing projects.

Overall, we have demonstrated that our multiscale framework is extremely flexible and accurate, especially for small sample sizes and low counts. While we have applied our method to several datasets available from different projects and drawn interesting conclusions based on the results of the analyses, we believe that the framework developed in this dissertation can be applied to broader settings where many unanswered questions still remain.

REFERENCES

Anders, S. and W. Huber (2010, October). Differential expression analysis for sequence count data. *Genome biology 11*(10), R106+.

Antoniadis, A., J. Bigot, and T. Sapatinas (2001). Wavelet Estimators in Nonparametric Regression: A Comparative Simulation Study. *Journal of Statistical Software 6*(6).

Baraniuk, R. (1999). Optimal tree approximation with wavelets.

Besbeas, P., I. De Feis, and T. Sapatinas (2004, August). A Comparative Simulation Study of Wavelet Shrinkage Estimators for Poisson Counts. *International Statistical Review 72*(2), 209–237.

Beylkin, G. (1992). On the Representation of Operators in Bases of Compactly Supported Wavelets. *SIAM Journal on Numerical Analysis 29*(6), 1716–1740.

Bickel, P. J. and E. Levina (2008, December). Covariance regularization by thresholding. *The Annals of Statistics 36*(6), 2577–2604.

Brown, L. D. and M. Levine (2007, October). Variance estimation in nonparametric regression via the difference sequence method. *The Annals of Statistics 35*(5), 2219–2232.

Cai, T. T. and L. Wang (2008, October). Adaptive variance function estimation in heteroscedastic nonparametric regression. *The Annals of Statistics 36*(5), 2025–2054.

Clyde, M. and E. I. George (2000, January). Flexible Empirical Bayes estimation for wavelets. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 62*(4), 681–698.

Coifman, R. R. and D. L. Donoho (1995). Translation-invariant De-noising.

Daniels, M. J. and R. E. Kass (2001). Shrinkage Estimators for Covariance Matrices. *Biometrics 57*(4), 1173–1184.

Daubechies, I. (1992). *Ten lectures on wavelets*, Volume 61 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM).

Delouille, V., J. Simoens, and R. von Sachs (2004, September). Smooth Design-Adapted Wavelets for Nonparametric Stochastic Regression. *Journal of the American Statistical Association 99*(467), 643–658.

Dey, K. K., C. J. Hsiao, and M. Stephens (2016, May). Clustering RNA-seq expression data using grade of membership models. *bioRxiv*, 051631+.

Donoho, D. L. (1993). Nonlinear Wavelet Methods for Recovery of Signals, Densities, and Spectra from Indirect and Noisy Data. In *In Proceedings of Symposia in Applied Mathematics*, Volume 47, pp. 173–205. AMS.

Donoho, D. L. and I. M. Johnstone (1995, December). Adapting to Unknown Smoothness via Wavelet Shrinkage. *Journal of the American Statistical Association 90*(432), 1200–1224.

Donoho, D. L. and J. M. Johnstone (1994, September). Ideal spatial adaptation by wavelet shrinkage. *Biometrika 81*(3), 425–455.

Eddelbuettel, D., R. François, J. Allaire, J. Chambers, D. Bates, and K. Ushey (2011). Rcpp: Seamless R and C++ integration. *Journal of Statistical Software 40*(8), 1–18.

Efron, B. (2004). Large-Scale Simultaneous Hypothesis Testing: The Choice of a Null Hypothesis. *Journal of the American Statistical Association 99*(465), 96–104.

Efron, B. and R. Tibshirani (2002, June). Empirical bayes methods and false discovery rates for microarrays. *Genet. Epidemiol. 23*(1), 70–86.

Fan, J. and Q. Yao (1998). Efficient Estimation of Conditional Variance Functions in Stochastic Regression. *Biometrika 85*(3), 645–660.

Frazee, A. C., S. Sabunciyan, K. D. Hansen, R. A. Irizarry, and J. T. Leek (2014, July). Differential expression analysis of RNA-seq data at single-base resolution. *Biostatistics (Oxford, England) 15*(3), 413–426.

Fryzlewicz, P. and G. P. Nason (2001). Poisson intensity estimation using wavelets and the Fisz transformation.

Fryzlewicz, P. and G. P. Nason (2004, September). A Haar-Fisz Algorithm for Poisson Intensity Estimation. *Journal of Computational and Graphical Statistics 13*(3), 621–638.

Fusi, N. and J. Listgarten (2016). Flexible Modelling of Genetic Effects on Function-Valued Traits. In M. Singh (Ed.), *Research in Computational Molecular Biology*, Volume 9649 of *Lecture Notes in Computer Science*, pp. 95–110. Springer International Publishing.

Gao, H.-y. (1997). Wavelet Shrinkage Estimates For Heteroscedastic Regression Models.

Gart, J. J. and J. R. Zweifel (1967, June). On the bias of various estimators of the logit and its variance with application to quantal bioassay. *Biometrika 54*(1), 181–187.

GTEx Consortium (2015, May). Human genomics. The Genotype-Tissue Expression (GTEx) pilot analysis: multitissue gene regulation in humans. *Science (New York, N.Y.) 348*(6235), 648–660.

Guo, Y., S. Mahony, and D. K. Gifford (2012, August). High Resolution Genome Wide Binding Event Finding and Motif Discovery Reveals Transcription Factor Spatial Binding Constraints. *PLoS Comput Biol 8*(8), e1002638+.

Hesselberth, J. R., X. Chen, Z. Zhang, P. J. Sabo, R. Sandstrom, A. P. Reynolds, R. E. Thurman, S. Neph, M. S. Kuehn, W. S. Noble, S. Fields, and J. A. Stamatoyannopoulos (2009, April). Global mapping of protein-DNA interactions in vivo by digital genomic footprinting. *Nature methods 6*(4), 283–289.

Jiang, D., C. Tang, and A. Zhang (2004, November). Cluster Analysis for Gene Expression Data: A Survey. *IEEE Trans. on Knowl. and Data Eng. 16*(11), 1370–1386.

Johnstone, I. M. and B. W. Silverman (2005, August). Empirical Bayes selection of wavelet thresholds. *The Annals of Statistics 33*(4), 1700–1752.

Koenker, R. and I. Mizera (2014). Convex optimization, shape constraints, compound decisions, and Empirical Bayes rules. *Journal of the American Statistical Association 109*(506), 674–685.

Kolaczyk, E. D. (1996). Non-Parametric Estimation of Gamma-Ray Burst Intensities Using Haar Wavelets. In *The Astrophysical Journal*, pp. 340–349.

Kolaczyk, E. D. (1999, September). Bayesian Multiscale Models for Poisson Processes. *Journal of the American Statistical Association 94*(447), 920–933.

Lee, W. and J. S. Morris (2016, March). Identification of differentially methylated loci using wavelet-based functional mixed models. *Bioinformatics 32*(5), 664–672.

Liu, T.-Y. and Y. S. Song (2016, June). Prediction of ribosome footprint profile shapes from transcript sequences. *Bioinformatics 32*(12), i183–i191.

Love, M. I., W. Huber, and S. Anders (2014, December). Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biology 15*(12), 550+.

Marioni, J. C., C. E. Mason, S. M. Mane, M. Stephens, and Y. Gilad (2008, September). RNA-seq: An assessment of technical reproducibility and comparison with gene expression arrays. *Genome Research 18*(9), 1509–1517.

McCullagh, P. and J. A. Nelder (1989, August). *Generalized Linear Models* (Second Edition ed.). Monographs on Statistics & Applied Probability. London, United Kingdom: Chapman & Hall/CRC.

Menictas, M. and M. P. Wand (2015, March). Variational Inference for Heteroscedastic Semiparametric Regression. *Aust. N. Z. J. Stat. 57*(1), 119–138.

Nason, G. (2013). *wavethresh: Wavelets statistics and transforms*. R package version 4.6.6.

Nason, G. P. (1995). Choice of the Threshold Parameter in Wavelet Function Estimation. In A. Antoniadis and G. Oppenheim (Eds.), *Wavelets and Statistics*, Volume 103 of *Lecture Notes in Statistics*, pp. 261–280. Springer New York.

Pickrell, J. K., J. C. Marioni, A. A. Pai, J. F. Degner, B. E. Engelhardt, E. Nkadori, J.-B. Veyrieras, M. Stephens, Y. Gilad, and J. K. Pritchard (2010, April). Understanding mechanisms underlying human gene expression variation with RNA sequencing. *Nature 464*(7289), 768–772.

Pique-Regi, R., J. F. Degner, A. A. Pai, D. J. Gaffney, Y. Gilad, and J. K. Pritchard (2011, March). Accurate inference of transcription factor binding from DNA sequence and chromatin accessibility data. *Genome Research 21*(3), 447–455.

Pritchard, J. K., M. Stephens, and P. Donnelly (2000, June). Inference of population structure using multilocus genotype data. *Genetics 155*(2), 945–959.

Rabiner, L. R. (1989, February). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE 77*(2), 257–286.

Raj, A., H. Shim, Y. Gilad, J. K. Pritchard, and M. Stephens (2015, September). msCentipede: Modeling Heterogeneity across Genomic Sites and Replicates Improves Accuracy in the Inference of Transcription Factor Binding. *PLoS ONE 10*(9), e0138030+.

Robertson, G., M. Hirst, M. Bainbridge, M. Bilenky, Y. Zhao, T. Zeng, G. Euskirchen, B. Bernier, R. Varhol, A. Delaney, N. Thiessen, O. L. Griffith, A. He, M. Marra, M. Snyder, and S. Jones (2007, August). Genome-wide profiles of STAT1 DNA association using chromatin immunoprecipitation and massively parallel sequencing. *Nature methods 4*(8), 651–657.

Robinson, M. D., D. J. McCarthy, and G. K. Smyth (2010, January). edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics (Oxford, England) 26*(1), 139–140.

Sardy, S., D. B. Percival, A. G. Bruce, H.-Y. Gao, and W. Stuetzle (1999, April). Wavelet shrinkage for unequally spaced data. *Statistics and Computing 9*(1), 65–75.

Shim, H. and M. Stephens (2015). Wavelet-based genetic association analysis of functional phenotypes arising from high-throughput sequencing assays. *Ann. Appl. Stat. 9*(2), 665–686.

Shim, H., Z. Xing, E. Pantaleo, and M. Stephens (in preparation). Bayesian multiscale models for detecting differences in high-thoughput sequencing data between multiple groups and their applications in small sample sizes.

Silverman, B. W. (1985). Some Aspects of the Spline Smoothing Approach to Non-Parametric Regression Curve Fitting. *Journal of the Royal Statistical Society. Series B (Methodological) 47*(1), 1–52.

Silverman, B. W. (1999). Wavelets in Statistics: Beyond the Standard Assumptions. *Philosophical Transactions: Mathematical, Physical and Engineering Sciences 357*(1760), 2459–2473.

Song, L. and G. E. Crawford (2010, February). DNase-seq: a high-resolution technique for mapping active gene regulatory elements across the genome from mammalian cells. *Cold Spring Harbor protocols 2010*(2), pdb.prot5384+.

Stephens, M. (2016). False Discovery Rates: A New Deal. *bioRxiv*.

Tibshirani, R. (1996). Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological) 58*(1), 267–288.

Timmermann, K. E. and R. D. Nowak (1999, April). Multiscale modeling and estimation of Poisson processes with application to photon-limited imaging. *Information Theory, IEEE Transactions on 45*(3), 846–862.

Varadhan, R. (2014). *SQUAREM: Squared extrapolation methods for accelerating fixed-point iterations*. R package version 2014.8-1.

Wang, Z., M. Gerstein, and M. Snyder (2009, January). RNA-Seq: a revolutionary tool for transcriptomics. *Nature Reviews Genetics 10*(1), 57–63.

Wilbanks, E. G. and M. T. Facciotti (2010, July). Evaluation of Algorithm Performance in ChIP-Seq Peak Detection. *PLoS ONE 5*(7), e11471+.

Xing, Z. and M. Stephens (2016, May). Smoothing via Adaptive Shrinkage (smash): denoising Poisson and heteroskedastic Gaussian signals.

Zhang, Y., T. Liu, C. A. Meyer, J. Eeckhoute, D. S. Johnson, B. E. Bernstein, C. Nusbaum, R. M. Myers, M. Brown, W. Li, and X. S. Liu (2008, September). Model-based analysis of ChIP-Seq (MACS). *Genome biology 9*(9), R137+.

APPENDIX A

SUPPLEMENTARY MATERIAL

A.1 Poisson denoising using *smash*

We formally present the likelihood factorization in the *smash* model for Poisson denoising. First, we summarize the data in a recursive manner by defining:

$$Y_{Jk} \equiv Y_k \tag{A.1}$$

for $k = 1, ..., T$, with $T = 2^J$, and

$$Y_{jk} = Y_{j+1,2k} + Y_{j+1,2k+1} \tag{A.2}$$

for resolution $j = 0, ..., J-1$ and location $k = 0, ..., 2^j - 1$. Hence, we are summing more blocks of observations as we move to coarser levels.

This recursive scheme leads to:

$$Y_{jk} = \sum_{l=k2^{J-j}+1}^{(k+1)2^{J-j}} Y_l \tag{A.3}$$

for $j = 0, ..., J$ and $k = 0, ..., 2^j - 1$.

Similarly, define the following:

$$\mu_{Jk} \equiv \mu_k \tag{A.4}$$

for $k = 1, ..., T$, and

$$\mu_{jk} = \mu_{j+1,2k} + \mu_{j+1,2k+1} \tag{A.5}$$

for $j = 0, ..., J-1$ and $k = 0, ..., 2^j - 1$, and

$$\alpha_{jk} = \log(\mu_{j+1,2k}) - \log(\mu_{j+1,2k+1}) \tag{A.6}$$

for $s = 0, ..., J-1$ and $l = 0, ..., 2^j - 1$. The α's defined this way are analogous to the (true) Haar wavelet coefficients for Gaussian signals.

Using this recursive representation, the likelihood for $\boldsymbol{\alpha}$ factorizes into a product of likelihoods, where $\boldsymbol{\alpha}$ is the vector of all the α_{sl}'s. See Kolaczyk (1999)Kolaczyk (1999) for example. Specifically,

$$\begin{aligned} L(\boldsymbol{\alpha}|\mathbf{Y}) &= P(\mathbf{Y}|\boldsymbol{\alpha}) && \text{(A.7)} \\ &= P(Y_{0,0}|\mu_{0,0}) \prod_{j=0}^{J-1} \prod_{k=0}^{2^j-1} P(Y_{j+1,2k}|Y_{j,k}, \alpha_{j,k}) && \text{(A.8)} \\ &= L(\mu_{0,0}|Y_{0,0}) \prod_{j=0}^{J-1} \prod_{k=0}^{2^j-1} L(\alpha_{j,k}|Y_{j+1,2k}, Y_{j,k}). && \text{(A.9)} \end{aligned}$$

Note that $Y_{00}|\mu_{00} \sim \text{Pois}(\mu_{00})$. For any given j, k, Y_{jk} is a sum of two independent Poisson random variables, and is itself a Poisson random variable. Hence

$$Y_{j+1,2k}|Y_{jk}, \alpha_{jk} \sim \text{Bin}(Y_{jk}, \frac{1}{1+e^{-\alpha_{jk}}} \equiv \frac{\mu_{j+1,2k}}{\mu_{jk}})$$

A.1.1 Estimates and standard errors for α_j

Each α_j is a ratio of the form $\log(\mu_{a:b}/\mu_{c:d})$ whose maximum likelihood estimate (MLE) is $\log(Y_{a:b}/Y_{c:d})$. The main challenge here is that the MLE is not well behaved when either the numerator or denominator of $Y_{a:b}/Y_{c:d}$ is 0. To deal with this, when either is 0 we use Tukey's modification Gart and Zweifel (1967). Specifically, letting S denote $Y_{a:b}$, F denote $Y_{c:d}$ and $N = S + F$ (corresponding to thinking of these as successes and failures in a binomial experiment, given $Y_{a:b} + Y_{c:d}$), we use

$$\hat{\alpha} = \begin{cases} \log\{(S+0.5)/(F+0.5)\} - 0.5 & S = 0 \\ \log\{S/F\} & S = 1, 2, ..., N-1 \\ \log\{(S+0.5)/(F+0.5)\} + 0.5 & S = N \end{cases} \tag{A.10}$$

$$se(\hat{\alpha}) = \sqrt{V^*(\hat{\alpha}) - \frac{1}{2}\{V_3(\hat{\alpha})\}^2 \left\{ V_3(\hat{\alpha}) - \frac{4}{N} \right\}} \tag{A.11}$$

where

$$V_3(\hat{\alpha}) = \frac{N+1}{N}\left(\frac{1}{S+1} + \frac{1}{F+1}\right) \quad S = 0, ..., N \tag{A.12}$$

$$V^*(\hat{\alpha}) = V_3(\hat{\alpha})\left\{1 - \frac{2}{N} + \frac{V_3(\hat{\alpha})}{2}\right\} \tag{A.13}$$

The square of the standard error in (A.11) corresponds to V^{**} from p. 182 of Gart and Zweifel (1967), and is chosen because it is less biased for the true variance of $\hat{\alpha}$ (when N is small) as compared to the asymptotic variance of the MLE (see Gart and Zweifel (1967)). The other two variance estimators from Gart and Zweifel (1967), V_1^{++} and V^{++}, were also considered in simulations and gave similar results, but V^{**}

was chosen for its simple form.

A.1.2 Signal reconstruction

The first step to reconstructing the signal is to find the posterior means of $p_{jk} := \frac{\mu_{j+1,2k}}{\mu_{jk}}$ and $q_{jk} := \frac{\mu_{j+1,2k+1}}{\mu_{jk}}$ (for $j = 0, ..., J-1$ and $k = 0, ..., 2^j - 1$). Specifically, for each j and k, we require

$$E(p_{jk}) \equiv E\left(\frac{e^{\alpha_{jk}}}{1+e^{\alpha_{jk}}}\right) \tag{A.14}$$

$$E(q_{jk}) \equiv E\left(\frac{e^{-\alpha_{jk}}}{1+e^{-\alpha_{jk}}}\right). \tag{A.15}$$

Given the posterior means and variances for α_{jk} from *ash*, we can approximate (A.14)-(A.15) using the Delta method. First, define

$$f(x) = \frac{e^x}{1+e^x} \tag{A.16}$$

and consider the Taylor expansion of $f(x)$ about $f(E(x))$:

$$f(x) \approx f(E(x)) + f'(E(x))(x - E(x)) + \frac{f''(E(x))}{2}(x - E(x))^2 \tag{A.17}$$

where

$$f'(x) = \frac{e^x}{(1+e^x)^2} \tag{A.18}$$

$$f''(x) = \frac{e^x(1-e^x)}{(1+e^x)^3} \tag{A.19}$$

Thus

$$E(p_{jk}) \approx f(E(\alpha_{jk})) + \frac{f''(E(\alpha_{jk}))}{2} Var(\alpha_{jk}), \tag{A.20}$$

$$E(q_{jk}) \approx f(-E(\alpha_{jk})) + \frac{f''(-E(\alpha_{jk}))}{2} Var(\alpha_{jk}), \tag{A.21}$$

which can be computed by plugging in $E(\alpha)$ and $Var(\alpha)$ from *ash*.

Finally, we approximate the posterior mean for μ_b by noting that μ_b can be written as a product of the p's and q's for any $b = 1, 2, ..., B$. Specifically, let $\{c_1, ..., c_J\}$ be the binary representation of $b - 1$, and $d_m = \sum_{j=1}^{m} c_j 2^{m-j}$ for $j = 1, ..., J - 1$. Then

$$\mu_b = \mu_{00} p_{00}^{1-c_1} p_{1,d_1}^{1-c_2} ... p_{J-1,d_{J-1}}^{1-c_J} q_{00}^{c_1} q_{1,d_1}^{c_2} ... q_{J-1,d_{J-1}}^{c_J} \tag{A.22}$$

where we usually estimate μ_{00} by $\sum_l Y_l$ (see Kolaczyk (1999)). Using the independence of the p's and q's from different scales, we have:

$$\begin{aligned} E(\mu_b) = \mu_{00} E(p_{00})^{1-c_1} E(p_{1,d_1})^{1-c_2} ... E(p_{J-1,d_{J-1}})^{1-c_J} \\ \cdot E(q_{00})^{c_1} E(q_{1,d_1})^{c_2} ... E(q_{J-1,d_{J-1}})^{c_J}. \end{aligned} \tag{A.23}$$

We can also approximate the posterior variance of μ_b. (This allows creation of an approximate credible interval by making a normal approximation.) From (A.22)

we have:

$$E(\mu_b^2) = \mu_{00}^2 E(p_{00}^2)^{1-c_1} E(p_{1,d_1}^2)^{1-c_2} ... E(p_{J-1,d_{J-1}}^2)^{1-c_J} \cdot E(q_{00}^2)^{c_1} E(q_{1,d_1}^2)^{c_2} ... E(q_{J-1,d_{J-1}}^2)^{c_J}. \quad \text{(A.24)}$$

To compute this we again use the Delta method (with $f(x) = \left(\frac{e^x}{1+e^x}\right)^2$) to obtain:

$$E(p_{jk}^2) \approx \left(f(E(\alpha_{jk})) + \frac{f''(E(\alpha_{jk}))}{2} Var(\alpha_{jk}) \right)^2 + \{f'(E(\alpha_{jk}))\}^2 Var(\alpha_{jk}) \quad \text{(A.25)}$$

$$E(q_{jk}^2) \approx \left(f(-E(\alpha_{jk})) + \frac{f''(-E(\alpha_{jk}))}{2} Var(\alpha_{jk}) \right)^2 + \{f'(E(-\alpha_{jk}))\}^2 Var(\alpha_{jk}). \quad \text{(A.26)}$$

Finally we combine (A.23) and (A.24) to find $Var(\mu_b)$.

A.1.3 Simulation results

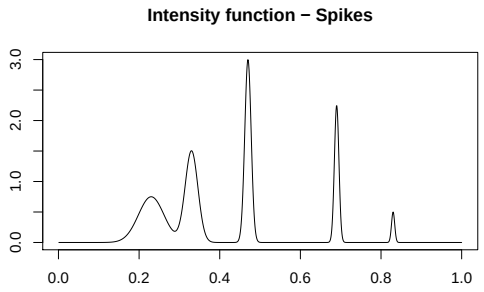
The results from the simulation study for Poisson noise are shown in the following tables. For reference, we first plot the intensity functions used in the simulation studies in Figure A.1. Each of the six tables (A.1-A.6) presents results for the corresponding mean functions, and includes results for all three (min,max) intensity levels ((0.01,3), (1/8,8), (1/128,128)). As mentioned in the main text, we also explored several options for the Gaussian de-noising stage of the HF algorithm, and include the results in the following tables. Specifically, we considered four different options for HF (all with 50 external cyclespins):

1. Hybrid of (1) Greedy tree de-noising (Baraniuk (1999)) and (2) wavelet thresholding using "leave-half-out" crossvalidation (e.g. Nason (1995)). $j_0 = 3$ (default), and the noise level was estimated from the data. This corresponds to **H:CV+BT CS** in Fryzlewicz and Nason (2004). Note that the algorithm does not always converge for this option, resulting in NA values in the tables below.

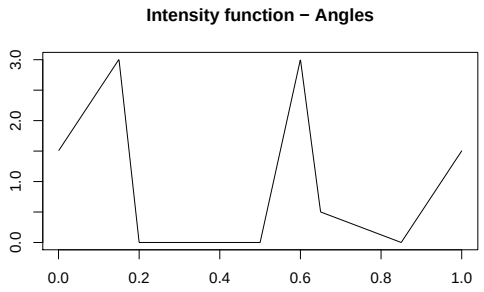
2. Wavelet thresholding using the universal threshold (Donoho and Johnstone (1994)). $j_0 = 3$ (default), and the noise level was estimated from the data. This corresponds to **F⋈U CS** in Fryzlewicz and Nason (2004).

3. Wavelet thresholding using the universal threshold for the non-decimated wavelet transform. Results averaged over $j_0 = 4, 5, 6, 7$, and the noise level was estimated from the data.

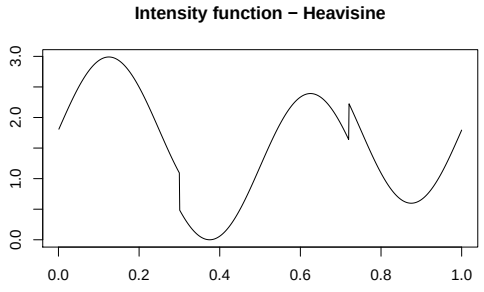
4. Wavelet thresholding using the universal threshold for the non-decimated wavelet transform. Results averaged over $j_0 = 4, 5, 6, 7$, and the noise level was set to be 1 (which is the asymptotic variance under the Fisz transform). This is the option for which we included the results in the main text, as it had the best overall performance.



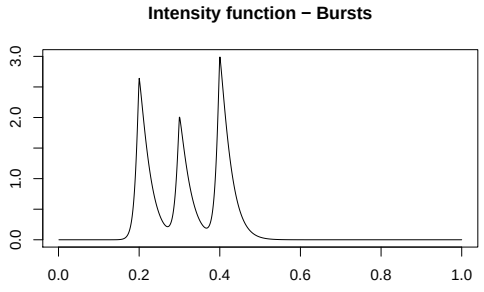
(a)



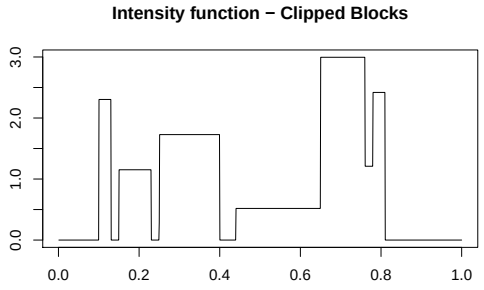
(b)



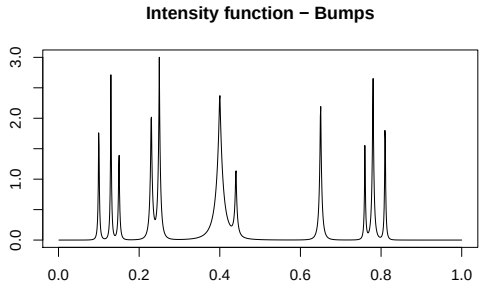
(c)



(d)



(e)



(f)

Figure A.1: The six intensity functions used in the Poisson simulations, which are rescaled to achieve the desired (min,max) intensities in the simulations.

	(0.01,3)	(1/8,8)	(1/128,128)
SMASH	690.01	329.26	48.87
BMSM	1007.34	397.79	41.88
BMMIM	930.08	436.54	368.36
Haar-Fisz (1)	8194.60	NA	184.81
Haar-Fisz (2)	8074.05	408.32	158.68
Haar-Fisz (3)	5904.98	317.12	73.00
Haar-Fisz (4)	722.19	287.44	18.06
Anscombe	1329.87	388.59	18.17
Haar thresholds	803.89	326.11	120.02
L1_penalty	3085.65	2209.67	1516.78

Table A.1: Comparison of methods for denoising Poisson data for the "Spikes" test function for 3 different (min,max) intensities ((0.01,3), (1/8,8), (1/128,128)). Performance is measured using MISE over 100 independent datasets, with smaller values indicating better performance. Values colored in red indicates the smallest MISE amongst all methods (excluding Haar-Fisz (1-3)) for a given (min, max) intensity.

	(0.01,3)	(1/8,8)	(1/128,128)
SMASH	145.26	68.47	10.25
BMSM	147.40	73.87	10.49
BMMIM	245.89	100.17	71.26
Haar-Fisz (1)	NA	91.86	25.23
Haar-Fisz (2)	1020.76	78.84	12.22
Haar-Fisz (3)	614.95	125.78	11.29
Haar-Fisz (4)	314.41	122.79	9.08
Anscombe	419.04	146.64	9.50
Haar thresholds	264.38	162.16	89.37
L1_penalty	655.82	284.75	2979.23

Table A.2: Comparison of methods for denoising Poisson data for the "Angles" test function for 3 different (min,max) intensities ((0.01,3), (1/8,8), (1/128,128)). Performance is measured using MISE over 100 independent datasets, with smaller values indicating better performance. Values colored in red indicates the smallest MISE amongst all methods (excluding Haar-Fisz (1-3)) for a given (min, max) intensity.

	(0.01,3)	(1/8,8)	(1/128,128)
SMASH	81.41	43.21	7.21
BMSM	85.29	44.22	7.35
BMMIM	205.09	81.49	11.22
Haar-Fisz (1)	135.42	62.01	8.10
Haar-Fisz (2)	105.72	61.66	6.29
Haar-Fisz (3)	273.97	104.99	9.98
Haar-Fisz (4)	274.26	105.47	9.23
Anscombe	372.06	128.43	9.10
Haar thresholds	226.27	143.01	85.74
L1_penalty	385.47	71.17	8.68

Table A.3: Comparison of methods for denoising Poisson data for the "Heavisine" test function for 3 different (min,max) intensities ((0.01,3), (1/8,8), (1/128,128)). Performance is measured using MISE over 100 independent datasets, with smaller values indicating better performance. Values colored in red indicates the smallest MISE amongst all methods (excluding Haar-Fisz (1-3)) for a given (min, max) intensity.

	(0.01,3)	(1/8,8)	(1/128,128)
SMASH	487.34	234.35	33.11
BMSM	706.04	301.86	34.42
BMMIM	654.11	290.91	531.79
Haar-Fisz (1)	6870.35	NA	157.15
Haar-Fisz (2)	6807.29	384.56	149.23
Haar-Fisz (3)	5619.13	318.57	94.20
Haar-Fisz (4)	618.39	299.39	25.20
Anscombe	869.16	343.15	26.03
Haar thresholds	540.58	271.16	107.44
L1_penalty	1836.73	1280.16	719.59

Table A.4: Comparison of methods for denoising Poisson data for the "Bursts" test function for 3 different (min,max) intensities ((0.01,3), (1/8,8), (1/128,128)). Performance is measured using MISE over 100 independent datasets, with smaller values indicating better performance. Values colored in red indicates the smallest MISE amongst all methods (excluding Haar-Fisz (1-3)) for a given (min, max) intensity.

	(0.01,3)	(1/8,8)	(1/128,128)
SMASH	307.80	137.28	6.82
BMSM	355.15	143.09	6.91
BMMIM	472.24	205.20	150.03
Haar-Fisz (1)	NA	141.71	31.23
Haar-Fisz (2)	879.20	316.44	26.71
Haar-Fisz (3)	741.49	389.79	48.41
Haar-Fisz (4)	632.21	338.55	29.72
Anscombe	804.70	384.46	28.94
Haar thresholds	467.64	228.14	82.99
L1_penalty	1021.58	627.67	1961.42

Table A.5: Comparison of methods for denoising Poisson data for the "Clipped Blocks" test function for 3 different (min,max) intensities ((0.01,3), (1/8,8), (1/128,128)). Performance is measured using MISE over 100 independent datasets, with smaller values indicating better performance. Values colored in red indicates the smallest MISE amongst all methods (excluding Haar-Fisz (1-3)) for a given (min, max) intensity.

	(0.01,3)	(1/8,8)	(1/128,128)
SMASH	2597.46	1194.62	141.21
BMSM	4036.77	1889.94	171.07
BMMIM	3440.75	2226.20	291.22
Haar-Fisz (1)	9515.68	NA	187.6
Haar-Fisz (2)	9249.05	1288.30	130.19
Haar-Fisz (3)	6286.05	1756.08	248.28
Haar-Fisz (4)	3113.37	1658.74	184.66
Anscombe	4038.42	2241.74	195.75
Haar thresholds	3517.89	1652.55	173.72
L1_penalty	4705.77	3676.42	1774.51

Table A.6: Comparison of methods for denoising Poisson data for the "Bumps" test function for 3 different (min,max) intensities ((0.01,3), (1/8,8), (1/128,128)). Performance is measured using MISE over 100 independent datasets, with smaller values indicating better performance. Values colored in red indicates the smallest MISE amongst all methods (excluding Haar-Fisz (1-3)) for a given (min, max) intensity.

A.2 Translation invariance for multiscale methods *smash* and *Multiseq*

It is common in multi-scale analysis to perform analyses over all B (or T, using notation for the Gaussian version of *smash*) circulant shifts of the data, because this is known to consistently improve accuracy. (The b-th circulant shift of the signal $\boldsymbol{Y}$ is created from $\boldsymbol{Y}$ by moving the first $B - b$ elements of $\boldsymbol{Y}$ b positions to the right and then putting the last b elements of $\boldsymbol{Y}$ in the first b locations.)

To implement this in practice, we begin by computing the α coefficients, and their corresponding standard errors, for all B circulant shifts of the data. This is done efficiently (in $O(\log_2 B)$ operations), using ideas from Coifman and Donoho (1995). Details are as in Kolaczyk (1999); indeed our software implementation benefitted from MATLAB code provided by Kolaczyk (1999) for the TI table construction, which we ported to C++ and integrated with R using Rcpp (Eddelbuettel et al. (2011)).

This yields a table of α coefficients, with B coefficients at each of $\log_2(B)$ resolution levels, and a corresponding table of standard errors. As in the Gaussian case we then apply *ash* separately to the B coefficients at each resolution level, to obtain a posterior mean and posterior variance for each α. We then use the methods above to compute quantities of interest averaged over all B shifts of the data. For example, our final estimate of the mean signal is given by

$$\frac{1}{B}\sum_{b=1}^{B}\hat{\mu}_k^{(b)} \tag{A.27}$$

where $\hat{\mu}_k^{(b)}$ denotes the posterior mean of μ_k computed from the b-th circulant shift of the data. Again, using ideas from Coifman and Donoho (1995) this averaging can be done in $O(\log_2 B)$ operations.

A.3 Estimating effect sizes using *Multiseq*

A.3.1 Reparametrization of μ and β

One major problem with shrinking μ and β separately is that the likelihood for the two cannot be factorized, so that the posterior distributions of μ and β will not be independent. One solution might be to center the covariate g, which would allow the likelihood to be factorized *if* this was a linear regression model. In our case, this would reduce the covariance between μ and β, but most likely not to zero. To get around the problem, we make use of the idea presented in Wakefield (2009). Specifically, let $\hat{\mu}$ ($se(\hat{\mu})$) and $\hat{\beta}$ ($se(\hat{\beta})$) be the MLEs (and their standard errors) of μ and β respectively, as in a standard logistic regression. We then have (asymptotically)

$$\begin{pmatrix} \hat{\mu} \\ \hat{\beta} \end{pmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu \\ \beta \end{bmatrix}, \begin{bmatrix} I_{00} & I_{01} \\ I_{01} & I_{11} \end{bmatrix}^{-1} \right) \tag{A.28}$$

where $\mathbf{I}$ is the Fisher's information matrix for the MLEs. Hence

$$se(\hat{\mu}) = \sqrt{\frac{I_{11}}{I_{00}I_{11} - I_{01}^2}} \tag{A.29}$$

$$se(\hat{\beta}) = \sqrt{\frac{I_{00}}{I_{00}I_{11} - I_{01}^2}} \tag{A.30}$$

$$cov(\hat{\mu}, \hat{\beta}) = -\frac{I_{01}}{I_{00}I_{11} - I_{01}^2} \tag{A.31}$$

$$\tag{A.32}$$

Now, by defining

$$\mu^* = \mu + \frac{I_{01}}{I_{00}}\beta = \mu - \frac{cov(\hat{\mu}, \hat{\beta})}{se^2(\hat{\beta})}\beta \tag{A.33}$$

$$\hat{\mu}^* = \hat{\mu} + \frac{I_{01}}{I_{00}}\hat{\beta} = \hat{\mu} - \frac{cov(\hat{\mu}, \hat{\beta})}{se^2(\hat{\beta})}\hat{\beta} \tag{A.34}$$

we have that

$$cov(\hat{\mu}^*, \hat{\beta}) = 0 \tag{A.35}$$

so that we can write $P(\hat{\mu}^*, \hat{\beta}|\mu^*, \beta)$ as $P(\hat{\mu}^*|\mu^*) \times P(\hat{\beta}|\beta)$ (at least asymptotically). We can then set independent shrinkage priors for μ^* and β in order to obtain independent posterior estimates for the two parameters. The posterior for the multi-scale coefficients μ can be easily obtained by

$$\mu = \mu^* + \gamma\beta \tag{A.36}$$

where $\gamma = -\frac{I_{01}}{I_{00}}$.

A.3.2 Computing $\hat{\mu}$ and $\hat{\beta}$ and their standard errors from $\hat{\alpha}$ and $se(\hat{\alpha})$

We derive analytic forms for the estimates and standard errors of μ and β when g is a two-group categorical variable. Specifically, assume that we have groups labelled 0 and 1. WLOG, set group 0 to be the baseline, so that $\mu \equiv \alpha^0$ and $\beta \equiv \alpha^1 - \alpha^0$.

Using the procedure outlined in Appendix A.1.1, we obtain estimates and standard errors for α^0 and α^1. As described in Section 3.1, we approximate the likelihood for α^0 and α^1 by a Gaussian likelihood, so that

$$L(\alpha^i; \hat{\alpha}^i) = \phi(\alpha^i; \hat{\alpha}^i, \mathbb{V}(\hat{\alpha}^i)) \qquad (i = 0, 1) \tag{A.37}$$

Hence,

$$L(\mu; \hat{\mu}) = \phi(\alpha^0; \hat{\alpha}^0, \mathbb{V}(\hat{\alpha}^0)) \tag{A.38}$$

$$L(\beta; \hat{\beta}) = \phi(\beta; \hat{\alpha}^1 - \hat{\alpha}^0, \mathbb{V}(\hat{\alpha}^1) + \mathbb{V}(\hat{\alpha}^0)) \tag{A.39}$$

In practice, since we wish to obtain the likelihood for μ^* instead of μ, we have that

$$L(\mu^*; \hat{\mu}^*) = \phi(\mu^*; \hat{\mu} + \frac{I_{01}}{I_{00}}\hat{\beta}, \mathbb{V}(\hat{\mu}) + (\frac{I_{01}}{I_{00}})^2\mathbb{V}(\hat{\beta})) \tag{A.40}$$

A.3.3 Estimating the baseline μ_b^o

Estimating the baseline in *Multiseq* is extremely similar to estimating the mean in *smash*. As a first step, we need to find the posterior means and variances of

$$\log(p_{jk}) = \log\left(\frac{e^{\mu_{jk}}}{1 + e^{\mu_{jk}}}\right) \tag{A.41}$$

$$\log(q_{jk}) = \log\left(\frac{e^{-\mu_{jk}}}{1 + e^{-\mu_{jk}}}\right) \tag{A.42}$$

Note that *ash* returns the posterior means and variances of μ^*_{jk} and β_{jk}, which are independent due to the reparametrization step performed in A.3.1. Hence, it is trivial to see that

$$E(\mu_{jk}) = E(\mu^*_{jk}) + \gamma E(\beta_{jk}) \tag{A.43}$$

$$Var(\mu_{jk}) = Var(\mu^*_{jk}) + \gamma^2 Var(\beta_{jk}) \tag{A.44}$$

We then follow the steps outlined in A.1.2 to compute $E(\log(p_{jk}))$, $Var(\log(p_{jk}))$, $E(\log(q_{jk}))$ and $Var(\log(q_{jk}))$, with the only difference being that we need to redefine $f(x)$ as $\log(\frac{e^x}{1+e^x})$ to compute the posterior means, and as $(\log(\frac{e^x}{1+e^x}))^2$ to compute the posterior variances. Finally, computing $E(\mu^o_b)$ and $Var(\mu^o_b)$ from $E(\log(p_{jk}))$, $Var(\log(p_{jk}))$, $E(\log(q_{jk}))$ and $Var(\log(q_{jk}))$ is identical to how it is done in A.1.2.

A.3.4 Estimating the effect β^o_b when g is a two-group categorical variable

We consider estimating the effect size when g is a two-group categorical variable. To start off, define

$$p^{(j)}_{jk} = \frac{e^{\alpha^{(j)}_{jk}}}{1 + e^{\alpha^{(j)}_{jk}}} \tag{A.45}$$

$$q^{(j)}_{jk} = \frac{e^{-\alpha^{(j)}_{jk}}}{1 + e^{-\alpha^{(j)}_{jk}}} \tag{A.46}$$

where $\alpha_{jk}^{(0)} = \mu_{jk}$ and $\alpha_{jk}^{(1)} = \mu_{jk} + \beta_{jk}$. To estimate β_b^o, we need to compute the posterior means and variances of

$$\log\left(\frac{p_{jk}^{(1)}}{p_{jk}^{(0)}}\right) \equiv \log\left(\frac{1+e^{-(\mu_{jk})}}{1+e^{-(\mu_{jk}+\beta_{jk})}}\right) \equiv \log\left(\frac{1+e^{-(\mu_{jk}^*+\gamma_{jk}\beta_{jk})}}{1+e^{-(\mu_{jk}^*+(1+\gamma_{jk})\beta_{jk})}}\right) \quad \text{(A.47)}$$

$$\log\left(\frac{q_{jk}^{(1)}}{q_{jk}^{(0)}}\right) \equiv \log\left(\frac{1+e^{\mu_{jk}}}{1+e^{\mu_{jk}+\beta_{jk}}}\right) \equiv \log\left(\frac{1+e^{\mu_{jk}^*+\gamma_{jk}\beta_{jk}}}{1+e^{\mu_{jk}+(1+\gamma_{jk})\beta_{jk}}}\right) \quad \text{(A.48)}$$

Here we make use of Taylor approximations again. Specifically, we first consider a random variable X and an arbitrary real-valued function $g()$. To compute $E(g(X))$, note that

$$g(X) \approx g(E(X)) + g'(E(X))(X - E(X)) + \frac{g''(E(X))}{2}(X - E(X))^2 \quad \text{(A.49)}$$

so that

$$E(g(X)) \approx g(E(X)) + \frac{g''(E(X))}{2}Var(X) \quad \text{(A.50)}$$

Now define

$$f(x) = \log(1 + e^x) \quad \text{(A.51)}$$

and consider the second order Taylor approximations to $f(x)$ around $-\mu^*$ and μ^*:

$$f(x) \approx f(-\mu^*) + f'(-\mu^*)(x+\mu^*) + \frac{f''(-\mu^*)}{2}(x+\mu^*)^2 \tag{A.52}$$

$$f(x) \approx f(\mu^*) + f'(\mu^*)(x-\mu^*) + \frac{f''(\mu^*)}{2}(x-\mu^*)^2 \tag{A.53}$$

where

$$f'(x) = \frac{e^x}{1+e^x} \tag{A.54}$$

$$f''(x) = \frac{e^x}{(1+e^x)^2} \tag{A.55}$$

Replacing the dummy variable x above by the corresponding terms for the different cases ($-(\mu^* + \gamma\beta), -(\mu^* + (1+\gamma)\beta), (\mu^* + \gamma\beta)$ or $(\mu + (1+\gamma)\beta)$) gives us

$$\log\left(\frac{p_{jk}^{(1)}}{p_{jk}^{(0)}}\right) \approx f'(-\mu_{jk}^*)\beta_{jk} - \frac{f''(-\mu_{jk}^*)}{2}\beta_{jk}^2((1+\gamma_{jk})^2 - \gamma_{jk}^2) \tag{A.56}$$

$$\log\left(\frac{q_{jk}^{(1)}}{q_{jk}^{(0)}}\right) \approx -f'(\mu_{jk}^*)\beta_{jk} - \frac{f''(\mu_{jk}^*)}{2}\beta_{jk}^2((1+\gamma_{jk})^2 - \gamma_{jk}^2) \tag{A.57}$$

Using the independence of μ^* and β, we have

$$\begin{aligned}
E\left(\log\left(\frac{p_{jk}^{(1)}}{p_{jk}^{(0)}}\right)\right) \approx\ & E(f'(-\mu_{jk}^*))E(\beta_{jk}) \\
& -\frac{((1+\gamma_{jk})^2-\gamma_{jk}^2)}{2}E(f''(-\mu_{jk}^*))E(\beta_{jk}^2) \qquad \text{(A.58)} \\
E\left(\log\left(\frac{q_{jk}^{(1)}}{q_{jk}^{(0)}}\right)\right) \approx\ & -E(f'(\mu_{jk}^*))E(\beta_{jk}) \\
& -\frac{((1+\gamma_{jk})^2-\gamma_{jk}^2)}{2}E(f''(\mu_{jk}^*))E(\beta_{jk}^2) \qquad \text{(A.59)} \\
E\left(\log^2\left(\frac{p_{jk}^{(1)}}{p_{jk}^{(0)}}\right)\right) \approx\ & E(\{f'(-\mu_{jk}^*)\}^2)E(\beta_{jk}^2) \\
& +\frac{((1+\gamma_{jk})^2-\gamma_{jk}^2)^2}{4}E(\{f''(-\mu_{jk}^*)\}^2)E(\beta_{jk}^4) \\
& -((1+\gamma_{jk})^2-\gamma_{jk}^2)E(f'(-\mu_{jk}^*)f''(-\mu_{jk}^*))E(\beta_{jk}^3) \qquad \text{(A.60)} \\
E\left(\log^2\left(\frac{q_{jk}^{(1)}}{q_{jk}^{(0)}}\right)\right) \approx\ & E(\{f'(\mu_{jk}^*)\}^2)E(\beta_{jk}^2) \\
& +\frac{((1+\gamma_{jk})^2-\gamma_{jk}^2)^2}{4}E(\{f''(\mu_{jk}^*)\}^2)E(\beta_{jk}^4) \\
& +((1+\gamma_{jk})^2-\gamma_{jk}^2)E(f'(\mu_{jk}^*)f''(\mu_{jk}^*))E(\beta_{jk}^3) \qquad \text{(A.61)}
\end{aligned}$$

While *ash* returns the estimates of $E(\mu_{jk}^*)$ and $E(\beta_{jk})$, we can approximate the quantities $E(f'(-\mu_{jk}^*))$, $E(f'(\mu_{jk}^*))$, $E(f''(-\mu_{jk}^*))$, $E(f''(\mu_{jk}^*))$, $E(\{f'(-\mu_{jk}^*)\}^2)$, $E(\{f'(\mu_{jk}^*)\}^2)$, $E(\{f''(-\mu_{jk}^*)\}^2)$, $E(\{f''(\mu_{jk}^*)\}^2)$, $E(f'(-\mu_{jk}^*)f''(-\mu_{jk}^*))$ and $E(f'(\mu_{jk}^*)f''(\mu_{jk}^*))$ by making use of (A.50). By noting that the posterior distribution for β is a Gaussian mixture, we can compute $E(\beta_{jk}^k), k = 2, 3, 4$ exactly without having to use (A.50).

Specifically, suppose that

$$\beta \sim \sum_i \pi_i N(\theta_i, \sigma_i^2) \tag{A.62}$$

Then

$$E(\beta^h) = \sum_i \pi_i E(N_i^h) \tag{A.63}$$

for any nonnegative integer h, where $N_i \sim N(\theta_i, \sigma_i^2)$. Given that

$$E(N_i^3) = \theta_i^3 + 3\theta_i\sigma_i^2 \tag{A.64}$$

$$E(N_i^4) = \theta_i^4 + 6\theta_i^2\sigma_i^2 + 3\sigma_i^4 \tag{A.65}$$

we can then easily compute $E(\beta^3)$ and $E(\beta^4)$. Combining all these allows us to (approximately) compute the quantities in (A.58)-(A.61), and hence the posterior means and variances of the quantities in (A.47) and (A.48).

Finally, by observing that $\beta_b^o \equiv \frac{\lambda_b^1}{\lambda_b^0}$ and making use of the representation in (A.22) for each of $\lambda_b^i, i = 1, 2$, we have

$$\begin{aligned}\log(\beta_b^o) \;=\; & \log\left(\lambda_{00}^1\lambda_{00}^0\right) + \log\left(\frac{p_{00}^{(1)1-c_1}}{p_{00}^{(0)1-c_1}}\right) + \log\left(\frac{p_{1,d_1}^{(1)1-c_2}}{p_{1,d_1}^{(0)1-c_2}}\right) + ... + \log\left(\frac{p_{J-1,d_{J-1}}^{(1)1-c_J}}{p_{J-1,d_{J-1}}^{(0)1-c_J}}\right) \\ & + \log\left(\frac{q_{00}^{(1)c_1}}{q_{00}^{(0)c_1}}\right) + \log\left(\frac{q_{1,d_1}^{(1)c_2}}{q_{1,d_1}^{(0)c_2}}\right) + ... + \log\left(\frac{q_{J-1,d_{J-1}}^{(1)c_J}}{q_{J-1,d_{J-1}}^{(0)c_J}}\right)\end{aligned} \tag{A.66}$$

so that the posterior mean and variance of $\log(\beta_b^o)$ can be easily computed using the posterior means and variances of (A.47) and (A.48) and using the fact that the terms in (A.66) are independent.

A.3.5 Estimating the effect β_b^o when g is a quantitative variable

We consider estimating the effect size when g is a quantitative variable. The procedure for this scenario is almost identical to that in A.3.4, and is in fact algebraically simpler. We first define $\log\left(\frac{p_{jk}^{(1)}}{p_{jk}^{(0)}}\right)$ and $\log\left(\frac{q_{jk}^{(1)}}{q_{jk}^{(0)}}\right)$ as in (A.47) and (A.48). Next we define $f(x)$ as in (A.51). However, we only consider the first order Taylor expansion around $-\mu^*$ and μ^*, the reason being that higher order terms are difficult to compute under the linearity assumption in (3.2). Hence, the approximations are given by

$$f(x) \approx f(-\mu^*) + f'(-\mu^*)(x + \mu^*) \tag{A.67}$$

$$f(x) \approx f(\mu^*) + f'(\mu^*)(x - \mu^*) \tag{A.68}$$

where

$$f'(x) = \frac{e^x}{1 + e^x} \tag{A.69}$$

which gives us

$$\log\left(\frac{p_{jk}^{(1)}}{p_{jk}^{(0)}}\right) \approx f'(-\mu_{jk}^*)\beta_{jk} \tag{A.70}$$

$$\log\left(\frac{q_{jk}^{(1)}}{q_{jk}^{(0)}}\right) \approx -f'(\mu_{jk}^*)\beta_{jk} \tag{A.71}$$

and thus

$$E\left(\log\left(\frac{\tilde{p}_{jk}}{p_{jk}}\right)\right) \approx E(f'(-\mu_{jk}^*))E(\beta_{jk}) \tag{A.72}$$

$$E\left(\log\left(\frac{\tilde{q}_{jk}}{q_{jk}}\right)\right) \approx -E(f'(\mu_{jk}^*))E(\beta_{jk}) \tag{A.73}$$

$$E\left(\log^2\left(\frac{\tilde{p}_{jk}}{p_{jk}}\right)\right) \approx E(\{f'(-\mu_{jk}^*)\}^2)E(\beta_{jk}^2) \tag{A.74}$$

$$E\left(\log^2\left(\frac{\tilde{q}_{jk}}{q_{jk}}\right)\right) \approx E(\{f'(\mu_{jk}^*)\}^2)E(\beta_{jk}^2) \tag{A.75}$$

As with the case when g is a two-group categorical variable, all the necessary quantities in (A.72)-(A.75) can be approximated using (A.50), allowing us to compute the posterior means and variances of (A.47) and (A.48). Computing $\log(\beta_b^o)$ is then exactly the same as in A.3.4.

A.3.6 Weighted Least Squares

As discussed in Section 3, while estimates and standard errors for μ_{jk} and β_{jk} can be computed analytically where g is a two-group categorical variable, we have to run

the full iterative procedure when fitting a logistic model for the case when g is a quantitative covariate. As such, computation can be slow, considering that we have to fit the model for each j and each k. Here we consider an alternative approach to compute the estimates and standard errors for μ_{jk} and β_{jk}, by using an approximate model instead. Specifically, instead of fitting (3.2), we approximate the likelihoods of α^i_{jk} using Gaussian likelihoods (for each i) as described in Appendix A.1.1. By using Gaussian likelihoods instead of Binomial likelihoods, the model in (3.2) becomes

$$\hat{\alpha}^i_{jk} = \mu_{jk} + \beta_{jk} g^i + u^i_{jk} + \epsilon^i_{jk} \tag{A.76}$$

where $\hat{\alpha}^i_{jk}$ are the observations for $i = 1, ..., m$, μ_{jk} is the intercept, β_{jk} is the coefficient for covariate g, u^i_{jk} are iid $N(0, (\sigma^{(u)}_{jk})^2)$ across i, and ϵ^i_{jk} are independent $N(0, (\sigma^i_{jk})^2)$ across i, independent of u^i_{jk}. Since we have standard errors for $\hat{\alpha}^i_{jk}$, we can simply substitute σ^i_{jk} by $\hat{\sigma}^i_{jk} \equiv se(\hat{\alpha}^i_{jk})$. This model can then be reformulated as

$$\hat{\alpha}^i_{jk} = \mu_{jk} + \beta_{jk} g^i + \tilde{\epsilon}^i_{jk} \tag{A.77}$$

where $\hat{\alpha}^i_{jk}$, μ_{jk} and β_{jk} are as above, and $\tilde{\epsilon}^i_{jk}$ are independent $N(0, (\sigma^{(u)}_{jk})^2 + (\hat{\sigma}^i_{jk})^2)$ across i. As such, we can fit (A.77) using a weighted least squares (WLS) approach with the form of heteroskedasticity known. The only unknown parameter in the weights is $\sigma^{(u)}_{jk}$, which will have to be estimated in order to choose the right weights for (A.77).

The standard procedure for fitting a logistic model is to use a iterative weighted

least squares approach. Although we have motivated (A.77) from anther point of view, fitting the model (A.77) using a standard WLS approach is actually (almost) equivalent to truncating the iterative procedure after the first step in the case of a logistic model. Thus, this presents another link between the two models (3.2) and (A.77), and provides good justification for approximating the former model using the latter.

Although the alternative model (A.77) was motivated by the goal of computational efficiency, we note that it has another conceptual advantage over the logistic model (3.2), in the case where sequencing depths might be different for each sample. In our quasibinomial implementation, the success (and failure) counts for samples in the same group will be pooled together (since each Bernoulli count is assumed to be i.i.d.). This could then lead to the information from that sample overwhelming the information from all the rest of the samples, which might bias the parameter estimates if that sample happens to be different from the rest. The WLS approach (A.77) weighs each sample according to its associated uncertainty instead, which is a better alternative.

To actually fit (A.77) and obtain estimates and standard errors for μ and β (dropping the scale and location subscripts for convenience), we seek to minimize the weighted sum of square differences:

$$\sum_i w_i^2(\hat{\alpha}_i^2 - \mu - \beta g_i)^2 \tag{A.78}$$

For notational convenience, the index i for sample has been moved to be the subscript

for the following derivations. Here, ideally it should be the case that $w_i = (\sigma_u^2 + \hat{\sigma}_i^2)^{-\frac{1}{2}}$. By taking partial derivatives w.r.t. μ and β and setting them to 0, we have:

$$\sum_i w_i^2(\hat{\alpha}_i^2 - \mu - \beta g_i) = 0 \tag{A.79}$$

$$\sum_i w_i^2 g_i(\hat{\alpha}_i^2 - \mu - \beta g_i) = 0 \tag{A.80}$$

After some algebra, we obtain the following WLS estimates:

$$\hat{\beta}_{WLS} = \frac{\sum_i w_i^2 \hat{\alpha}_i(g_i - \bar{g}_W)}{\sum_i g_i w_i^2(g_i - \bar{g}_W)} \tag{A.81}$$

$$\hat{\mu}_{WLS} = \frac{\sum_i w_i^2(\hat{\alpha}_i - \hat{\beta}_{WLS} g_i)}{\sum_i w_i^2} \tag{A.82}$$

From these we can also derive the standard errors for the estimates:

$$se(\hat{\beta}_{WLS}) = \sqrt{\frac{Var(w_i\hat{\alpha}_i)}{\sum_i g_i w_i^2(g_i - \bar{g}_W)}} \tag{A.83}$$

$$se(\hat{\mu}_{WLS}) = \sqrt{\sum_i \left(\frac{w_i}{\sum_j w_j^2} - \frac{\bar{g}_W w_i(g_i - \bar{g}_W)}{\sum_j g_j w_j^2(g_j - \bar{g}_W)}\right)^2 Var(w_i\hat{\alpha}_i)} \tag{A.84}$$

and also

$$Cov(\hat{\mu}_{WLS}, \hat{\beta}_{WLS}) = \frac{\sum_i w_i^2(g_i - \bar{g}_W)Var(w_i\hat{\alpha}_i)}{\sum_i w_i^2 \sum_i w_i^2(g_i - \bar{g}_W)^2} - \bar{g}_W se^2(\hat{\beta}_{WLS}) \tag{A.85}$$

to be used in the reparametrization as in the previous section. Further define

$$RSE = \sqrt{\frac{1}{m-2}\sum_i w_i^2(\hat{\alpha}_i - \hat{\mu}_{WLS} - \hat{\beta}_{WLS}g_i)^2} \tag{A.86}$$

as the residual standard error. Note that if w_i was exactly $(\sigma_u^2 + \hat{\sigma}_i^2)^{-\frac{1}{2}}$, the value of RSE would be 1. However, σ_u^2 is unknown, and hence must be estimated in some way.

To estimate σ_u, we start off by fitting the model with no random effect (i.e. $\sigma_u^2 = 0$). Note that we can simply substitute $Var(w_i\hat{\alpha}_i)$ in (A.83)-(A.85) with its estimate, RSE^2, when fitting the model. In this case, the value of RSE would be an indication of the presence of the random effect. *If* the additional sample specific random effect were absent, then the value of RSE would be close to 1. Assuming that our assumptions hold however, RSE would be different from 1, and would provide some information on the amount of "overdispersion" present. Hence, consider the "dispersion factor" defined by $\hat{\lambda} = \min\{1, RSE^2\}$. This is because $\sigma_u^2 + \hat{\sigma}_i^2 \geq \hat{\sigma}_i^2$, so we do not allow for "underdispersion".

To avoid heavy computation, we only seek to find a crude estimate for σ_u^2. For the model with no random effect, the "dispersion factor" $\hat{\lambda}$ is a multiplier on $\hat{\sigma}^2$, whereas the model in (A.77) captures over-dispersion using an additive effect. Nevertheless, the total amount of variation in the data is the same regardless of the model we fit, so we set

$$\sum_i(\hat{\sigma}_i^2 + \sigma_u^2) = \sum_i \hat{\lambda}\hat{\sigma}_i^2 \tag{A.87}$$

to solve for σ_u^2. This results in

$$\hat{\sigma}_u^2 = \frac{\hat{\lambda} - 1}{n} \sum_i \hat{\sigma}_i^2 \tag{A.88}$$

With the estimate in hand, we can then fit (A.77) with weights $w_i = (\hat{\sigma}_i^2 + \hat{\sigma}_u^2)^{-\frac{1}{2}}$. Provided that we have a moderate number of observations as well as a moderate number of binomial counts in each observation, the value of RSE we get from this fit should be close to 1, indicating that we have a decent fit. Given the estimates of μ and β, we can then estimate the baseline signal and the effect as outlined in A.3.3-A.3.5.

A.4 Estimating effect sizes using *HMM-seq*

A.4.1 EM updates to estimate π and q

Since multiscale methods will not be mentioned for Appendices A.4.1-A.4.2, we drop the superscript o on the effect size β to avoid clutter.

The marginal likelihood for the HMM model proposed in Section 3.4 is given by

$$\begin{aligned} P(\boldsymbol{Y}|q,\boldsymbol{\pi}) &= \sum_{\text{all } \beta,Z} P(\boldsymbol{Y}|\boldsymbol{\beta})P(\boldsymbol{\beta}|\boldsymbol{Z},\boldsymbol{\pi})P(\boldsymbol{Z}|q) \\ &= \sum_{\beta_1,\ldots,\beta_B,Z_2,\ldots,Z_B} \pi_{\beta_1} f(Y_1|\beta_1) \prod_{b=2}^{B} (q\pi_{\beta_b} Z_b + \\ &\qquad (1-q)(1-Z_b) I_{\{\beta_b=\beta_{b-1}\}}) f(Y_b|\beta_b) \end{aligned} \tag{A.89}$$

where Y are the observed sequencing reads, β is the effect size as described in (3.18) and the hidden variable in a HMM model, Z is latent variable indicating if a jump took place, and q and $\boldsymbol{\pi}$ are parameters of interest. Here $f(Y|\beta)$ is the Poisson p.m.f. as defined in (3.21), or the Negative Binomial p.m.f. when overdispersion is taken into account. To compute the MLE estimates for q and $\boldsymbol{\pi}$ given the likelihood (A.89), we use an EM algorithm. To derive the updates for the EM algorithm, we start off by computing the expected complete-data log-likelihood under the current

parameter estimates at iteration t:

$$
\begin{aligned}
& Q(q, \boldsymbol{\pi}|q^{(t)}, \boldsymbol{\pi}^{(t)}) \\
= \; & E_{\beta, Z|Y, q^{(t)}, \boldsymbol{\pi}^{(t)}} \log(P(\boldsymbol{\beta}, \boldsymbol{Z}, \boldsymbol{Y}|q, \boldsymbol{\pi})) \\
= \; & \sum_{\beta} \sum_{Z} \log(P(\boldsymbol{\beta}, \boldsymbol{Z}, \boldsymbol{Y}|q, \boldsymbol{\pi})) P(\boldsymbol{\beta}|\boldsymbol{Z}, q^{(t)}\boldsymbol{\pi}^{(t)}, \boldsymbol{Y}) P(\boldsymbol{Z}|q^{(t)}\boldsymbol{\pi}^{(t)}, \boldsymbol{Y})) \\
= \; & \sum_{j} \log(\pi_j) P(\beta_1 = x_j|q^{(t)}, \boldsymbol{\pi}^{(t)}, \boldsymbol{Y}) + \\
& \sum_{b=2}^{B} \sum_{i,j} [\log(q\pi_j) P(\beta_{b-1} = x_i, \beta_b = x_j|Z_b = 1, q^{(t)}\boldsymbol{\pi}^{(t)}, \boldsymbol{Y}) P(Z_b = 1|q^{(t)}\boldsymbol{\pi}^{(t)}, \boldsymbol{Y}) + \\
& I_{\{j=i\}} \log(1-q) P(\beta_{b-1} = i, \beta_b = j|Z_b = 0, q^{(t)}\boldsymbol{\pi}^{(t)}, \boldsymbol{Y}) P(Z_b = 0|q^{(t)}\boldsymbol{\pi}^{(t)}, \boldsymbol{Y})] \quad \text{(A.90)}
\end{aligned}
$$

Hence, we wish to find the parameters q and $\boldsymbol{\pi}$ that maximize the quantity (A.90), with some additional constraints and possible priors for the parameters, to solve for the M step of the EM algorithm. Specifically, with the prior added in, we wish to solve for

$$
\begin{aligned}
& \underset{q,\pi}{\operatorname{argmax}} \sum_{j} \log(\pi_j) \gamma_1^{(t)}(j) + \sum_{b=2}^{B} \{ \sum_{i} [\sum_{j \neq i} \log(q\pi_j) \xi_{b-1}^{(t)}(i,j) + \\
& \xi_{b-1}(t)(i,i) (\log(q\pi_i) \frac{q^{(t)}\pi_i^{(t)}}{1 - q^{(t)} + q^{(t)}\pi_i^{(t)}} + \log(1-q) \frac{1 - q^{(t)}}{1 - q^{(t)} + q^{(t)}\pi_i^{(t)}})] \} + \\
& \sum_{j} (\lambda_j - 1) \log(\pi_j) + (\alpha - 1) \log(q) + (\theta - 1) \log(1-q) \\
& \text{s.t. } \boldsymbol{\pi}^T \mathbf{1} = 0, q \in (0,1) \qquad \text{(A.91)}
\end{aligned}
$$

where $\gamma_b^{(t)}(j) := P(\beta_b = x_j|q^{(t)}, \boldsymbol{\pi}^{(t)}, \boldsymbol{Y})$, $\xi_{b-1}^{(t)}(i,j) P(\beta_{b-1} = i, \beta_b = j|q^{(t)}\boldsymbol{\pi}^{(t)}, \boldsymbol{Y})$,

and λ, α, θ are the hyperparameters for the priors. Note that we have used conjugate priors here, i.e. Beta distribution for q and Dirichlet distribution for $\boldsymbol{\pi}$, so that analytical updates can be derived.

Solving for (A.91) results in the updates

$$q^{(t+1)} = 1 - \frac{(\theta-1) + 1 - q^{(t)} \sum_{b=2}^{B} \sum_i \frac{\xi_{b-1}^{(t)}(i,i)}{1-q^{(t)}+q^{(t)}\pi_i^{(t)}}}{(B-1)+(\alpha-1)+(\theta-1)} \tag{A.92}$$

$$\pi_j^{(t+1)} = \frac{\gamma_1^{(t)}(j) + \sum_{b=2}^{B}(\sum_{i \neq j} \xi_{b-1}^{(t)}(i,j) + \frac{q^{(t)}\pi_i^{(t)}}{1-q^{(t)}+q^{(t)}\pi_i^{(t)}} \xi_{b-1}^{(t)}(j,j)) + (\lambda_j - 1)}{1 + (B-1) - \sum_{b=1}^{B} \sum_j \frac{1-q^{(t)}}{1-q^{(t)}+q^{(t)}\pi_i^{(t)}} \xi_{b-1}^{(t)}(j,j) + (\sum_{k=1}^{K} \lambda_k - K)} \tag{A.93}$$

Details for computing the quantities γ and ξ needed in the EM updates is described in Appendix A.4.2.

A.4.2 Forward-Backward algorithm to compute γ_b and ξ_b

To compute the quantities γ and ξ described in Appendix A.4.1, we make use of the Forward-Backward algorithm. Specifically, we first compute the forward and backward variables defined by

$$\alpha_b(k) = P(Y_1, ..., Y_b, \beta_b = x_k | q, \pi) \tag{A.94}$$

$$\omega_b(k) = P(Y_{b+1}, ..., Y_B | \beta_b = x_k, q, \pi) \tag{A.95}$$

The algorithm for computing these are exactly as described in Rabiner (1989). However, while a standard HMM has free parameters for each transition and emission

probability, these probabilities have special forms in *HMM-seq*, given in (3.19) and (3.21) respectively. Most notably, the special form of the transition probability matrix allows the Forward-Backward algorithm to be computed in $O(K)$ time rather than $O(K^2)$ (keeping B fixed), where K is the number of hidden states.

Once we have obtained the forward and backward variables, γ and ξ can be computed by

$$\begin{aligned} \gamma_b(k) &= P(\beta_b^o = x_k | Y, q, \pi) \\ &= \frac{\alpha_b(k)\omega_b(k)}{\sum_{k=1}^K \alpha_b(k)\omega_b(k)} \end{aligned} \tag{A.96}$$

$$\begin{aligned} \xi_b(j,k) &= P(\beta_b^o = x_j, \beta_{b+1}^o = x_k | Y, q, \pi) \\ &= \frac{\alpha_b(j)\left((1-q)\mathbb{I}_{\{j=k\}} + q\pi_k\right) P(Y_{b+1}|\beta_{b+1}^o = x_k)\omega_{b+1}(k)}{\sum_{j=1}^K \sum_{k=1}^K \alpha_b(j)\left((1-q)\mathbb{I}_{\{j=k\}} + q\pi_k\right) P(Y_{b+1}|\beta_{b+1}^o = x_k)\omega_{b+1}(k)} \end{aligned} \tag{A.97}$$

A.5 Supplementary simulation results for *Multiseq* and *HMM-seq*

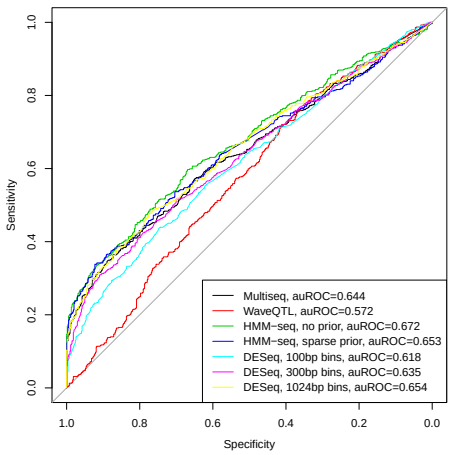
We present simulation results for DNase-NB and DNase-BB when the simulated effect size was estimated from the original data using *Multiseq*. Overall, the main patterns are similar to the results when the effect size was estimated using *bayesthr*, with performance for all the methods increasing with sample size. However, there are a few notable differences:

1. The differences between *Multiseq* and window-based methods are much smaller in general for all sample sizes. This is most likely because the effect sizes estimated using *bayesthr* tend to be be much narrower (often between 10bp and 50bp in length) than their counterparts estimated using *Multiseq*, which generates effect sizes that are broader (often between 100bp and 500bp in length). Note that most window-based approaches will not run properly when using bin sizes much smaller than 100bp, hence we use 100bp as the smallest possible bin size for window-based methods. Hence, window-based methods can pick up effects estimated using *Multiseq* much more easily than effects estimated using *bayesthr*, resulting in their improved performance.

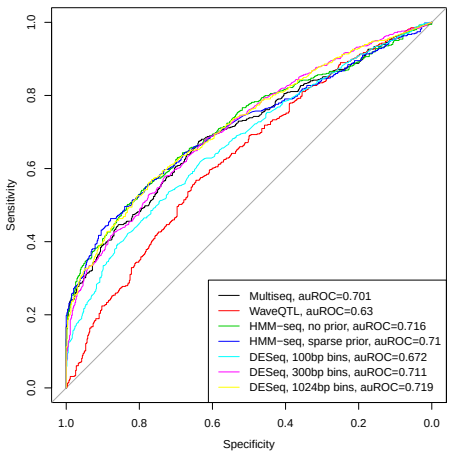
2. *WaveQTL* underperforms by a substantial margin for small sample sizes, when compared to the other methods. While the reason for this is not entirely known, the shape of the effects estimated using *Multiseq* is the most likely factor influencing the performance of *WaveQTL* in this case.

3. *HMM-seq* performs substantially worse for sample sizes 30 and 70. One possible

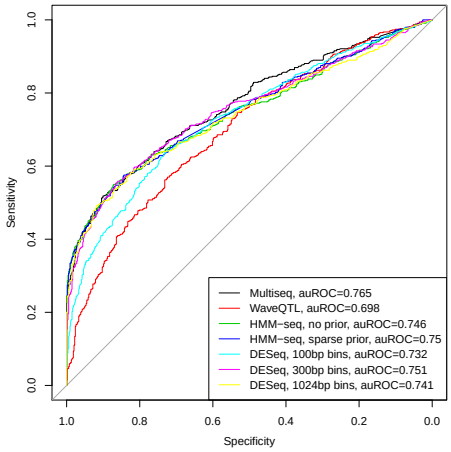
explanation is that effects estimated from *Multiseq* are much broader (as noted above) and much weaker than effects estimated from *bayesthr*, often being less than a 1.5-fold change. *HMM-seq* was not designed to pick up broad and weak effects well, while also being extremely sensitive to the baseline estimate from *Multiseq* for this kind of simulations. Hence, increased sample sizes do not increase power for *HMM-seq* as much when compared to other methods. This is also one of the main reasons for the non-robustness of *HMM-seq*.



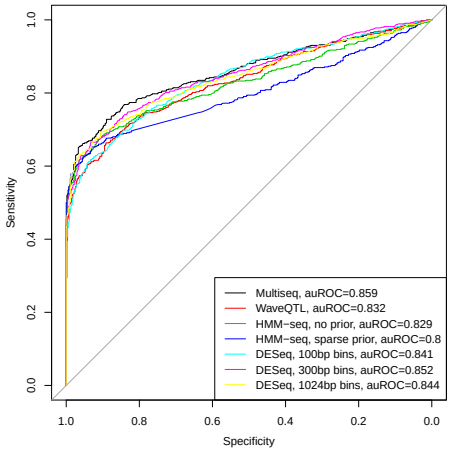
(a)



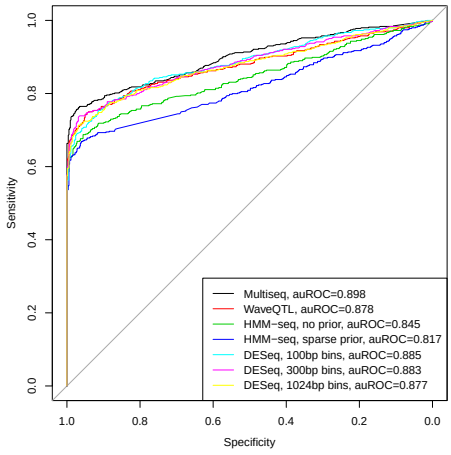
(b)



(c)

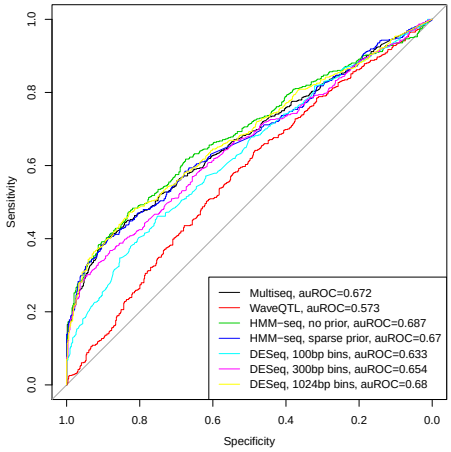


(d)

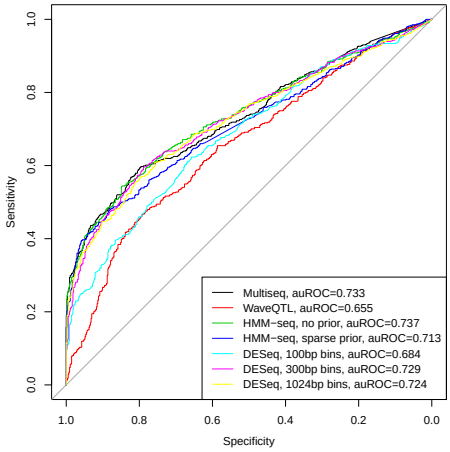


(e)

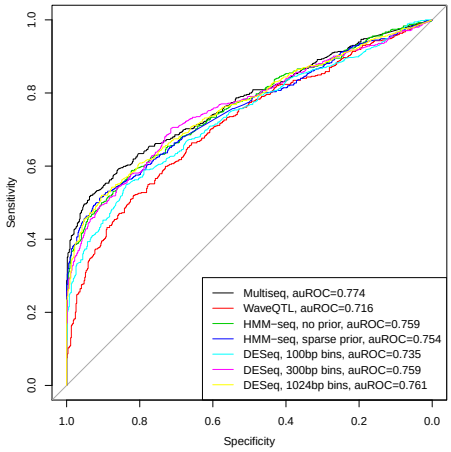
Figure A.2: ROC curves and areas under the curves for DNase-seq-based simulations under the Negative Binomial generation scheme, where the simulated effect is estimated from raw data using *multiseq*. For each of the 578 regions, one null case was simulated in which an effect is absent, and one non-null case was simulated in which an effect was present. Methods being compared include *Multiseq*, *WaveQTL*, *HMM-seq* with and without the sparse prior, and *DESeq* with various choices of window sizes. Panels (a), (b), (c), (d) and (e) show results when the sample sizes are 4, 6, 10, 30 and 70 respectively.



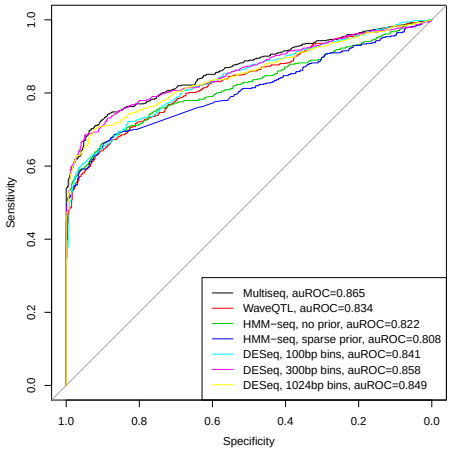
(a)



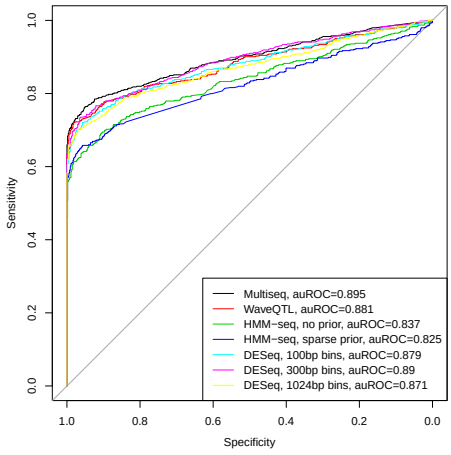
(b)



(c)



(d)



(e)

Figure A.3: ROC curves and areas under the curves for DNase-seq-based simulations under the Beta Binomial generation scheme, where the simulated effect is estimated from raw data using *multiseq*. For each of the 578 regions, one null case was simulated in which an effect is absent, and one non-null case was simulated in which an effect was present. Methods being compared include *Multiseq*, *WaveQTL*, *HMM-seq* with and without the sparse prior, and *DESeq* with various choices of window sizes. Panels (a), (b), (c), (d) and (e) show results when the sample sizes are 4, 6, 10, 30 and 70 respectively.

A.6 Supplementary results for inferring TF binding using *Multiseq*-based inference, CENTIPEDE and msCentipede

Table A.7: Comparison of accuracy in inferring binding sites using *Multiseq*-based inference, CENTIPEDE and msCentipede respectively. 43 TF-PWM pairs were considered. Performance is measured using auROCs for region sizes of 64bp, 128bp, 256bp, 512bp and 1024bp. For each TF-PWM pair, value(s) colored in red correspond to the largest auROC(s) across all region sizes, where tied values between methods will all be highlighted in red. In the case of ties within the same method, only one value will be highlighted.

TF	Region size	auROC		
		Multiseq	CENTIPEDE	msCentipede
Batf-S356	64	0.961	0.879	0.947
	128	0.984	0.921	0.963
	256	0.991	0.945	0.952
	512	0.980	0.964	0.949
	1024	0.977	0.879	0.951
Bhlhe40-S311	64	0.902	0.773	0.897
	128	0.933	0.838	0.929
	256	0.948	0.884	0.930
	512	0.951	0.909	0.926
	1024	0.947	0.643	0.910
Cebpb-S357	64	0.978	0.895	0.972
	128	0.992	0.952	0.986
	256	0.996	0.977	0.989

Continued on next page

Table A.7 – continued from previous page

TF	Region size	auROC		
		Multiseq	CENTIPEDE	msCentipede
	512	0.992	0.981	0.980
	1024	0.988	0.978	0.971
Ctcf-S2	64	0.884	0.856	0.892
	128	0.901	0.862	0.906
	256	0.903	0.840	0.900
	512	0.898	0.840	0.896
	1024	0.888	0.847	0.887
E2f4-S753	64	1.000	0.947	0.998
	128	0.996	0.959	0.993
	256	1.000	1.000	1.000
	512	1.000	1.000	1.000
	1024	1.000	1.000	1.000
E2f4-S754	64	0.999	0.998	0.998
	128	0.999	1.000	0.999
	256	1.000	1.000	1.000
	512	1.000	1.000	1.000
	1024	0.999	0.999	0.999
Ebf-S79	64	0.815	0.716	0.764
	128	0.857	0.769	0.812

Continued on next page

Table A.7 – continued from previous page

TF	Region size	auROC		
		Multiseq	CENTIPEDE	msCentipede
	256	0.873	0.807	0.814
	512	0.882	0.829	0.815
	1024	0.883	0.843	0.805
Egr1-S4	64	0.893	0.825	0.885
	128	0.918	0.867	0.913
	256	0.931	0.891	0.917
	512	0.935	0.535	0.922
	1024	0.943	0.536	0.912
Elf1-S81	64	0.921	0.874	0.929
	128	0.937	0.902	0.939
	256	0.942	0.915	0.933
	512	0.950	0.914	0.926
	1024	0.951	0.689	0.929
Elf1-S82	64	0.909	0.862	0.927
	128	0.921	0.889	0.933
	256	0.933	0.900	0.928
	512	0.939	0.904	0.915
	1024	0.943	0.630	0.905
Elk1-S841	64	0.993	0.987	0.993

Continued on next page

Table A.7 – continued from previous page

TF	Region size	auROC		
		Multiseq	CENTIPEDE	msCentipede
	128	0.996	0.994	0.995
	256	0.996	0.996	0.996
	512	0.996	0.997	0.996
	1024	0.994	0.800	0.996
Elk1-S88	64	0.997	0.991	0.997
	128	0.998	0.994	0.998
	256	0.999	0.997	0.998
	512	0.998	0.998	0.998
	1024	0.997	0.838	0.998
Elk1-S89	64	0.998	0.991	0.998
	128	0.999	0.996	0.999
	256	1.000	0.998	0.999
	512	0.999	0.999	0.999
	1024	0.999	0.815	0.999
Elk1-S90	64	1.000	1.000	1.000
	128	1.000	1.000	1.000
	256	1.000	1.000	1.000
	512	1.000	0.966	1.000
	1024	1.000	0.900	1.000

Continued on next page

Table A.7 – continued from previous page

TF	Region size	auROC		
		Multiseq	CENTIPEDE	msCentipede
Erra-S661	64	0.969	0.906	0.961
	128	0.990	0.910	0.973
	256	0.998	0.992	0.998
	512	0.998	0.998	0.998
	1024	0.996	0.996	0.983
Erra-S662	64	0.973	0.940	0.977
	128	0.998	0.974	0.994
	256	0.999	0.988	0.990
	512	0.999	0.999	0.996
	1024	0.998	0.998	0.997
Erra-S838	64	0.962	0.857	0.973
	128	0.995	0.941	0.991
	256	0.998	0.986	0.990
	512	0.995	0.984	0.966
	1024	0.984	0.967	0.952
Ets1-S100	64	0.999	0.996	0.999
	128	1.000	0.999	1.000
	256	0.999	0.998	1.000
	512	0.999	0.999	0.999

Continued on next page

Table A.7 – continued from previous page

TF	Region size	auROC		
		Multiseq	CENTIPEDE	msCentipede
	1024	0.999	0.903	0.999
Ets1-S101	64	1.000	1.000	1.000
	128	1.000	1.000	1.000
	256	1.000	1.000	1.000
	512	1.000	1.000	1.000
	1024	1.000	0.600	1.000
Ets1-S99	64	1.000	1.000	1.000
	128	1.000	1.000	1.000
	256	1.000	1.000	1.000
	512	1.000	1.000	1.000
	1024	1.000	0.952	0.999
Gabp-S116	64	0.970	0.956	0.973
	128	0.977	0.965	0.979
	256	0.981	0.971	0.980
	512	0.982	0.971	0.974
	1024	0.980	0.775	0.972
Irf3-S147	64	1.000	1.000	1.000
	128	1.000	1.000	1.000
	256	1.000	1.000	1.000

Continued on next page

Table A.7 – continued from previous page

TF	Region size	auROC		
		Multiseq	CENTIPEDE	msCentipede
	512	1.000	1.000	1.000
	1024	1.000	1.000	1.000
Irf4-S148	64	0.986	0.957	0.980
	128	0.991	0.994	0.993
	256	0.995	0.994	0.998
	512	0.995	0.992	0.990
	1024	0.991	0.880	0.992
Mafk-S383	64	0.613	0.603	0.696
	128	0.633	0.605	0.685
	256	0.648	0.614	0.644
	512	0.641	0.639	0.640
	1024	0.641	0.646	0.628
Mafk-S384	64	0.638	0.615	0.545
	128	0.656	0.638	0.678
	256	0.668	0.640	0.666
	512	0.669	0.649	0.661
	1024	0.663	0.653	0.637
Max-S325	64	0.989	0.989	0.989
	128	0.999	0.987	0.997

Continued on next page

Table A.7 – continued from previous page

TF	Region size	auROC		
		Multiseq	CENTIPEDE	msCentipede
	256	1.000	0.993	0.996
	512	1.000	0.999	0.998
	1024	0.999	0.995	0.998
Nfatc1-S186	64	0.940	0.819	0.930
	128	0.967	0.907	0.960
	256	0.983	0.960	0.979
	512	0.989	0.973	0.981
	1024	0.989	0.974	0.961
Nfatc1-S187	64	0.958	0.895	0.953
	128	0.974	0.940	0.979
	256	0.993	0.957	0.979
	512	0.993	0.979	0.983
	1024	0.990	0.967	0.978
Nfatc1-S188	64	0.983	0.871	0.974
	128	0.987	0.941	0.991
	256	0.993	0.963	0.962
	512	0.992	0.976	0.967
	1024	0.992	0.974	0.959
Nfe2-S392	64	1.000	0.999	1.000

Continued on next page

Table A.7 – continued from previous page

TF	Region size	auROC		
		Multiseq	CENTIPEDE	msCentipede
	128	1.000	0.999	1.000
	256	1.000	0.999	1.000
	512	1.000	0.999	1.000
	1024	1.000	0.997	1.000
Nfkb-S189	64	0.969	0.932	0.964
	128	0.981	0.946	0.976
	256	0.985	0.965	0.977
	512	0.986	0.972	0.978
	1024	0.985	0.753	0.972
Nfkb-S190	64	0.978	0.979	0.983
	128	0.991	0.986	0.991
	256	0.995	0.990	0.988
	512	0.996	0.990	0.985
	1024	0.992	0.814	0.981
Nrf1-S194	64	0.993	0.993	0.994
	128	0.995	0.995	0.995
	256	0.997	0.994	0.996
	512	0.997	0.774	0.994
	1024	0.997	0.767	0.997

Continued on next page

Table A.7 – continued from previous page

TF	Region size	auROC		
		Multiseq	CENTIPEDE	msCentipede
Pax5c20-S200	64	0.809	0.741	0.870
	128	0.849	0.789	0.889
	256	0.864	0.824	0.873
	512	0.869	0.845	0.846
	1024	0.869	0.650	0.827
Pu1-S123	64	0.770	0.703	0.810
	128	0.806	0.721	0.829
	256	0.831	0.749	0.821
	512	0.841	0.766	0.790
	1024	0.834	0.666	0.772
Rfx5-S240	64	0.990	0.984	0.982
	128	0.988	0.986	0.983
	256	0.986	0.986	0.985
	512	0.990	0.984	0.979
	1024	0.987	0.916	0.983
Rfx5-S241	64	0.974	0.972	0.971
	128	0.971	0.975	0.973
	256	0.975	0.975	0.970
	512	0.977	0.968	0.969

Continued on next page

Table A.7 – continued from previous page

TF	Region size	auROC		
		Multiseq	CENTIPEDE	msCentipede
	1024	0.973	0.862	0.973
Rfx5-S825	64	0.974	0.972	0.971
	128	0.971	0.975	0.973
	256	0.975	0.975	0.970
	512	0.977	0.968	0.969
	1024	0.973	0.862	0.973
Runx3-S250	64	0.849	0.778	0.865
	128	0.872	0.824	0.880
	256	0.892	0.852	0.877
	512	0.903	0.868	0.871
	1024	0.909	0.578	0.861
Srebp2-S342	64	1.000	1.000	1.000
	128	1.000	1.000	1.000
	256	1.000	1.000	1.000
	512	1.000	1.000	1.000
	1024	1.000	1.000	1.000
SrfV0416101-S159	64	0.834	0.657	0.847
	128	0.869	0.728	0.851
	256	0.895	0.790	0.840

Continued on next page

Table A.7 – continued from previous page

TF	Region size	auROC		
		Multiseq	CENTIPEDE	msCentipede
	512	0.898	0.827	0.836
	1024	0.899	0.844	0.808
SrfV0416101-S160	64	0.788	0.652	0.841
	128	0.835	0.693	0.864
	256	0.866	0.765	0.850
	512	0.869	0.799	0.804
	1024	0.863	0.812	0.780
Znf143-S43	64	0.967	0.837	0.962
	128	0.972	0.934	0.971
	256	0.976	0.953	0.954
	512	0.977	0.961	0.944
	1024	0.976	0.729	0.936

A.7 EM updates for *Cluster-seq*

To maximize the likelihood (4.1), we make use of an EM algorithm. Specifically, we first compute the expected complete-data log-likelihood given the parameters at iteration t, $\pi_{ik}^{(t)}$ and $\phi_{kb}^{(t)}$:

$$\begin{aligned} & Q(\boldsymbol{\pi}, \boldsymbol{\phi} | \boldsymbol{\pi}^{(t)}, \boldsymbol{\phi}^{(t)}) \\ = \; & E_{Z|R,\pi^{(t)},\phi^{(t)}}(\log(P(Z, R|\boldsymbol{\pi}, \boldsymbol{\phi}))) \\ = \; & \sum_{Z} P(\boldsymbol{Z}|\boldsymbol{R}, \boldsymbol{\pi}^{(t)}, \boldsymbol{\phi}^{(t)}) \log(P(\boldsymbol{R}, \boldsymbol{Z}|\boldsymbol{\pi}, \boldsymbol{\phi})) \\ = \; & \sum_{i} \sum_{j} \sum_{k} [P(Z_{ij} = k|\boldsymbol{R}, \boldsymbol{\pi}^{(t)}, \boldsymbol{\phi}^{(t)}) \log(\pi_{ik}) + \\ & \qquad P(Z_{ij} = k|\boldsymbol{R}, \boldsymbol{\pi}^{(t)}, \boldsymbol{\phi}^{(t)}) \log(\phi_{k,R_{ij}})] \end{aligned} \tag{A.98}$$

Hence, we need to compute

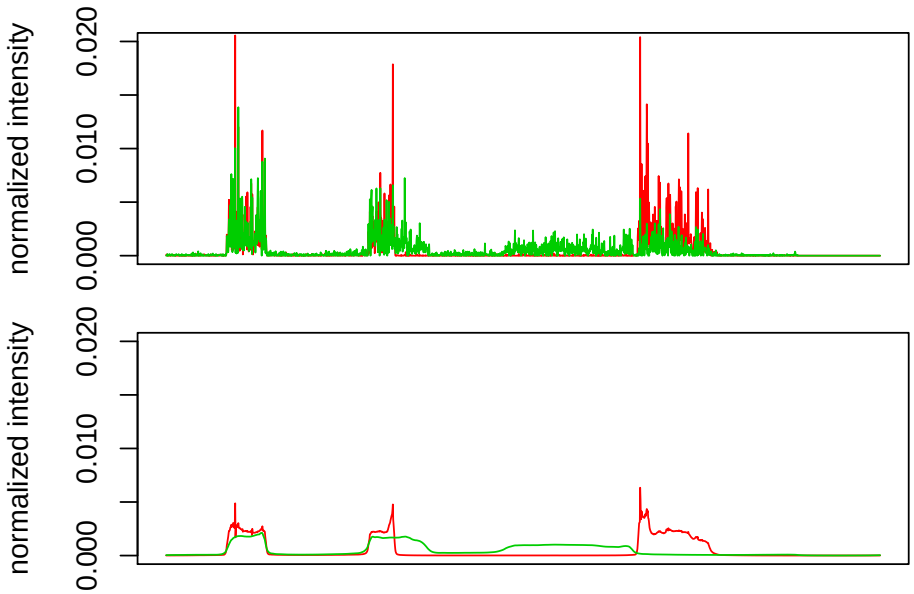
$$\begin{aligned} & \underset{\pi,\phi}{\operatorname{argmax}} \sum_{i} \sum_{j} \sum_{k} P(Z_{ij} = k|\boldsymbol{R}, \boldsymbol{\pi}^{(t)}, \boldsymbol{\phi}^{(t)})[\log(\pi_{ik}) + \log(\phi_{k,R_{ij}})] \\ & \text{s.t.} \sum_{k} \pi_{ik} = 1, \sum_{b} \phi_{kb} = 1 \end{aligned} \tag{A.99}$$

Solving (A.99) gives us the parameter updates for the M step:

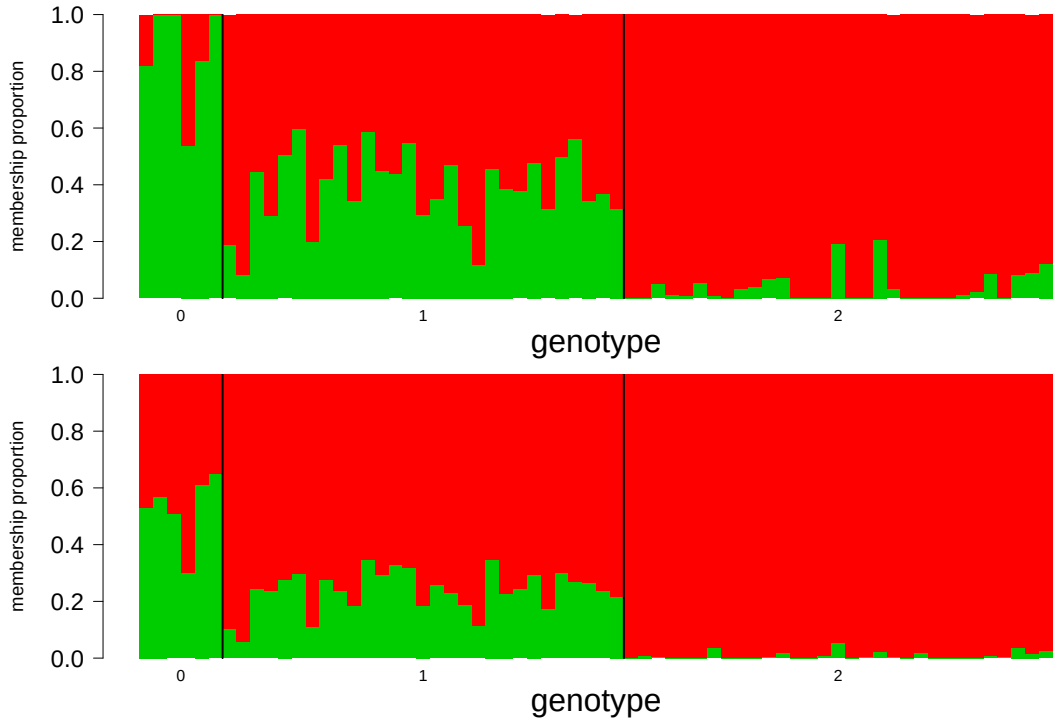
$$\begin{aligned}
\pi_{ik}^{(t+1)} &= \frac{\sum_{j:\text{read } j\in\text{sample } i} P(Z_{ij}=k|\boldsymbol{R},\boldsymbol{\pi}^{(t)},\boldsymbol{\phi}^{(t)})}{\sum_b' Y_{ib'}} \\
&= \frac{\sum_b Y_{ib}\left(\frac{\pi_{ik}^{(t)}\phi_{kb}^{(t)}}{\sum_{k'}\pi_{ik'}^{(t)}\phi_{k'b}^{(t)}}\right)}{\sum_{b'} Y_{ib'}} \qquad \text{(A.100)} \\
\phi_{kb}^{(t+1)} &= \frac{\sum_i \sum_{j:\text{read } j\in\text{base } b,\text{sample } i} P(Z_{ij}=k|\boldsymbol{R},\boldsymbol{\pi}^{(t)},\boldsymbol{\phi}^{(t)})}{\sum_i \sum_{j:\text{read } j\in\text{sample } i} P(Z_{ij}=k|\boldsymbol{R},\boldsymbol{\pi}^{(t)},\boldsymbol{\phi}^{(t)})} \\
&= \frac{\sum_i Y_{ib}\left(\frac{\pi_{ik}^{(t)}\phi_{kb}^{(t)}}{\sum_{k'}\pi_{ik'}^{(t)}\phi_{k'b}^{(t)}}\right)}{\sum_i \sum_{b'} Y_{ib'}\left(\frac{\pi_{ik}^{(t)}\phi_{kb'}^{(t)}}{\sum_{k'}\pi_{ik'}^{(t)}\phi_{k'b'}^{(t)}}\right)} \qquad \text{(A.101)}
\end{aligned}$$

This procedure is iterated until convergence, from whence we have the maximum likelihood estimators for $\boldsymbol{\pi}$ and $\boldsymbol{\phi}$.

A.8 Supplementary plots from running *Cluster-seq* on RNA-seq data for gene *OAS1*, with $K = 2$ and $K = 4$

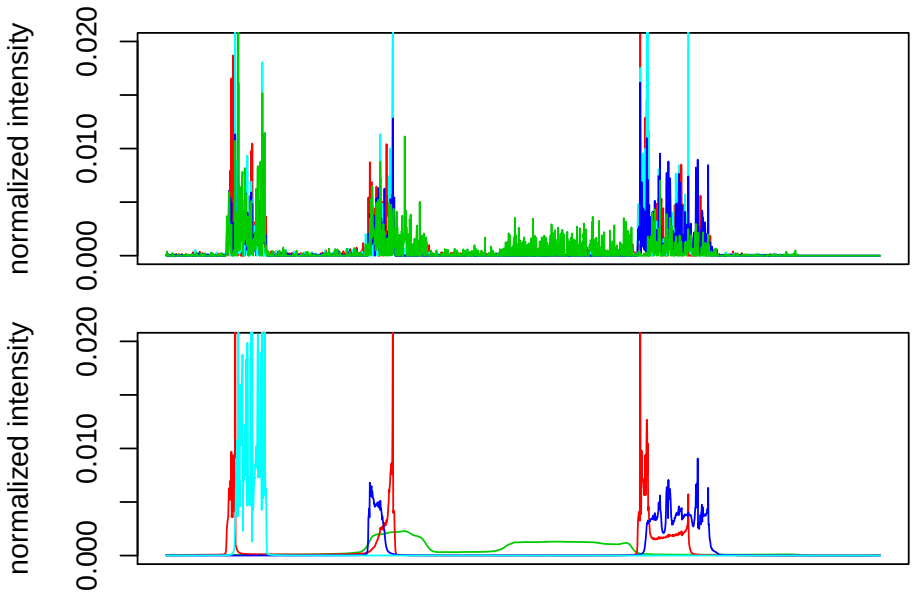


(a)

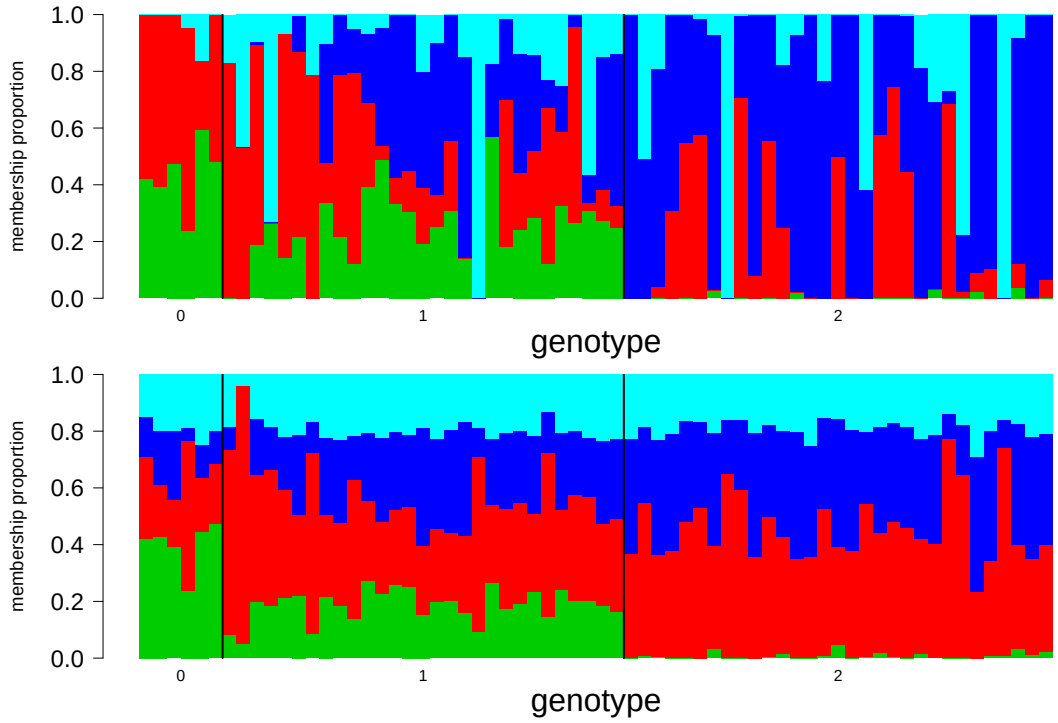


(b)

Figure A.4: Results from running the basic GoM model and *Cluster-seq* on RNA-seq data from gene *OAS1* with $K = 2$. Panel (a) shows the estimated cluster means; top plot in panel (a) corresponds to estimates from the basic GoM model, and bottom plot in panel (a) corresponds to estimates from *Cluster-seq*. Panel (b) shows the admixture plots; top plot in panel (b) corresponds to cluster memberships for the basic GoM model, and bottom plot in panel (b) corresponds to cluster memberships for *Cluster-seq*.

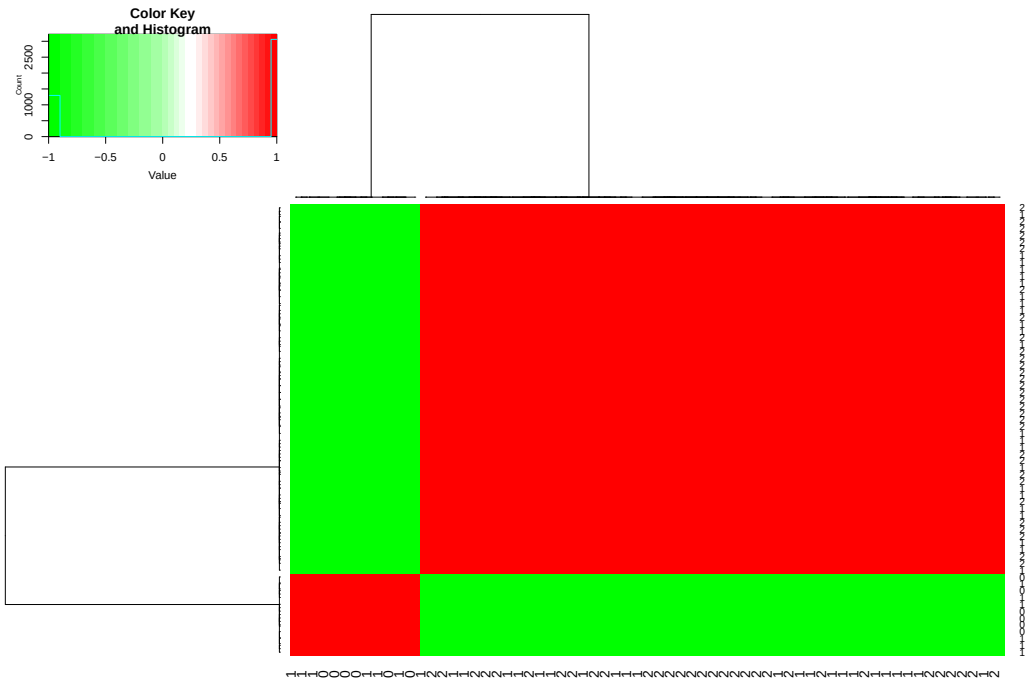


(a)

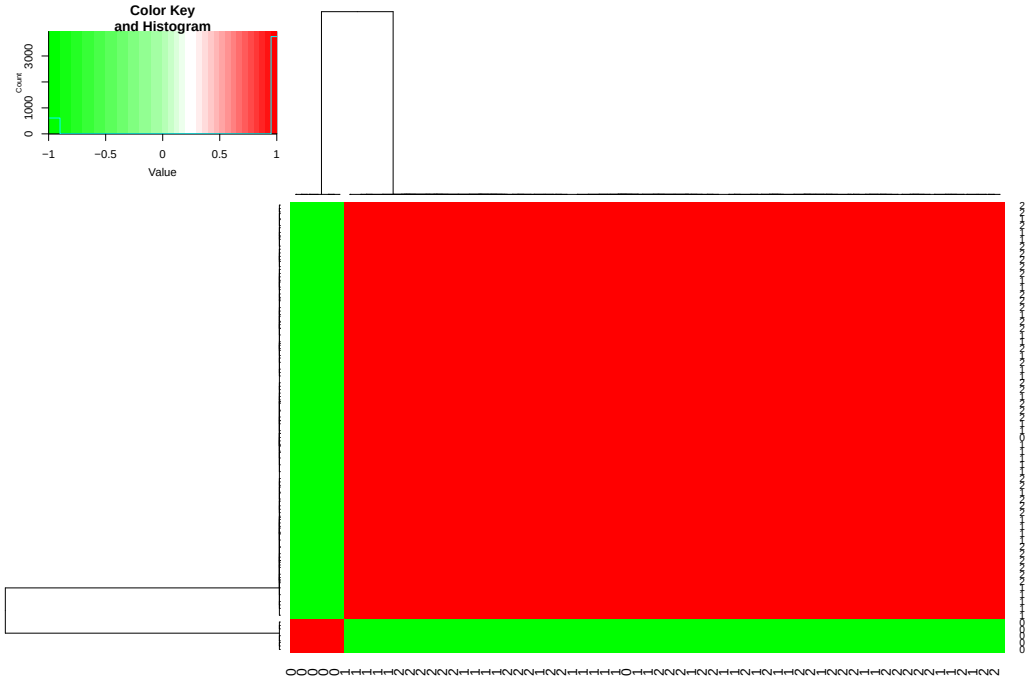


(b)

Figure A.5: Results from running the basic GoM model and *Cluster-seq* on RNA-seq data from gene *OAS1* with $K = 4$. Panel (a) shows the estimated cluster means; top plot in panel (a) corresponds to estimates from the basic GoM model, and bottom plot in panel (a) corresponds to estimates from *Cluster-seq*. Panel (b) shows the admixture plots; top plot in panel (b) corresponds to cluster memberships for the basic GoM model, and bottom plot in panel (b) corresponds to cluster memberships for *Cluster-seq*.

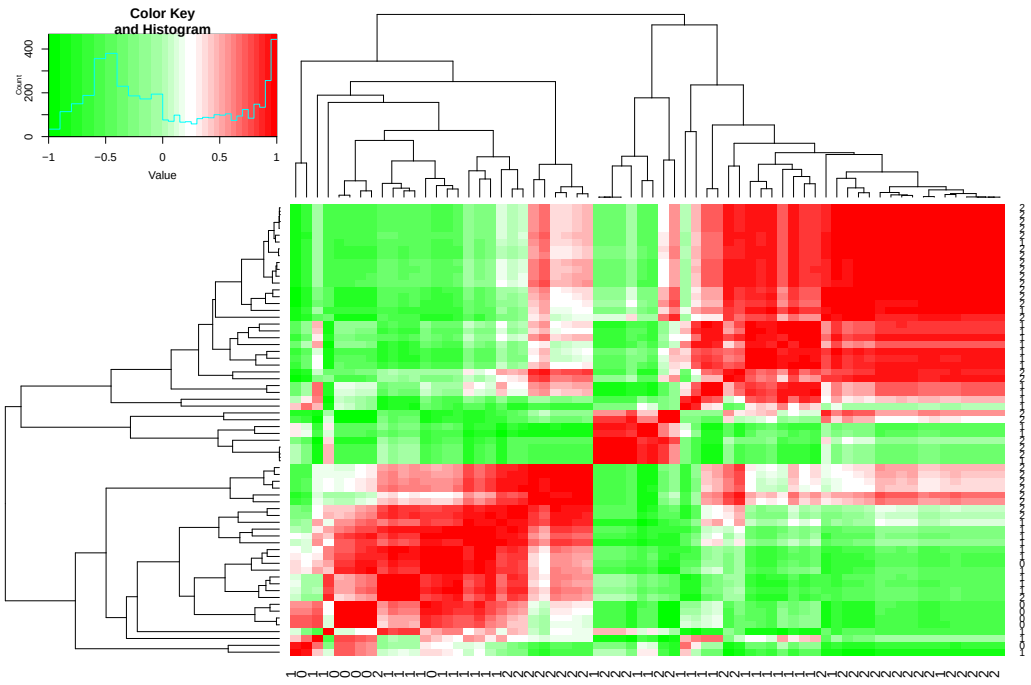


(a)

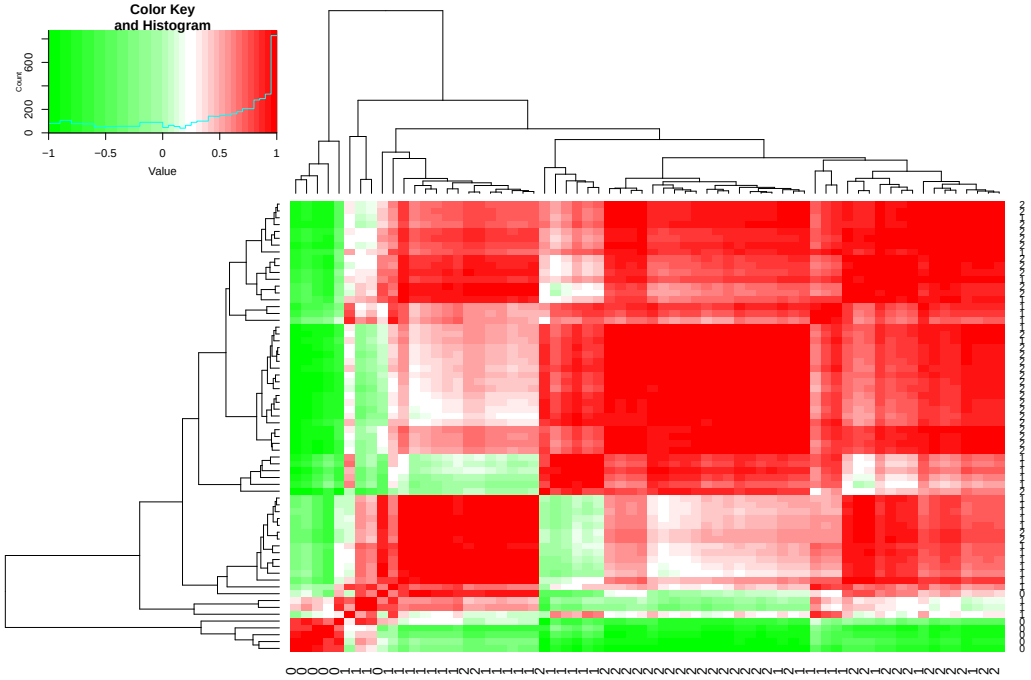


(b)

Figure A.6: Pairwise correlation heatmaps for both the basic GoM model (panel (a)) and *Cluster-seq* (panel (b)), $K = 2$. Each grid on the heatmap denotes the between sample correlation of the cluster memberships π, color-coded according to the amount of correlation. A hierarchical clustering of the correlations further reveals (dis)similarities between samples.



(a)



(b)

Figure A.7: Pairwise correlation heatmaps for both the basic GoM model (panel (a)) and *Cluster-seq* (panel (b)), $K = 4$. Each grid on the heatmap denotes the between sample correlation of the cluster memberships π, color-coded according to the amount of correlation. A hierarchical clustering of the correlations further reveals (dis)similarities between samples.

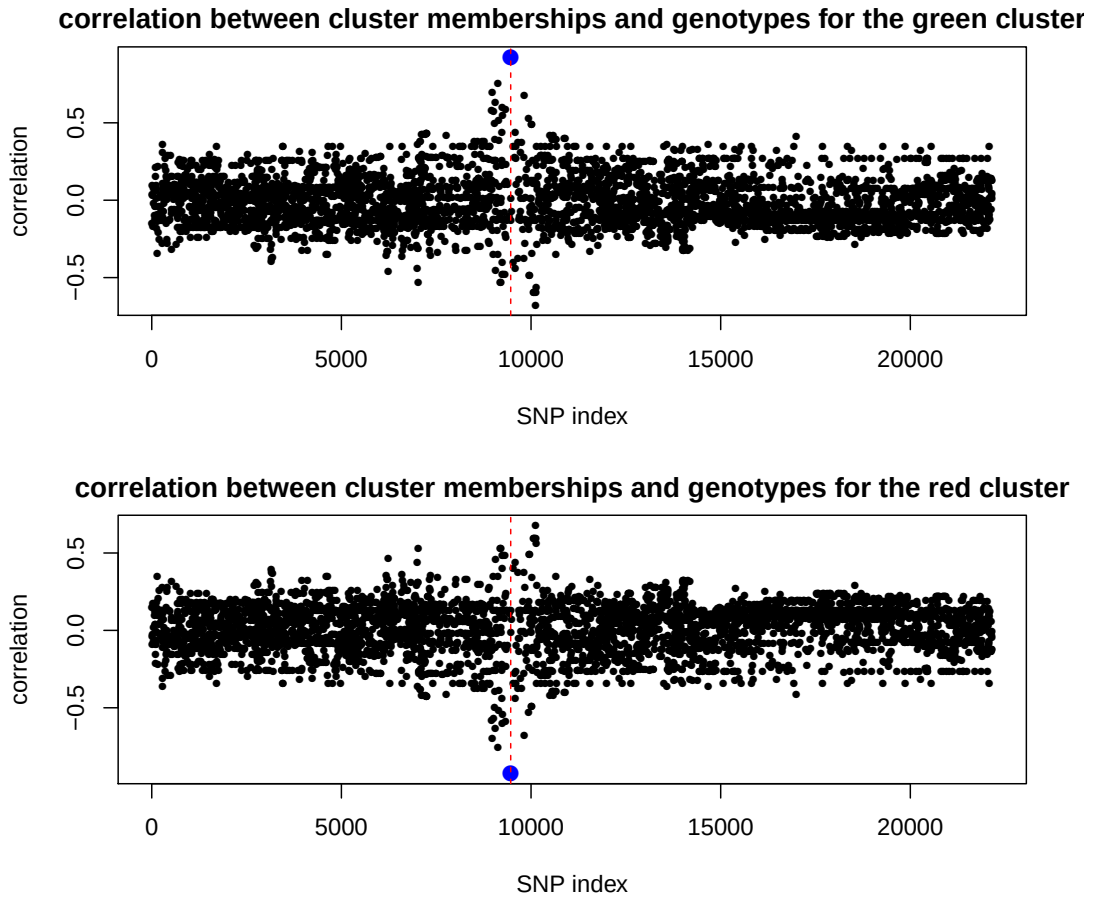


Figure A.8: Correlation plots between cluster memberships estimated using the basic GoM model ($K = 2$) and genotype classes for each SNP located within and near gene *OAS1*. The three plots correspond to the three clusters. For each cluster, the SNP most significantly associated with the cluster memberships is colored in blue, while the location of rs10774671 is marked by the vertical dashed red line.

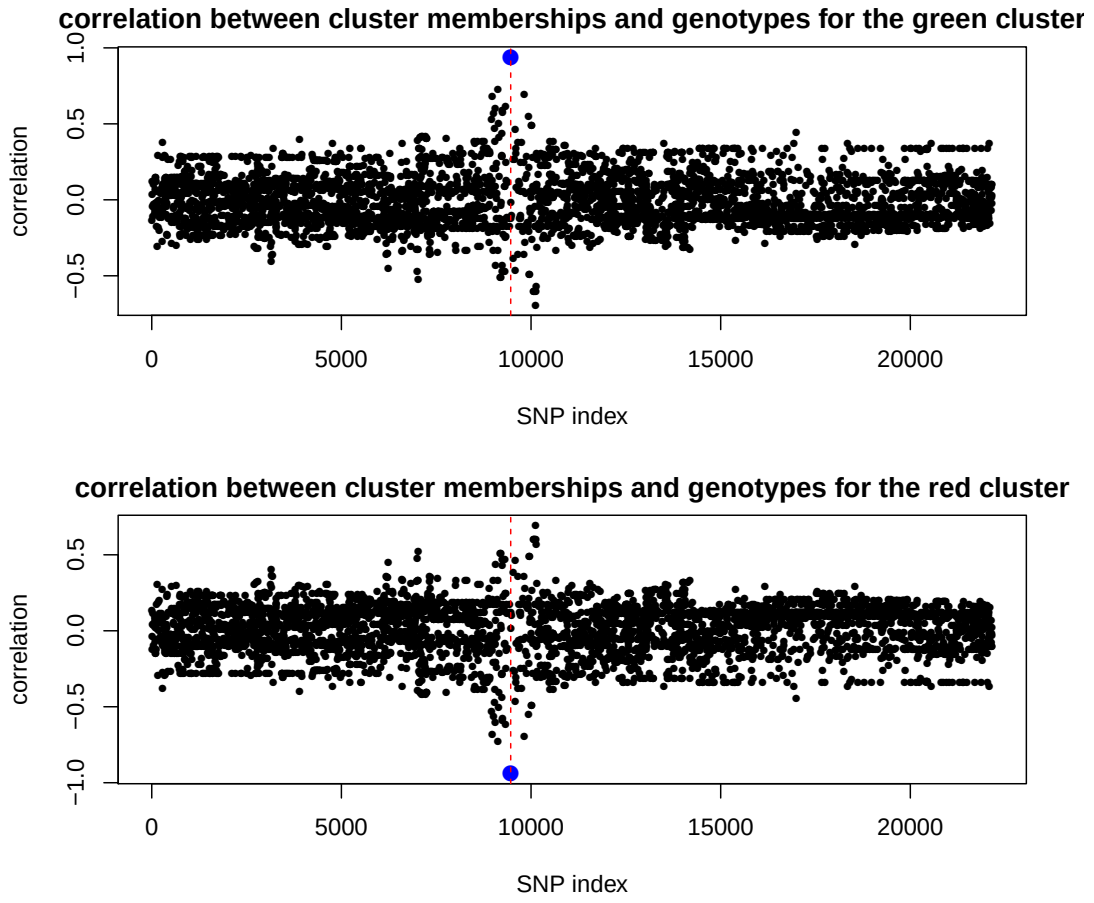


Figure A.9: Correlation plots between cluster memberships estimated using *Cluster-seq* ($K = 2$) and genotype classes for each SNP located within and near gene *OAS1*. The three plots correspond to the three clusters. For each cluster, the SNP most significantly associated with the cluster memberships is colored in blue, while the location of rs10774671 is marked by the vertical dashed red line.

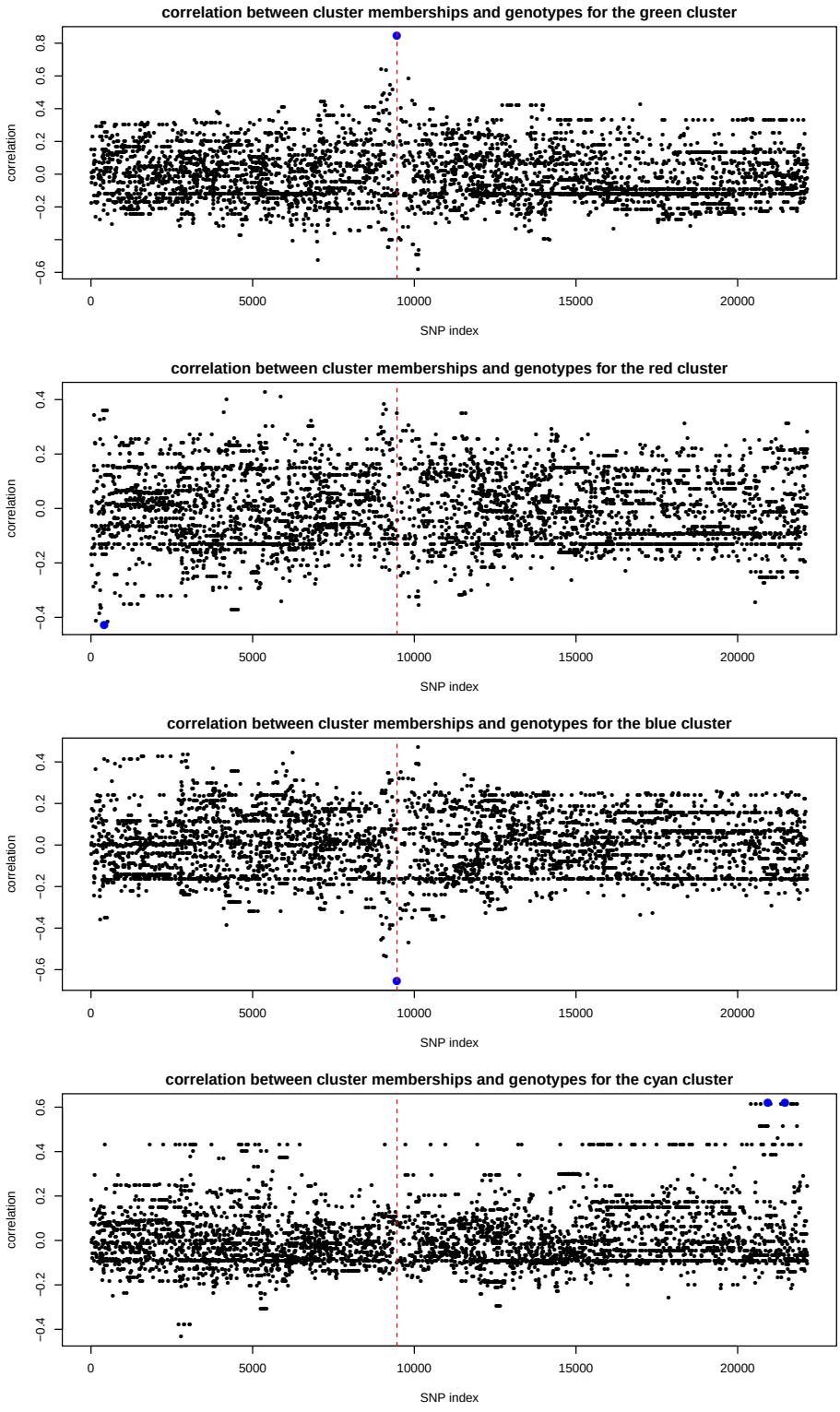


Figure A.10: Correlation plots between cluster memberships estimated using the basic GoM model ($K = 4$) and genotype classes for each SNP located within and near gene *OAS1*. The three plots correspond to the three clusters. For each cluster, the SNP most significantly associated with the cluster memberships is colored in blue, while the location of rs10774671 is marked by the vertical dashed red line.

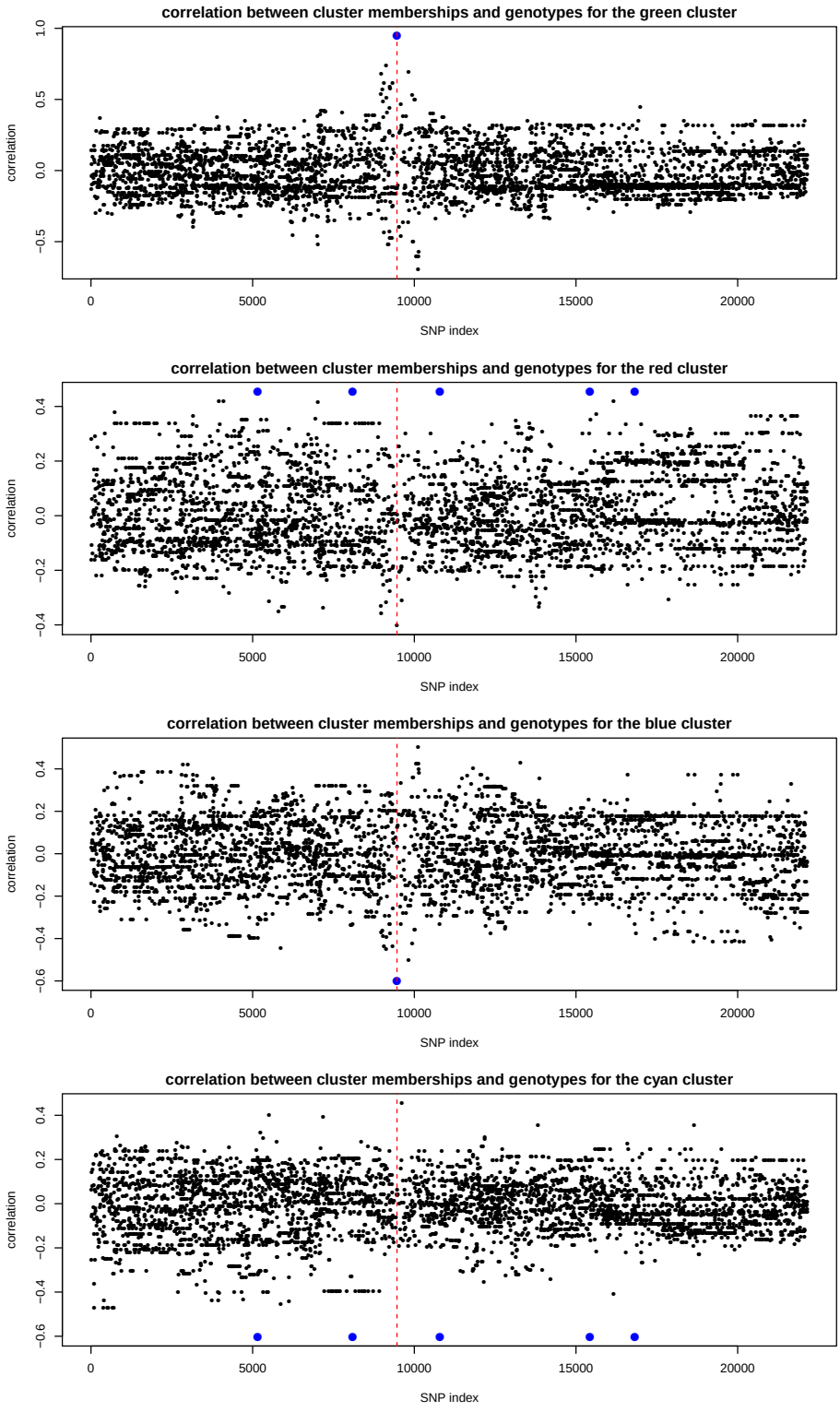
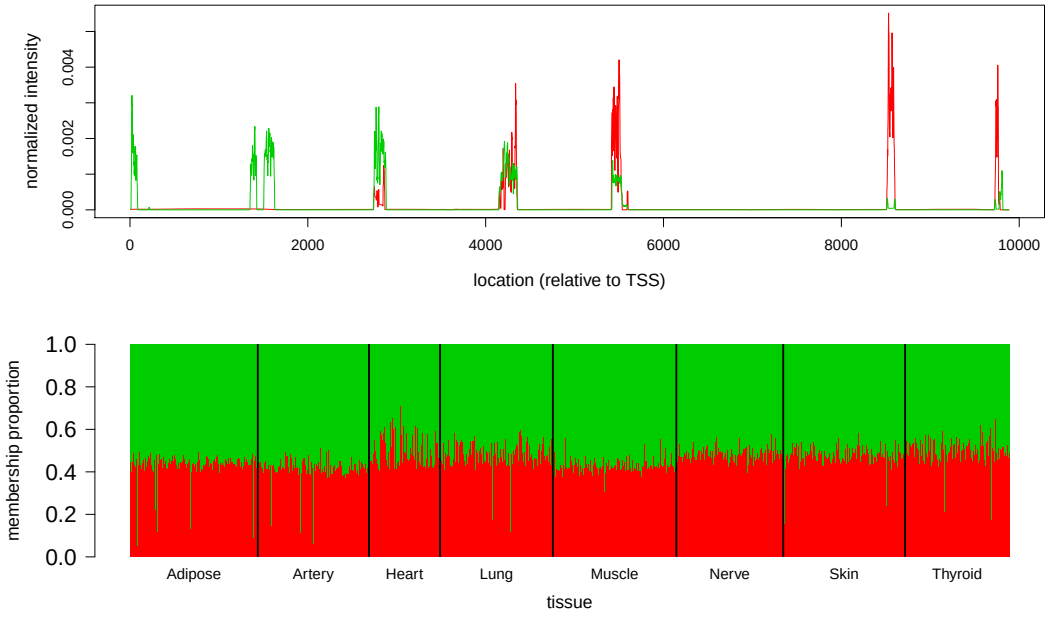


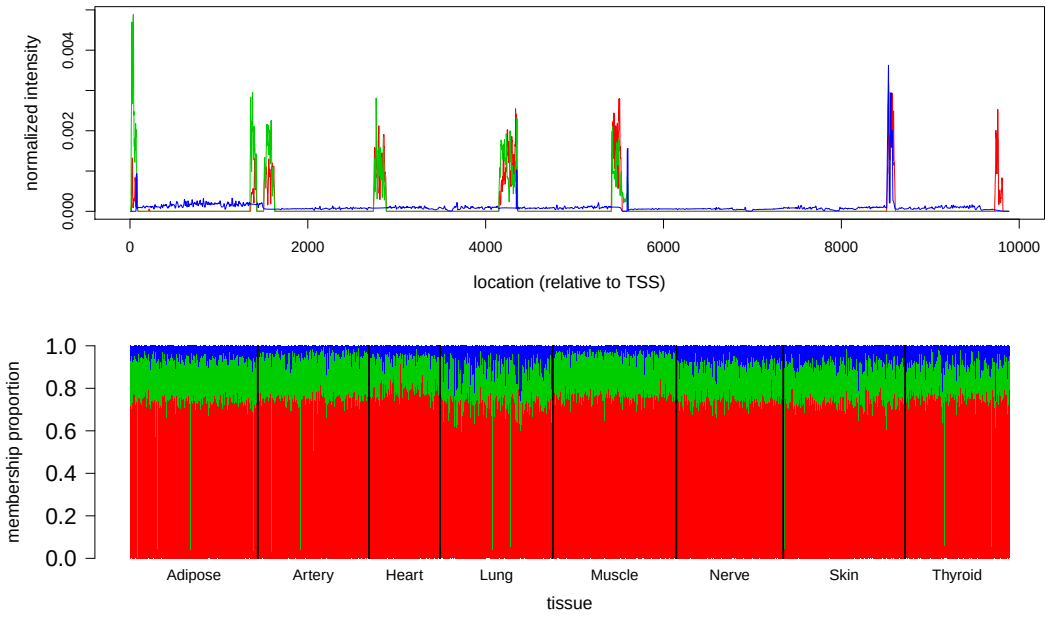
Figure A.11: Correlation plots between cluster memberships estimated using *Cluster-seq* ($K = 4$) and genotype classes for each SNP located within and near gene *OAS1*. The three plots correspond to the three clusters. For each cluster, the SNP most significantly associated with the cluster memberships is colored in blue, while the location of rs10774671 is marked by the vertical dashed red line.

A.9 Supplementary plots from running *Cluster-seq* on RNA-seq data for the GTEx project

Figure A.12: Results from running *Cluster-seq* on RNA-seq data from gene *RPL5* with $K = 2, 3, 4$ and 5, corresponding to panels (a), (b), (c) and (d) respectively. Top plot in each panel shows the estimated cluster means, while bottom plot in each panel shows the corresponding admixture plot.



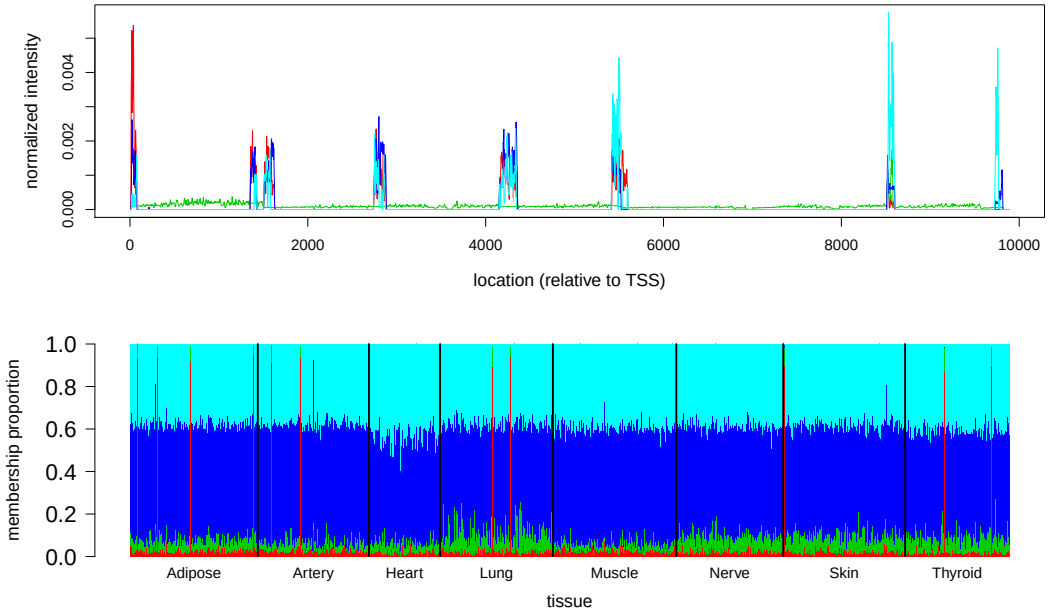
(a)



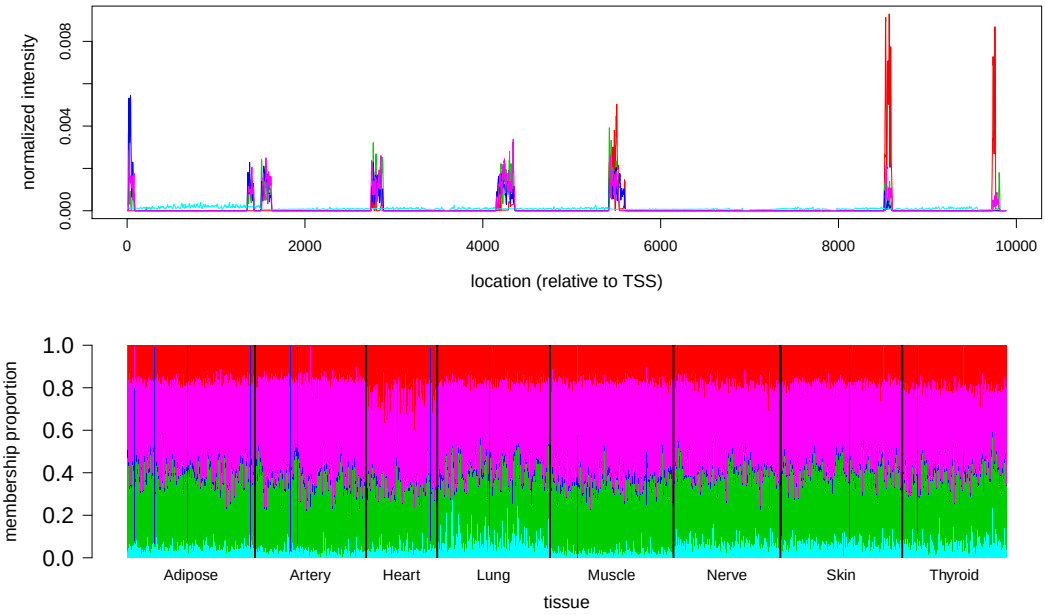
(b)

Continued on next page

– continued from previous page

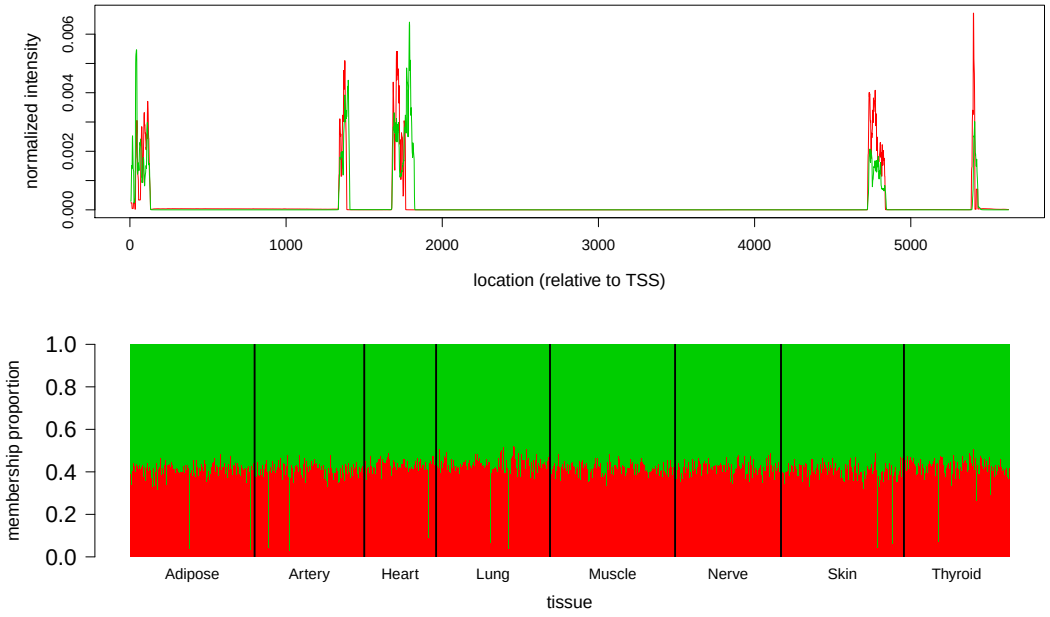


(c)

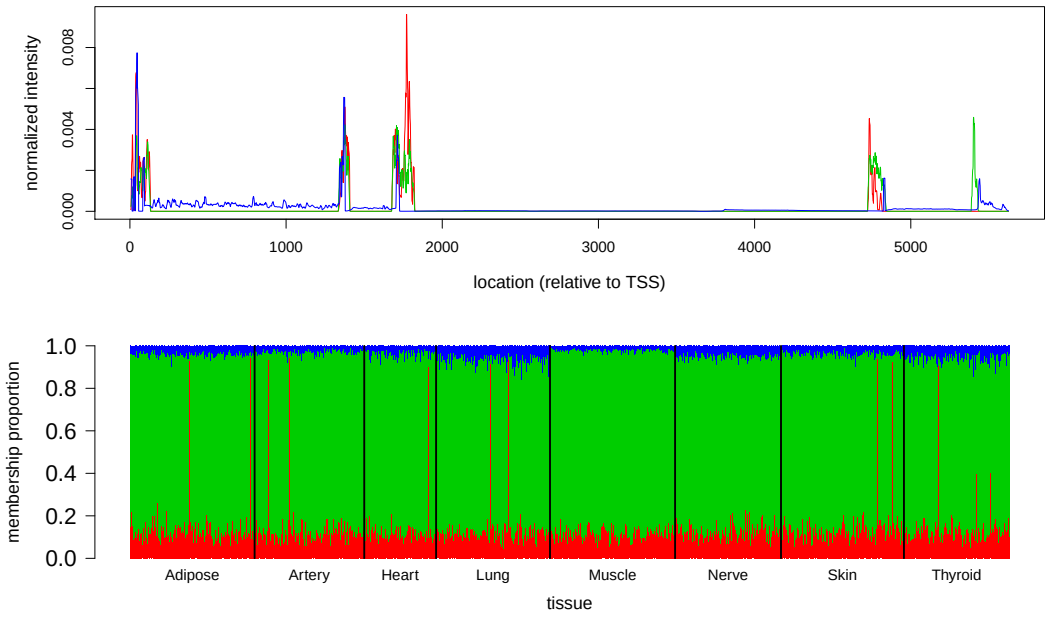


(d)

Figure A.13: Results from running *Cluster-seq* on RNA-seq data from gene *RPL24* with $K = 2, 3, 4$ and 5, corresponding to panels (a), (b), (c) and (d) respectively. Top plot in each panel shows the estimated cluster means, while bottom plot in each panel shows the corresponding admixture plot.



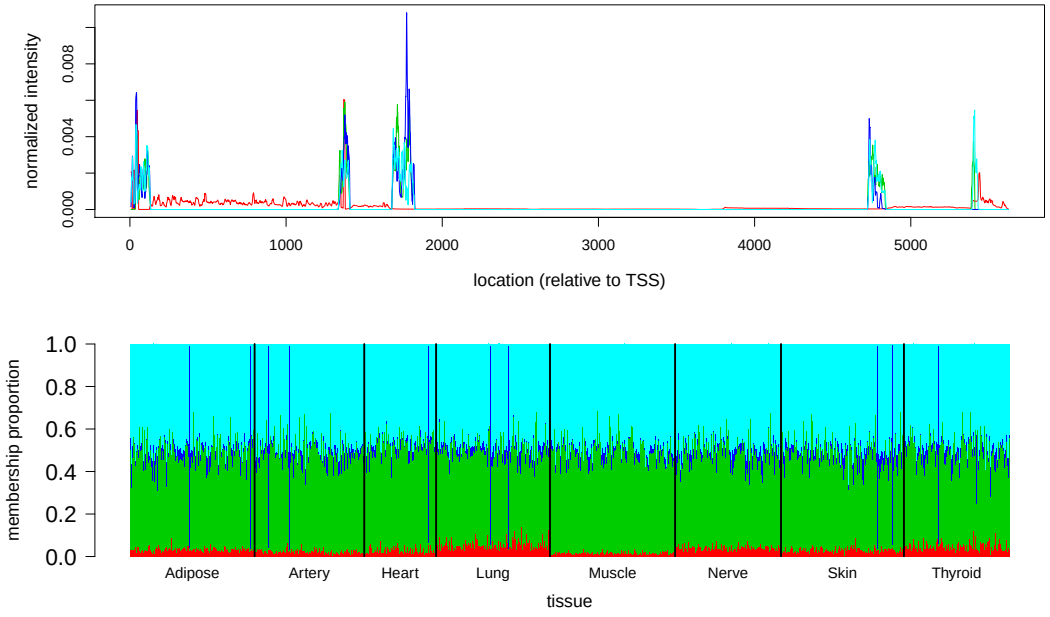
(a)



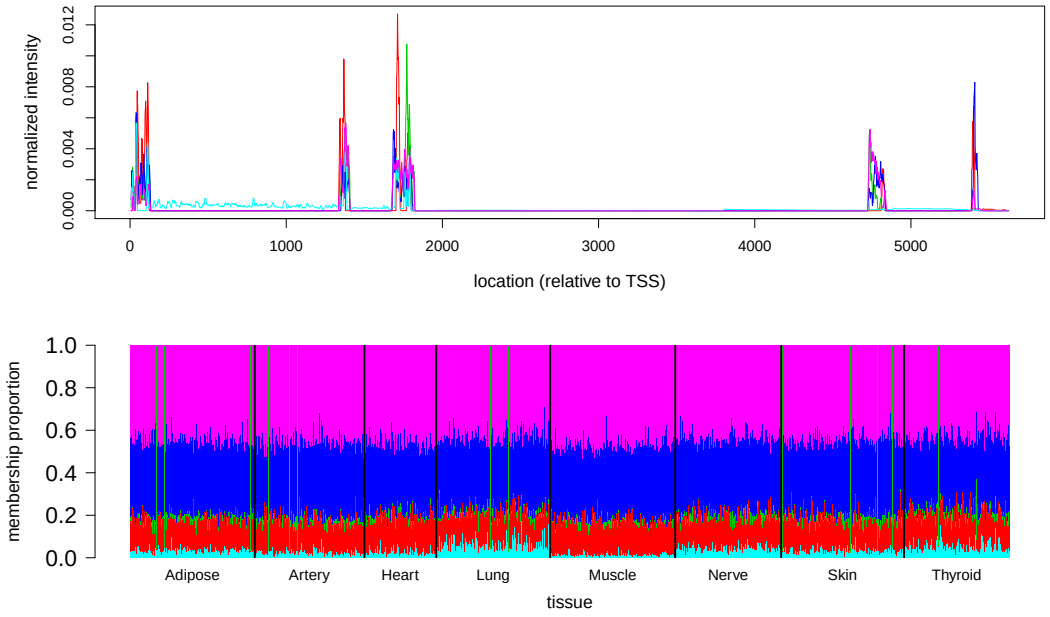
(b)

Continued on next page

– continued from previous page

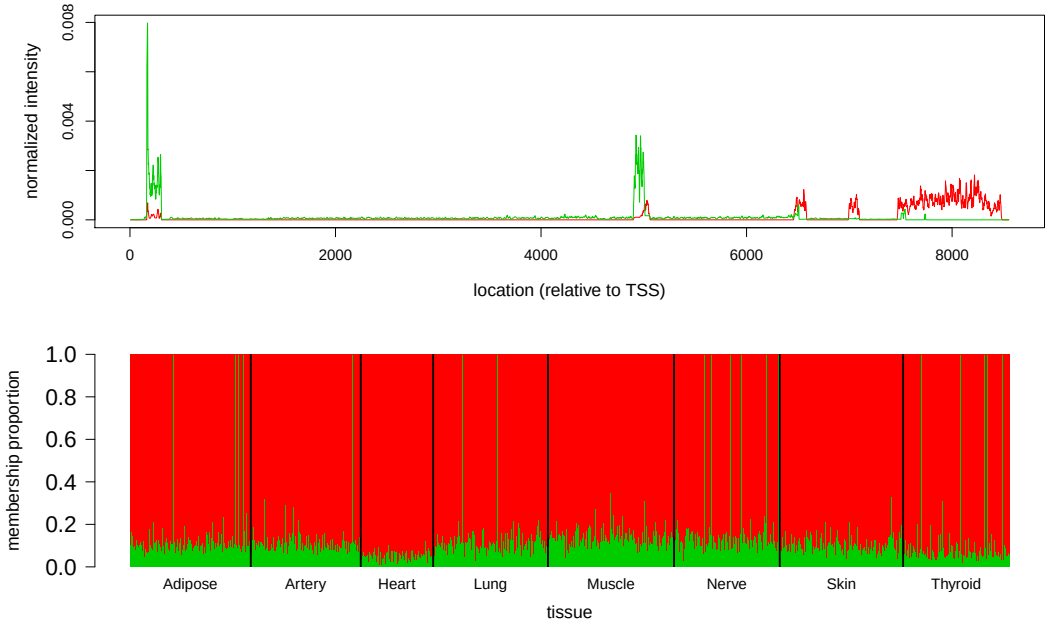


(c)

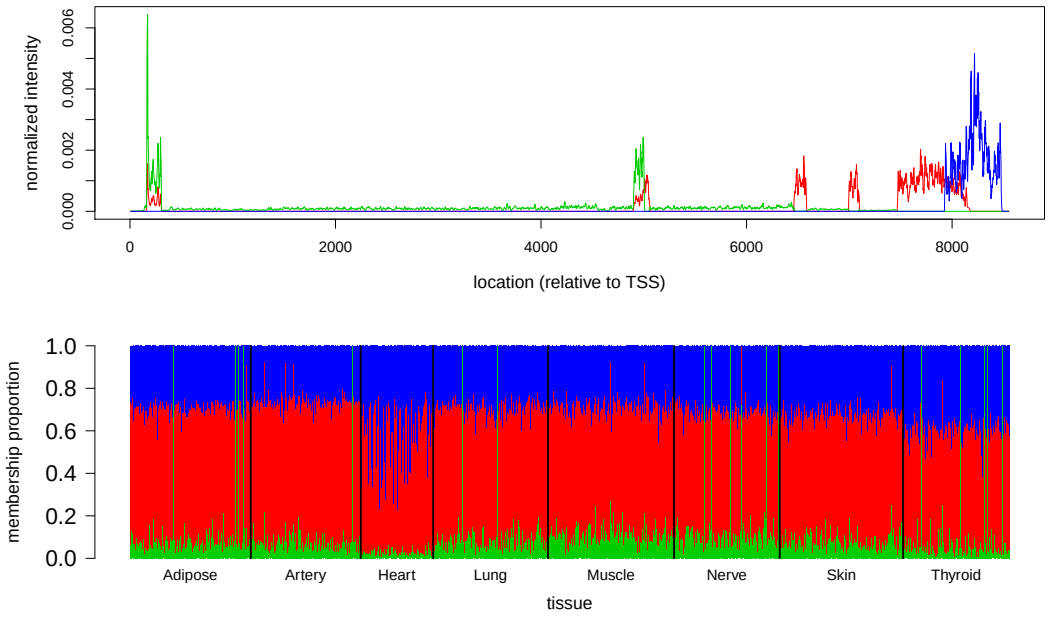


(d)

Figure A.14: Results from running *Cluster-seq* on RNA-seq data from gene *GPX3* with $K = 2, 3, 4$ and 5, corresponding to panels (a), (b), (c) and (d) respectively. Top plot in each panel shows the estimated cluster means, while bottom plot in each panel shows the corresponding admixture plot.



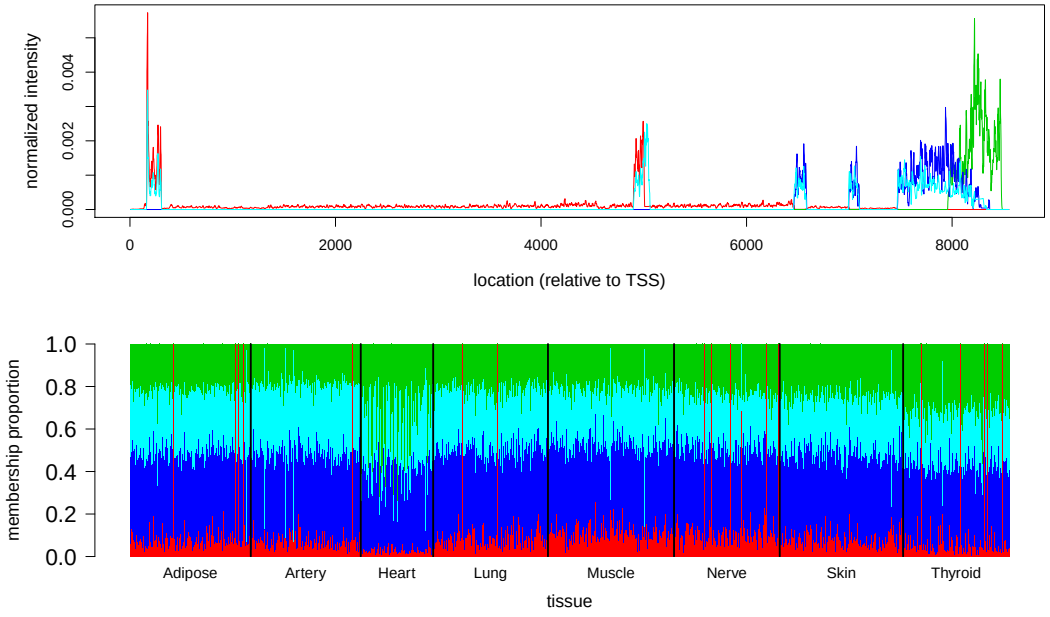
(a)



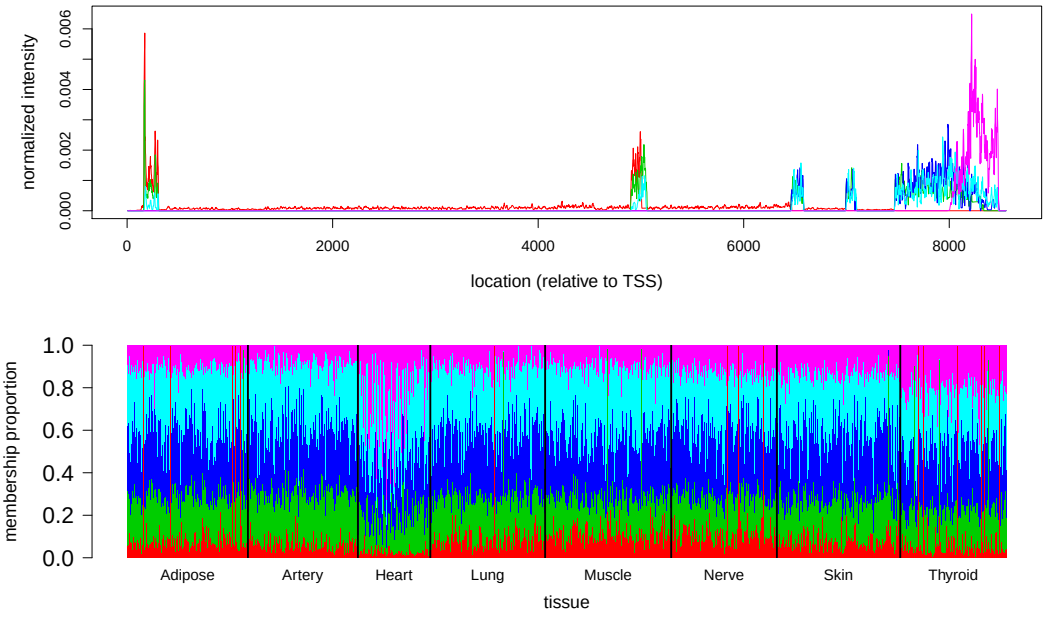
(b)

Continued on next page

– continued from previous page

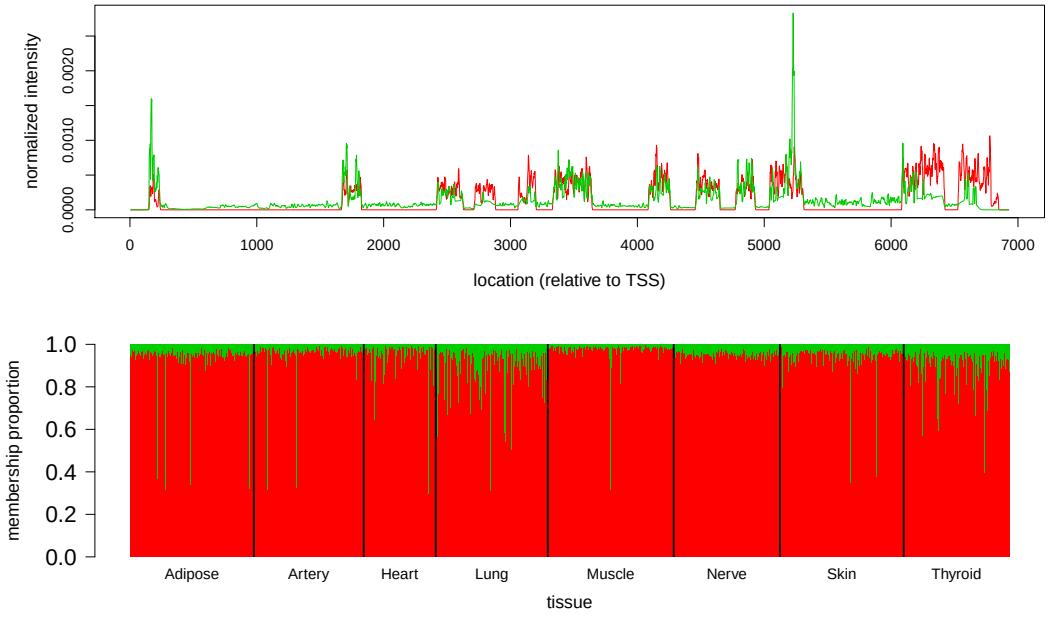


(c)

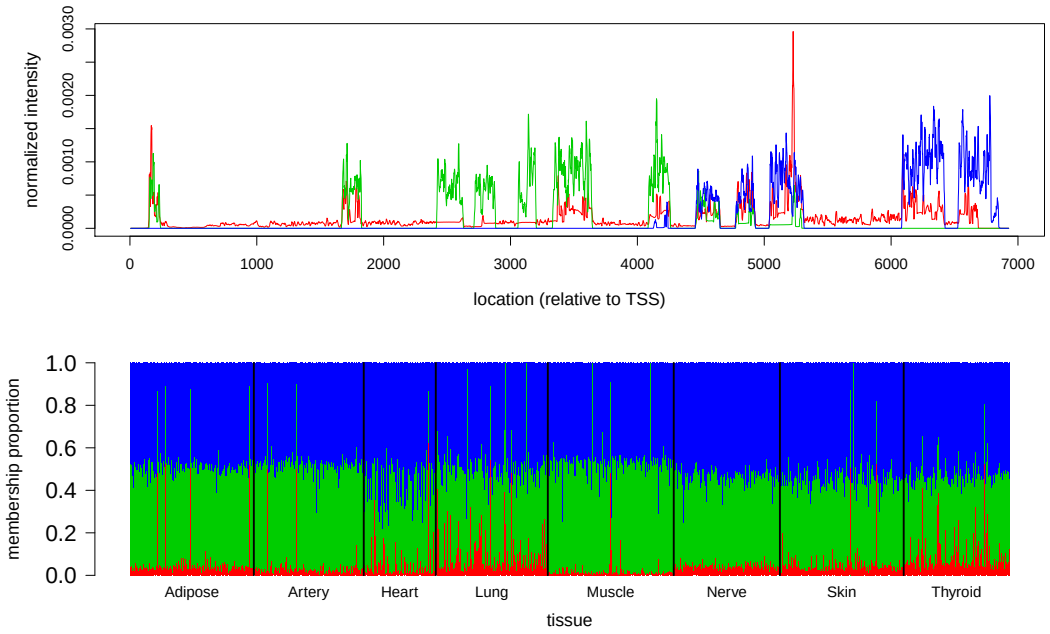


(d)

Figure A.15: Results from running *Cluster-seq* on RNA-seq data from gene *HSP90AB1* with $K = 2, 3, 4$ and 5, corresponding to panels (a), (b), (c) and (d) respectively. Top plot in each panel shows the estimated cluster means, while bottom plot in each panel shows the corresponding admixture plot.



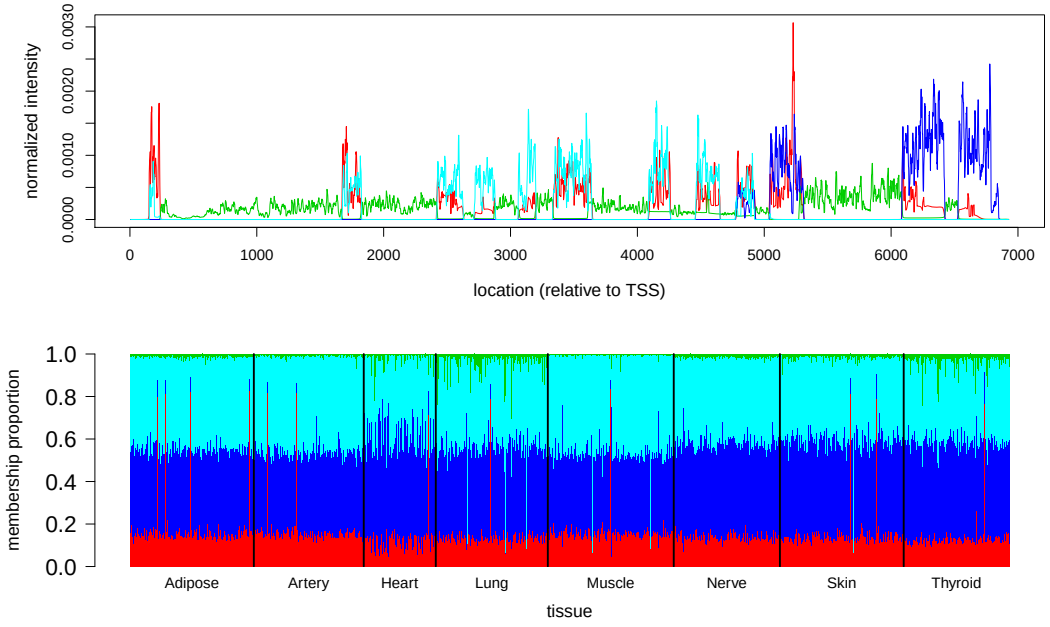
(a)



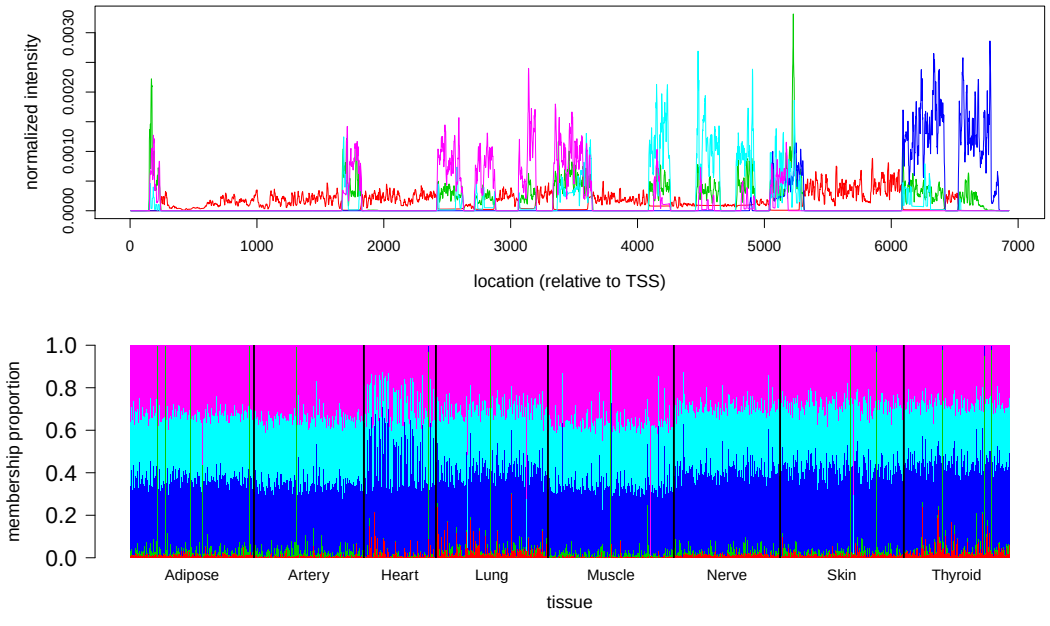
(b)

Continued on next page

– continued from previous page

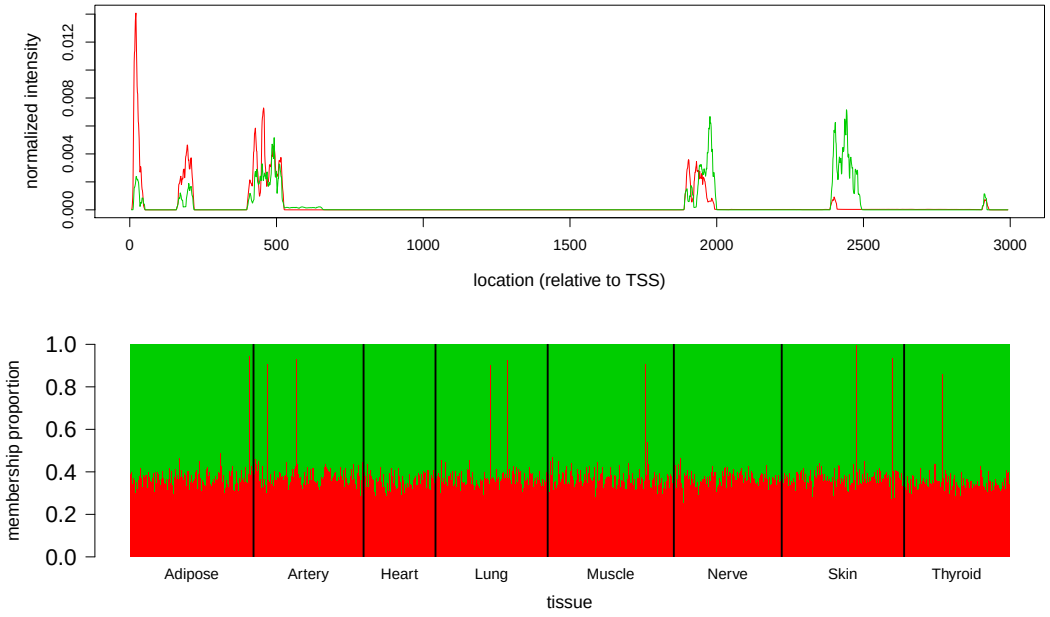


(c)

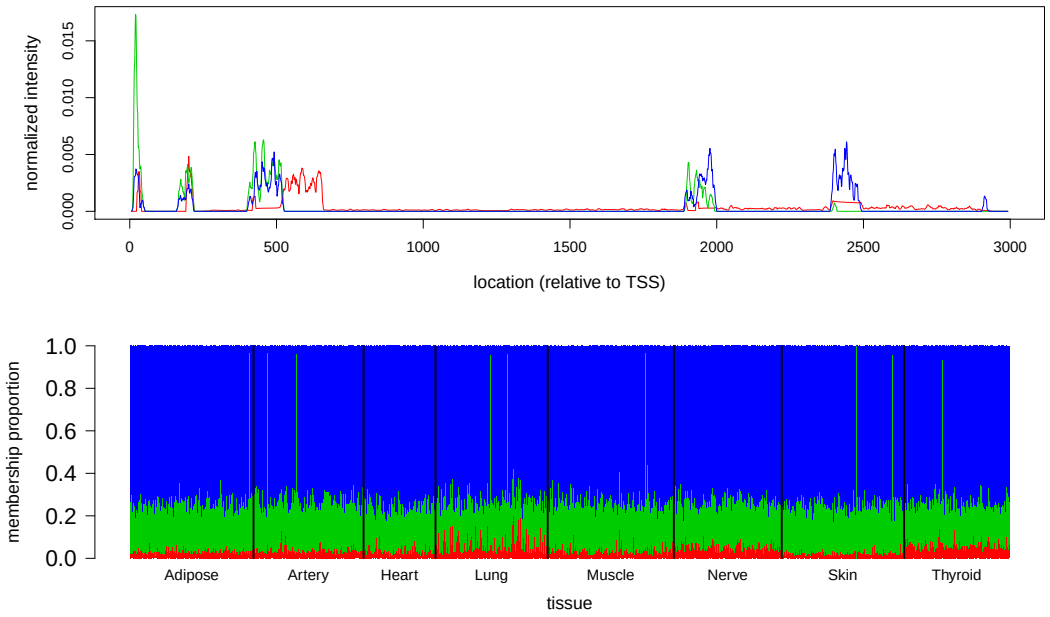


(d)

Figure A.16: Results from running *Cluster-seq* on RNA-seq data from gene *RPS12* with $K = 2, 3, 4$ and 5, corresponding to panels (a), (b), (c) and (d) respectively. Top plot in each panel shows the estimated cluster means, while bottom plot in each panel shows the corresponding admixture plot.



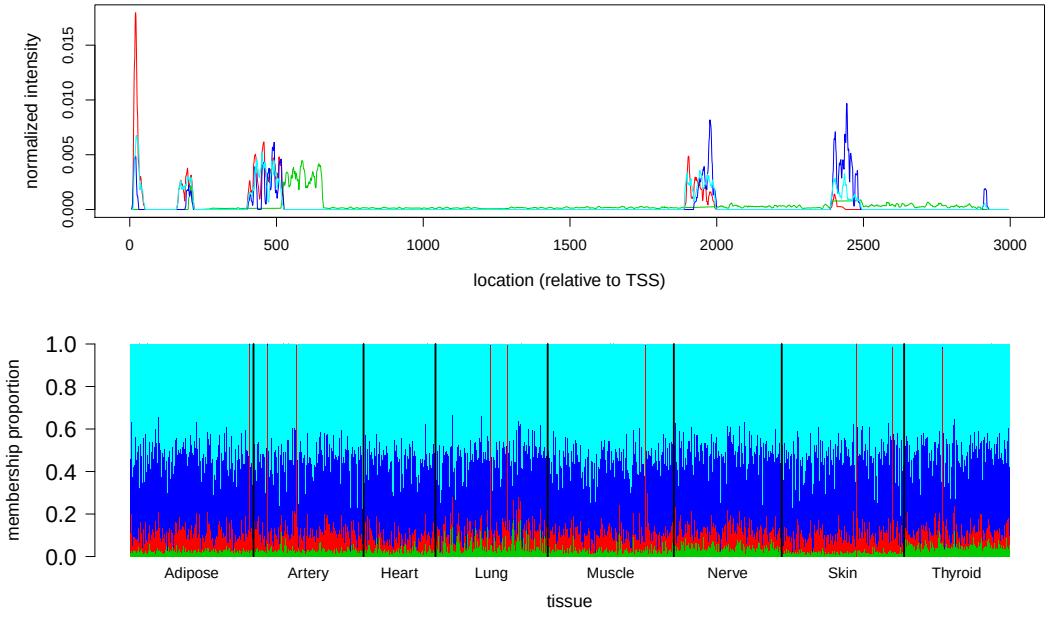
(a)



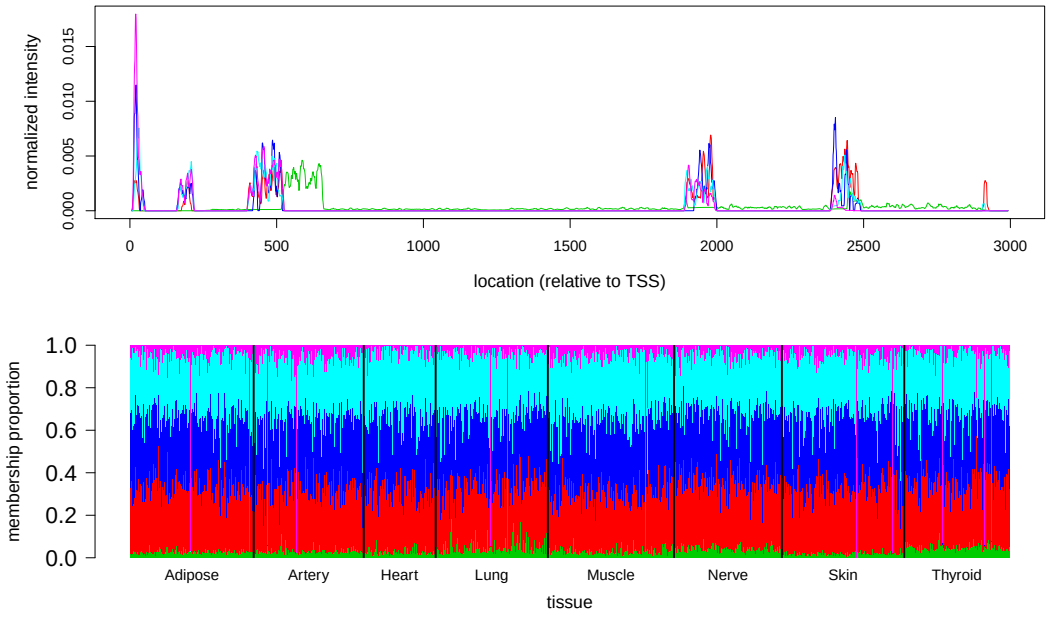
(b)

Continued on next page

– continued from previous page

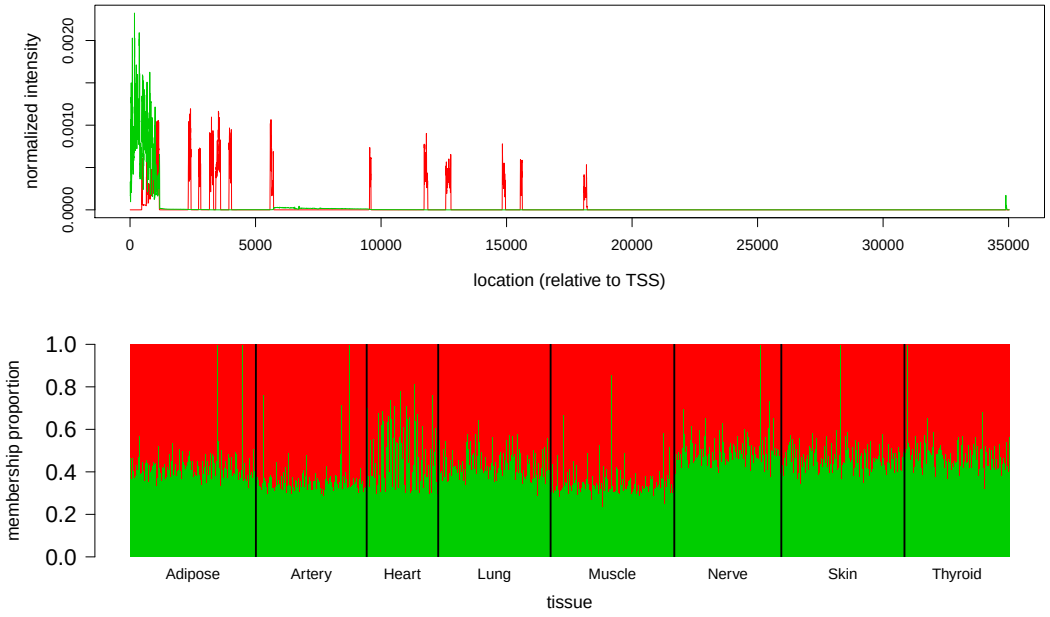


(c)

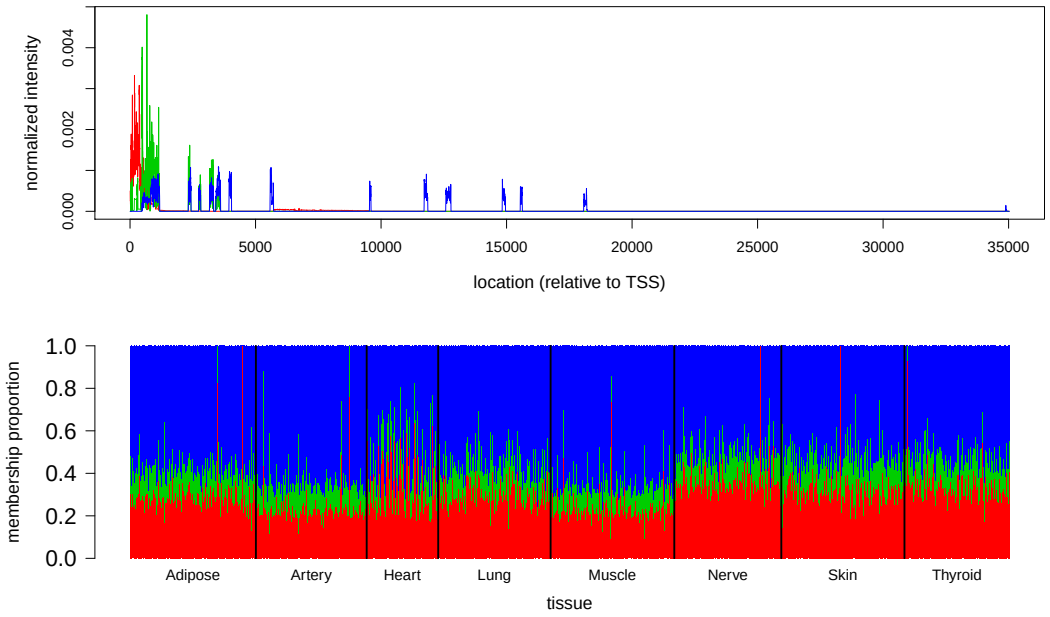


(d)

Figure A.17: Results from running *Cluster-seq* on RNA-seq data from gene *PSAP* with $K = 2, 3, 4$ and 5, corresponding to panels (a), (b), (c) and (d) respectively. Top plot in each panel shows the estimated cluster means, while bottom plot in each panel shows the corresponding admixture plot.



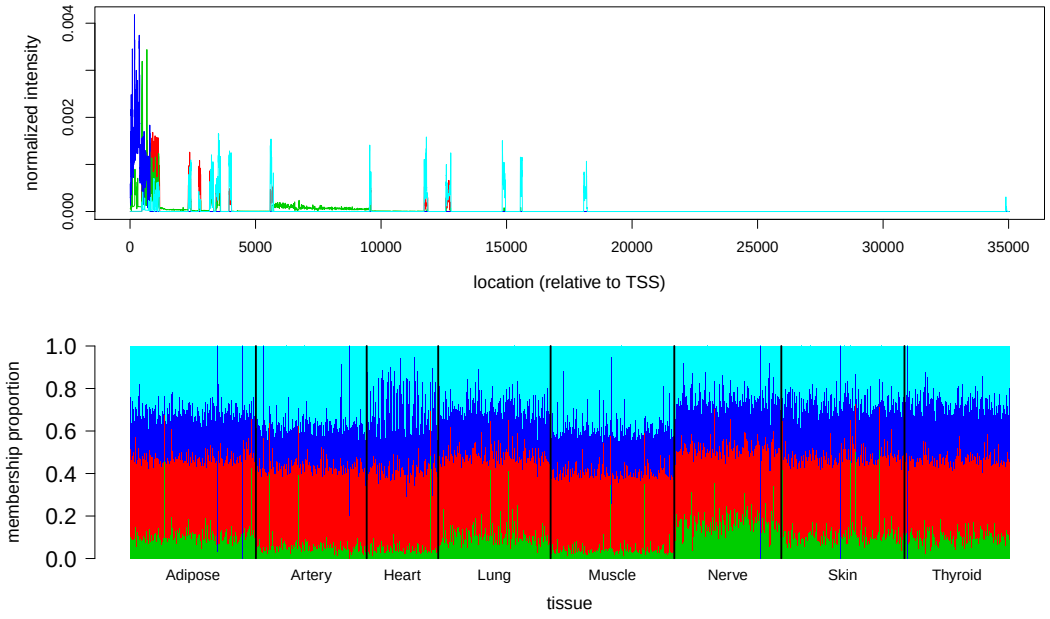
(a)



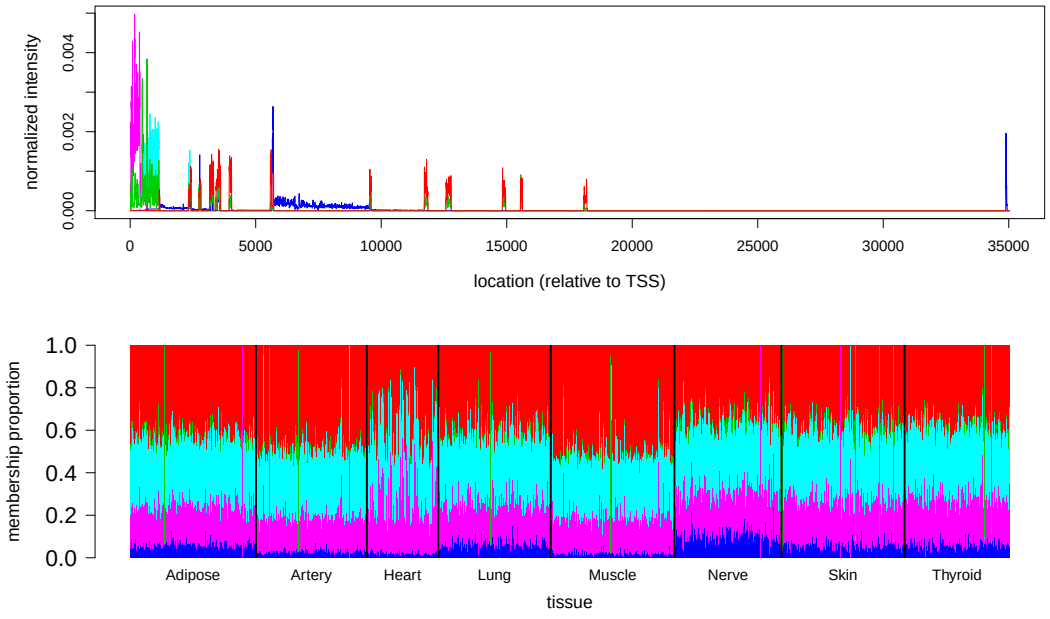
(b)

Continued on next page

– continued from previous page

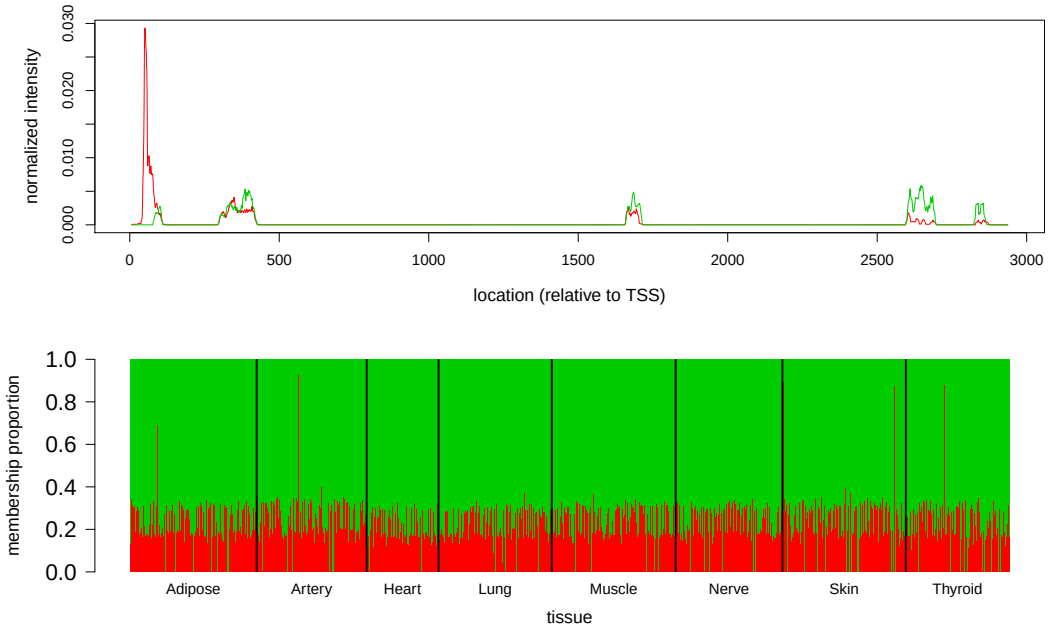


(c)

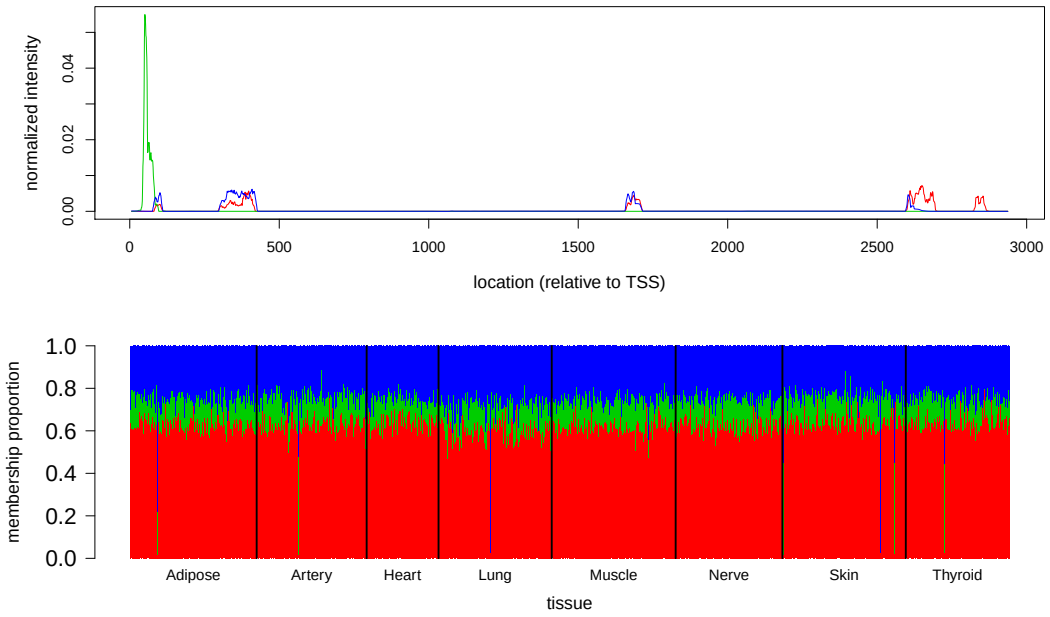


(d)

Figure A.18: Results from running *Cluster-seq* on RNA-seq data from gene *RPLP2* with $K = 2, 3, 4$ and 5, corresponding to panels (a), (b), (c) and (d) respectively. Top plot in each panel shows the estimated cluster means, while bottom plot in each panel shows the corresponding admixture plot.



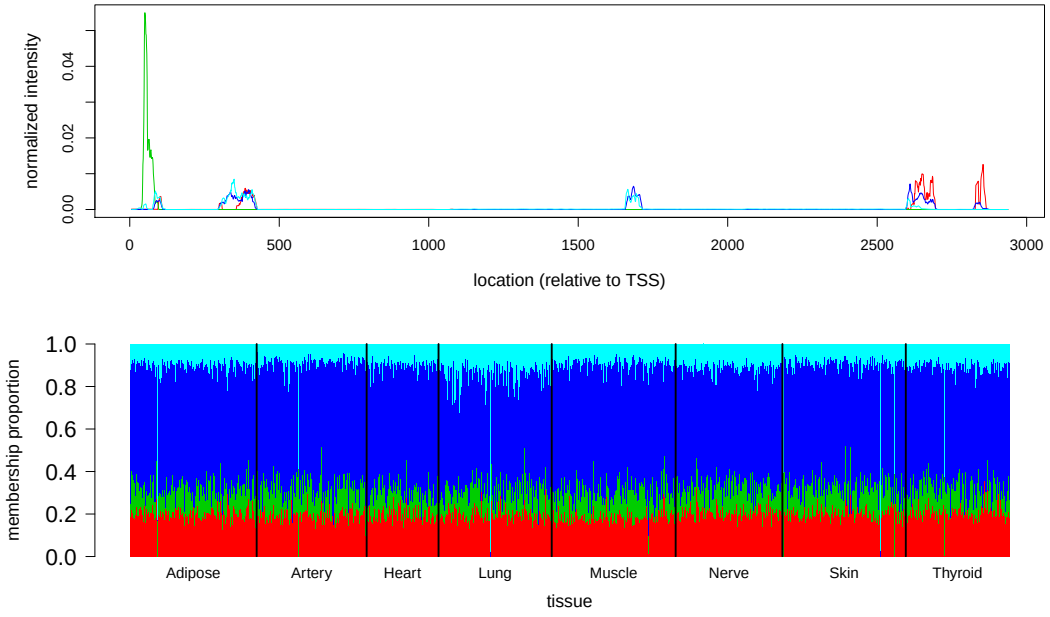
(a)



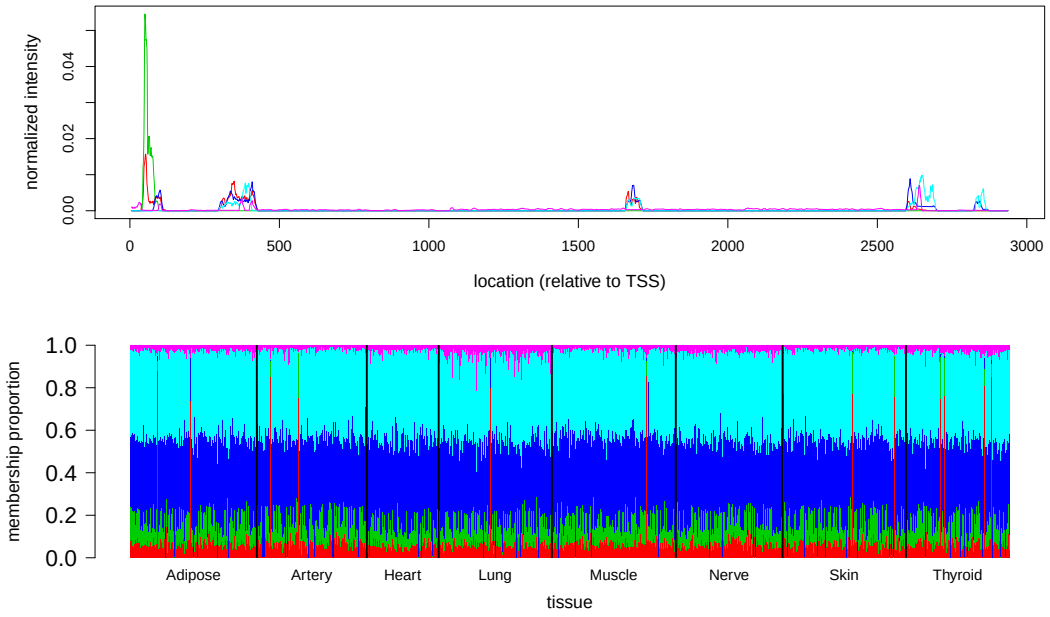
(b)

Continued on next page

– continued from previous page

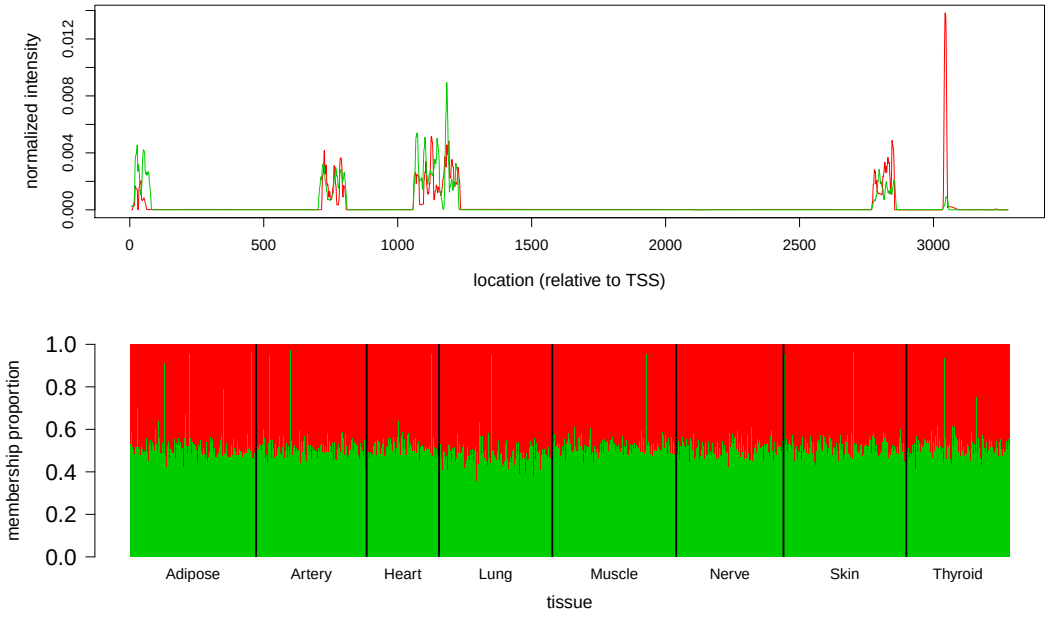


(c)

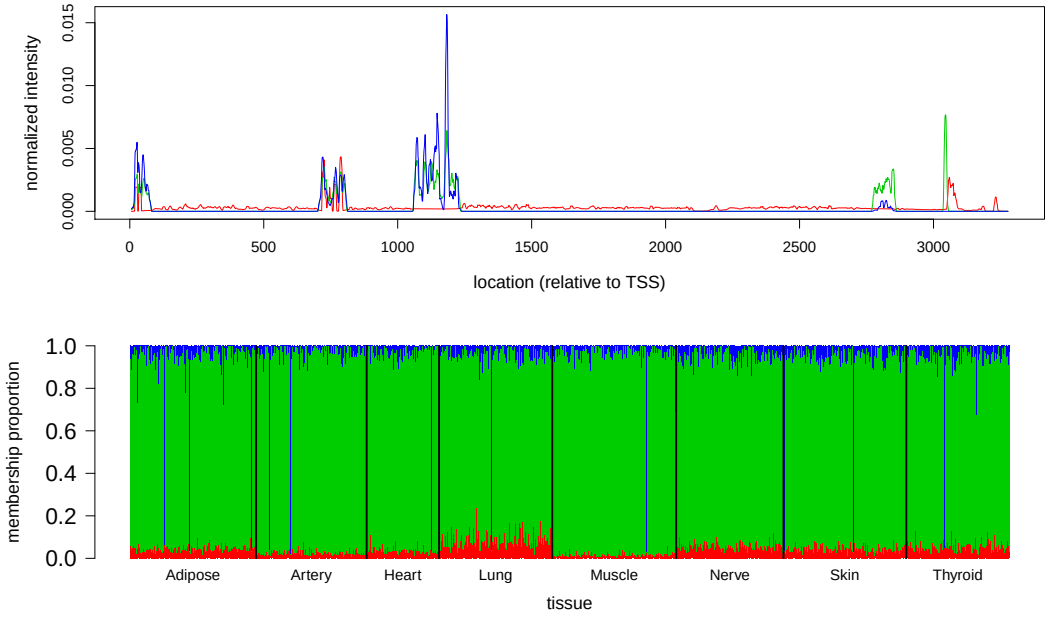


(d)

Figure A.19: Results from running *Cluster-seq* on RNA-seq data from gene *RPS13* with $K = 2, 3, 4$ and 5, corresponding to panels (a), (b), (c) and (d) respectively. Top plot in each panel shows the estimated cluster means, while bottom plot in each panel shows the corresponding admixture plot.



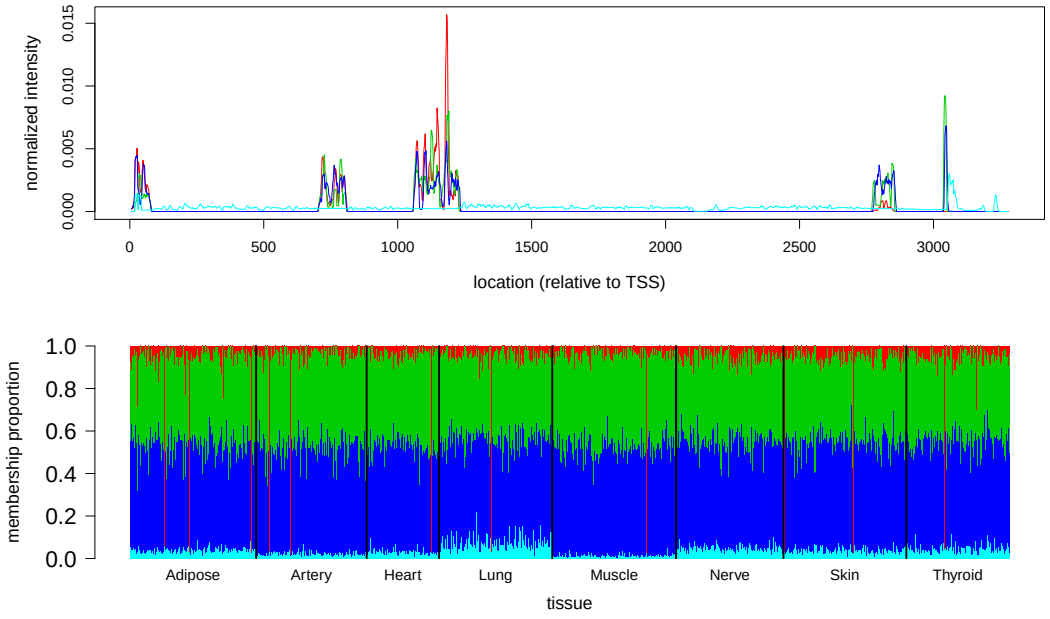
(a)



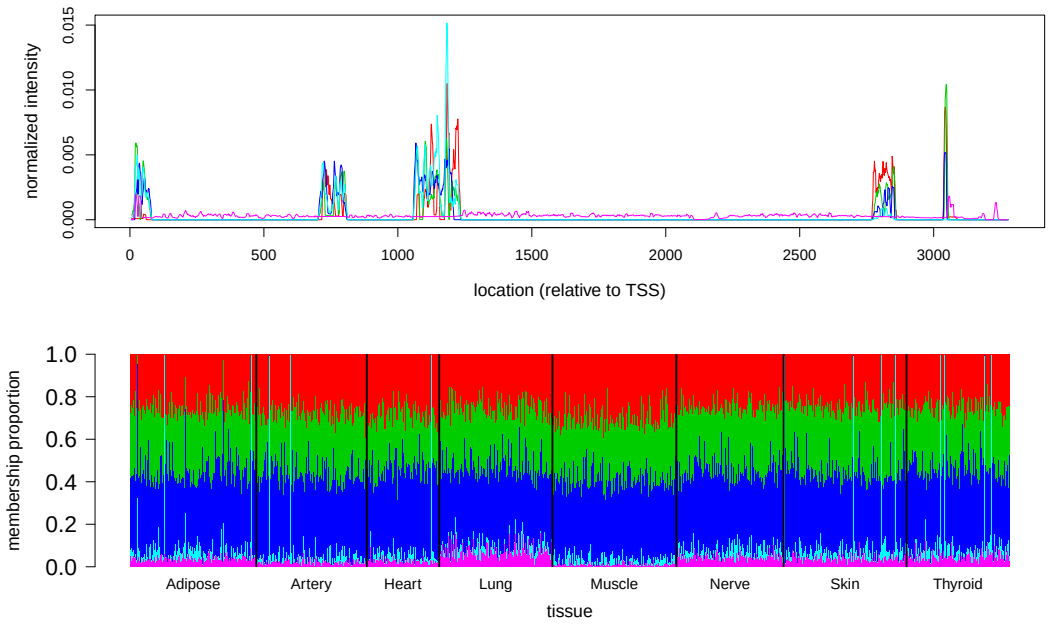
(b)

Continued on next page

– continued from previous page

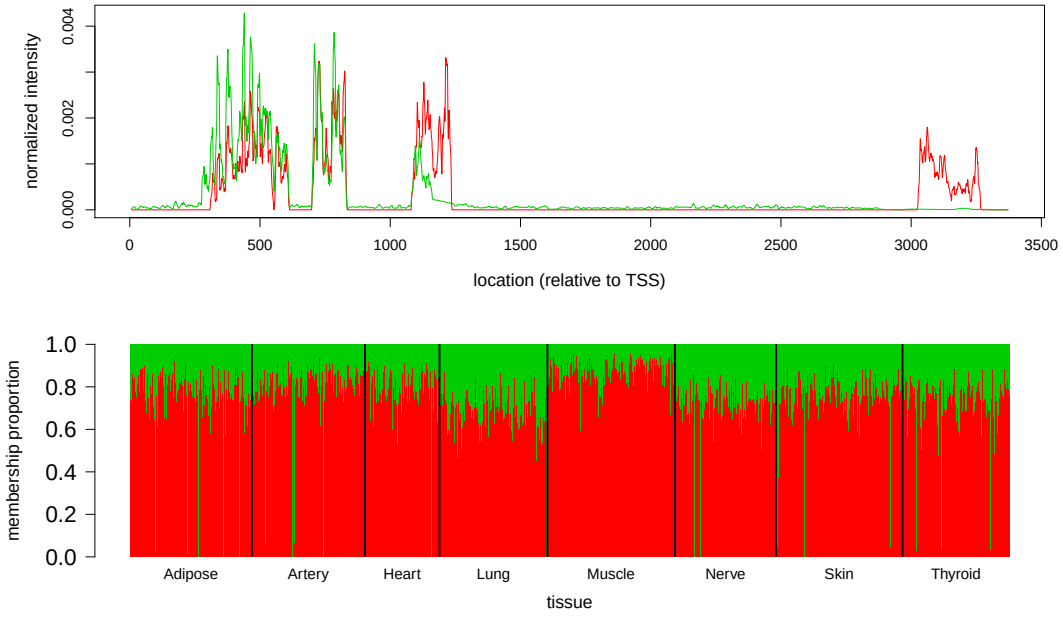


(c)

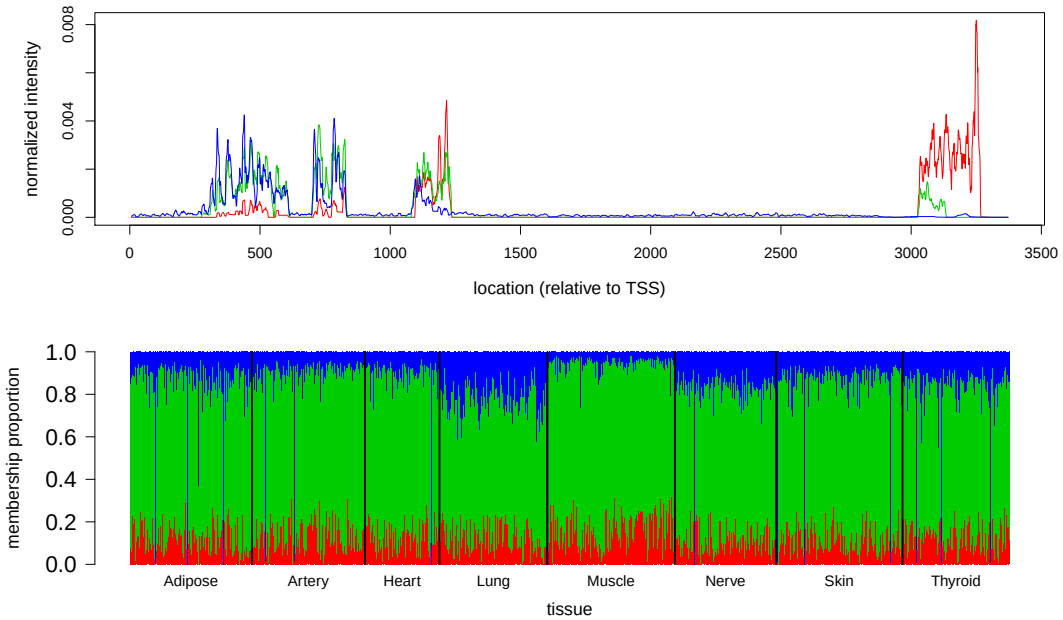


(d)

Figure A.20: Results from running *Cluster-seq* on RNA-seq data from gene *FTH1* with $K = 2, 3, 4$ and 5, corresponding to panels (a), (b), (c) and (d) respectively. Top plot in each panel shows the estimated cluster means, while bottom plot in each panel shows the corresponding admixture plot.



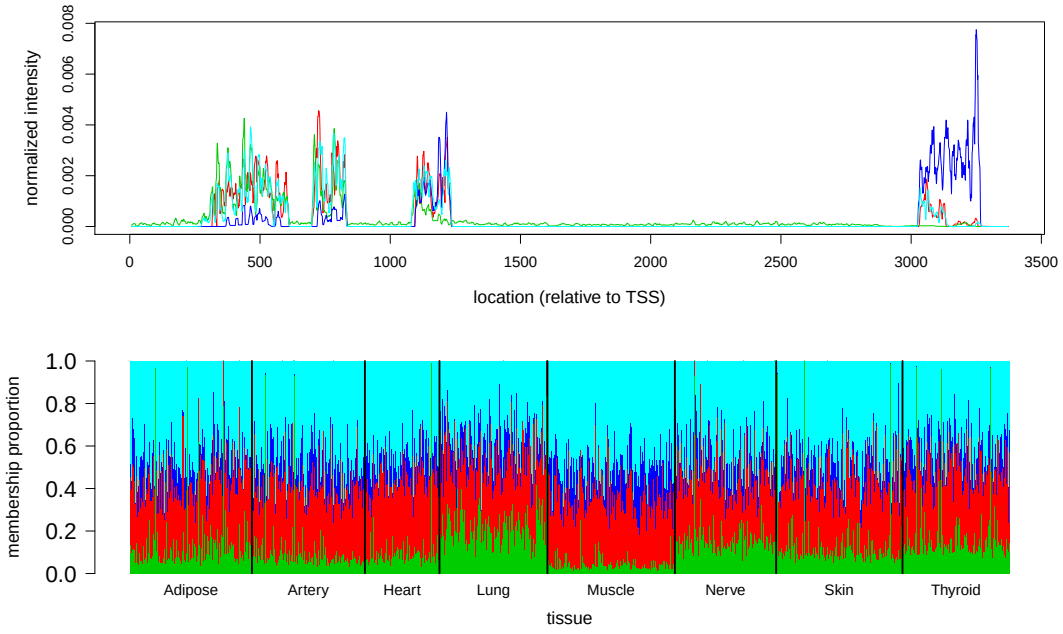
(a)



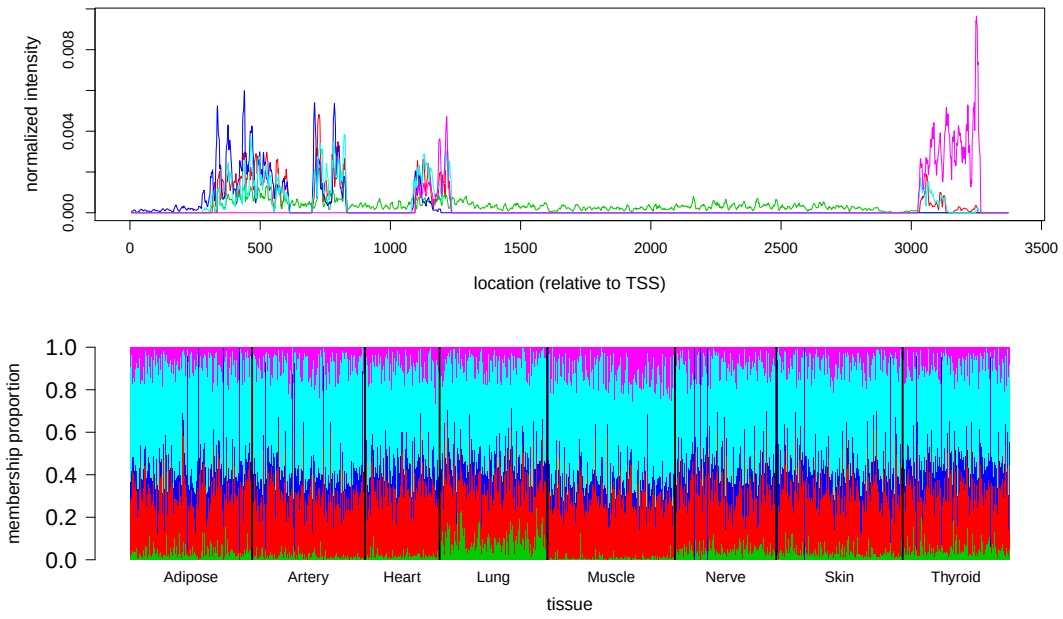
(b)

Continued on next page

– continued from previous page

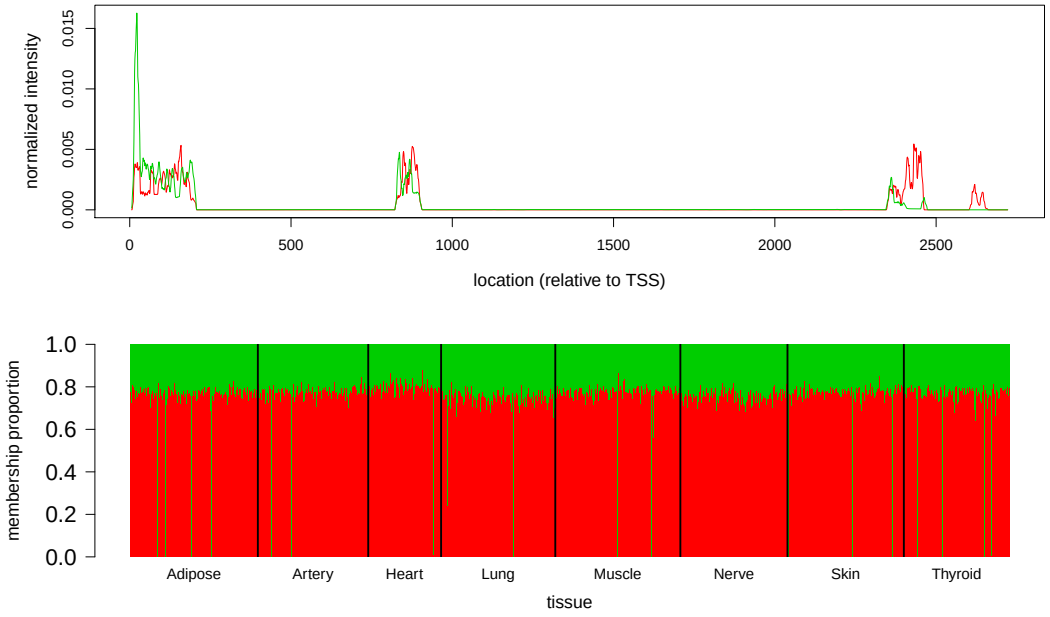


(c)

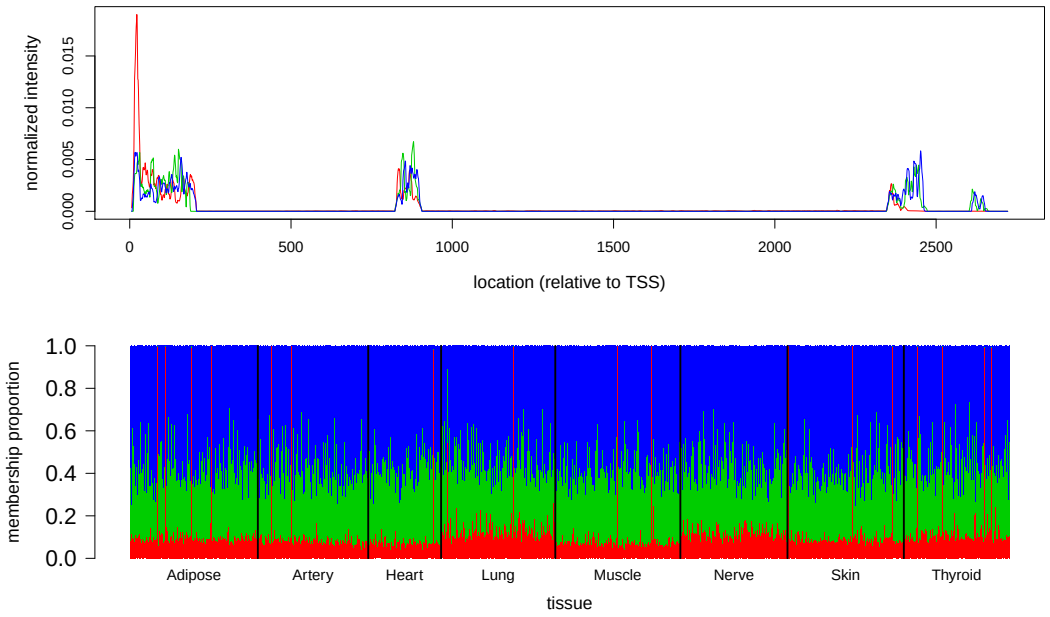


(d)

Figure A.21: Results from running *Cluster-seq* on RNA-seq data from gene *RPLP1* with $K = 2, 3, 4$ and 5, corresponding to panels (a), (b), (c) and (d) respectively. Top plot in each panel shows the estimated cluster means, while bottom plot in each panel shows the corresponding admixture plot.



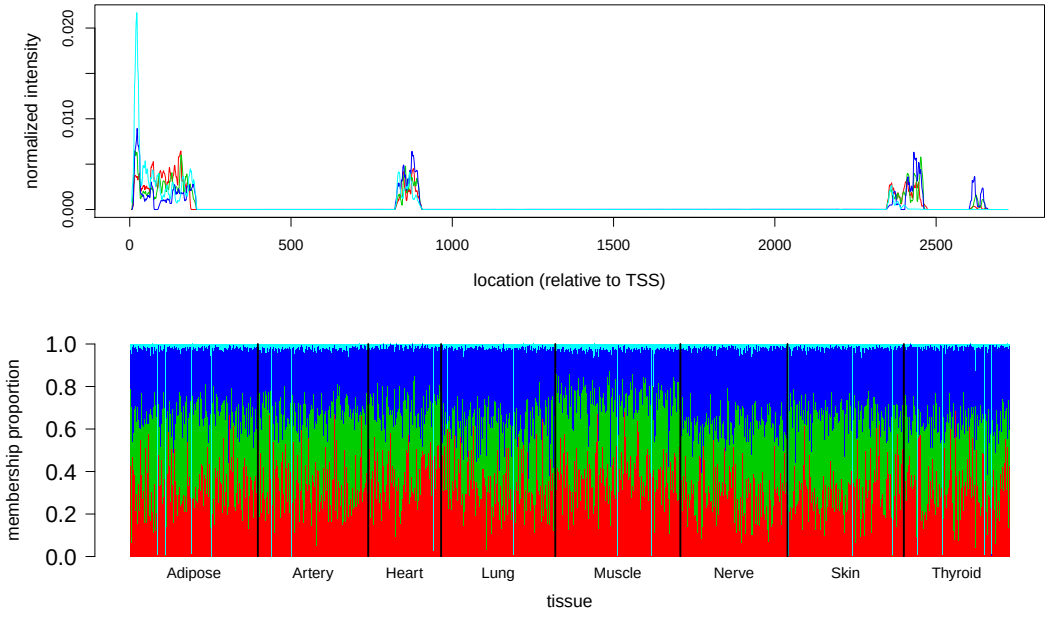
(a)



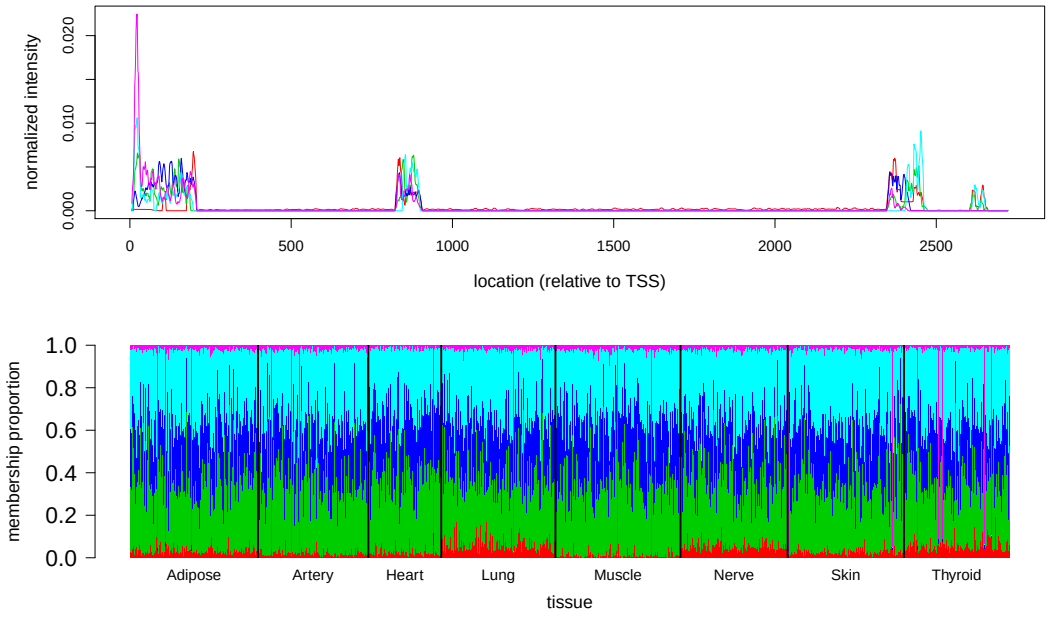
(b)

Continued on next page

– continued from previous page

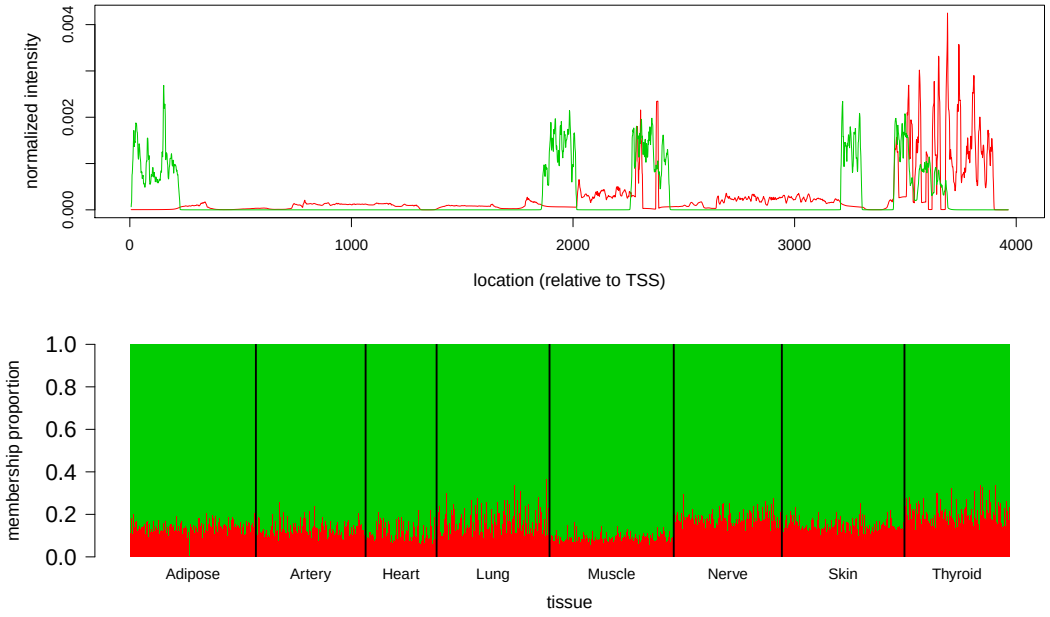


(c)

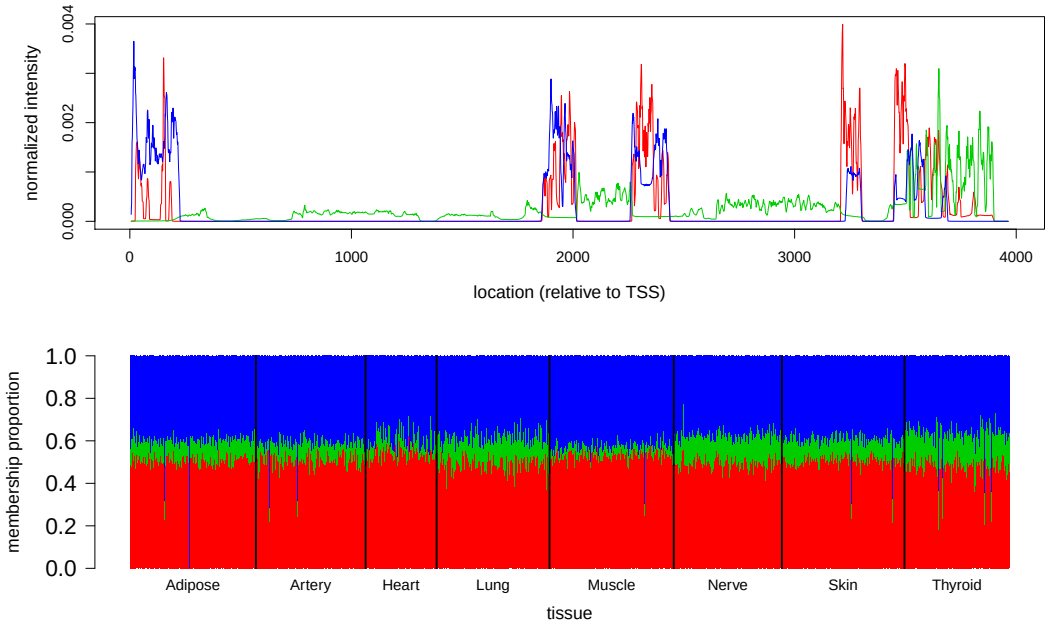


(d)

Figure A.22: Results from running *Cluster-seq* on RNA-seq data from gene *OAZ1* with $K = 2, 3, 4$ and 5, corresponding to panels (a), (b), (c) and (d) respectively. Top plot in each panel shows the estimated cluster means, while bottom plot in each panel shows the corresponding admixture plot.



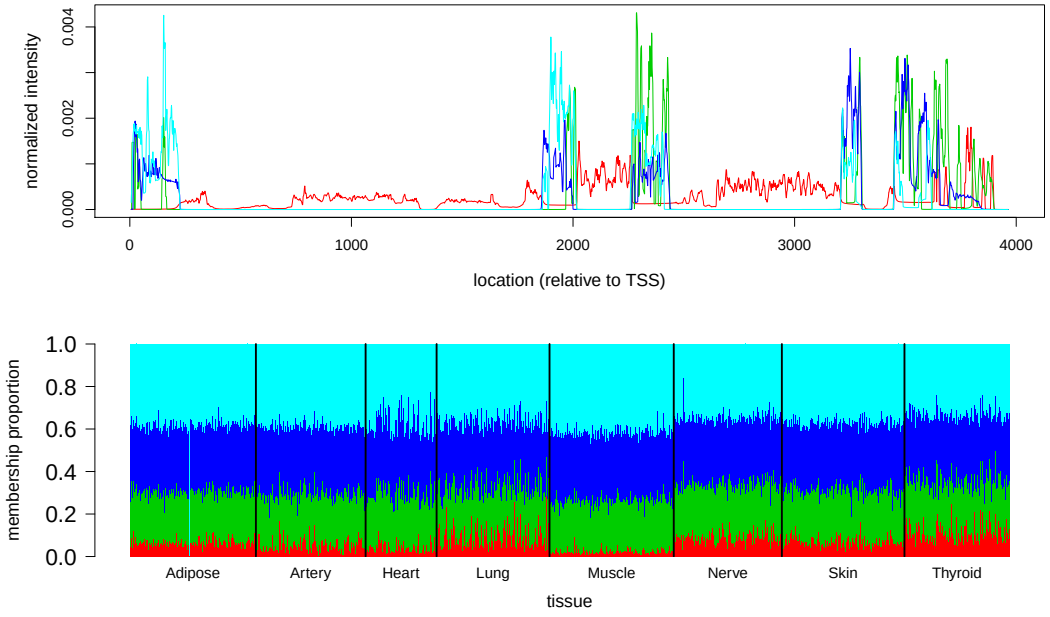
(a)



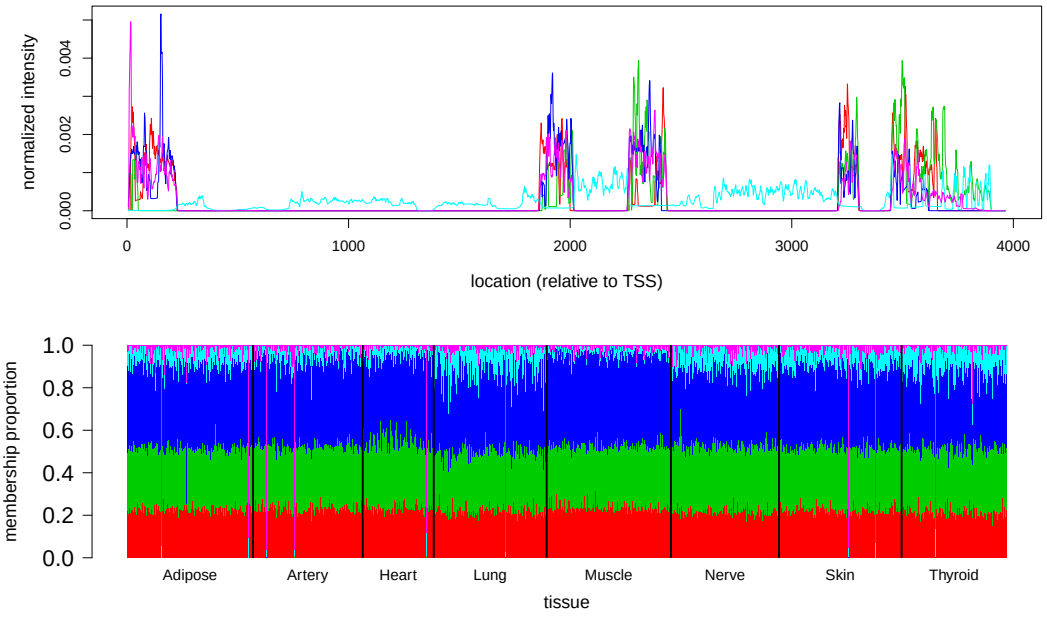
(b)

Continued on next page

– continued from previous page

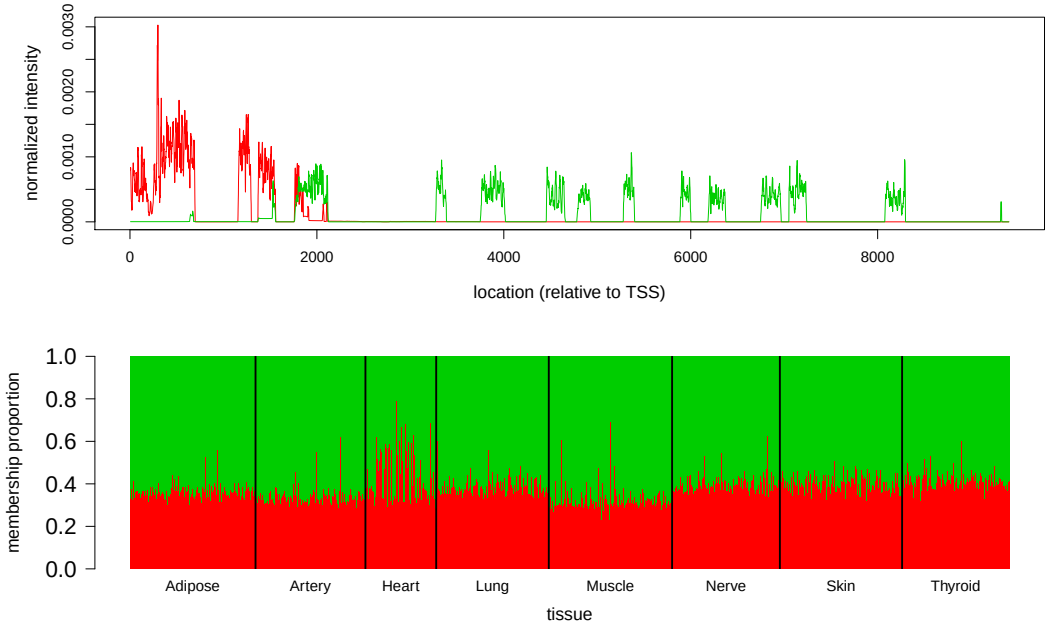


(c)

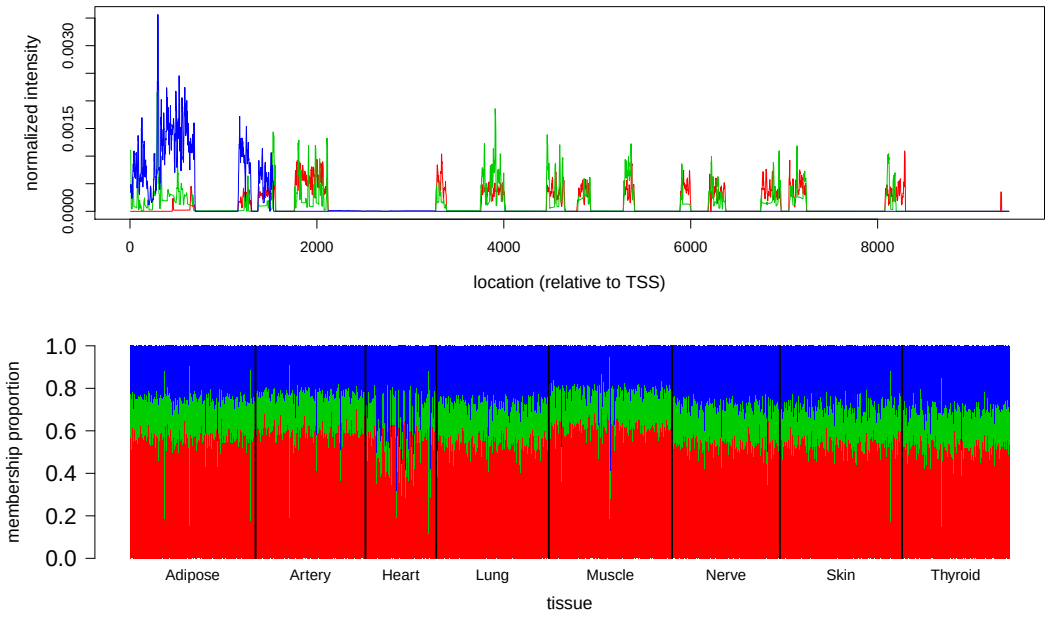


(d)

Figure A.23: Results from running *Cluster-seq* on RNA-seq data from gene *EEF2* with $K = 2, 3, 4$ and 5, corresponding to panels (a), (b), (c) and (d) respectively. Top plot in each panel shows the estimated cluster means, while bottom plot in each panel shows the corresponding admixture plot.



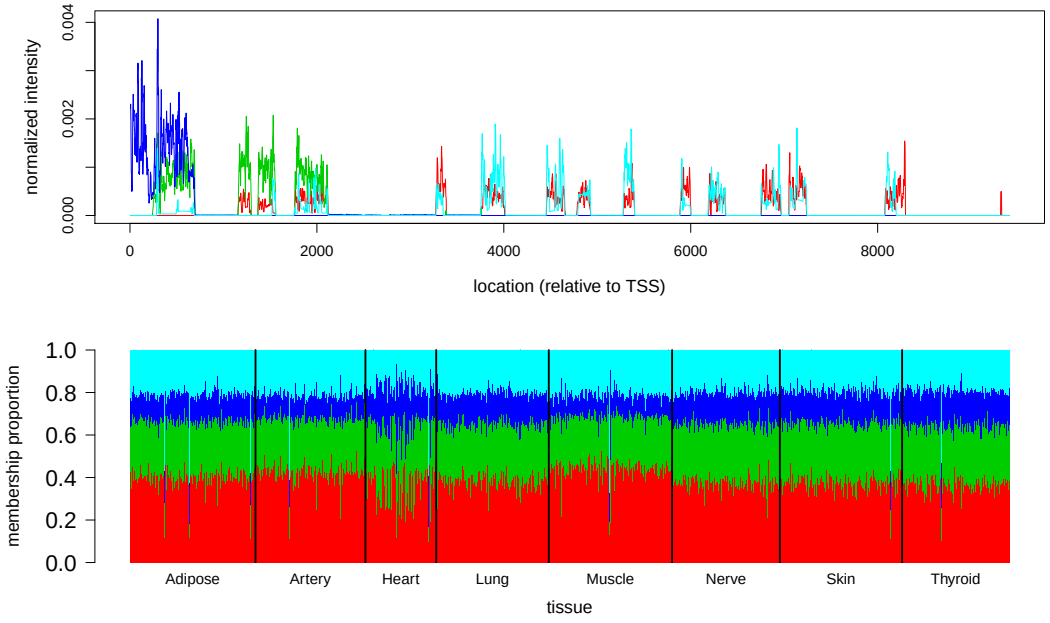
(a)



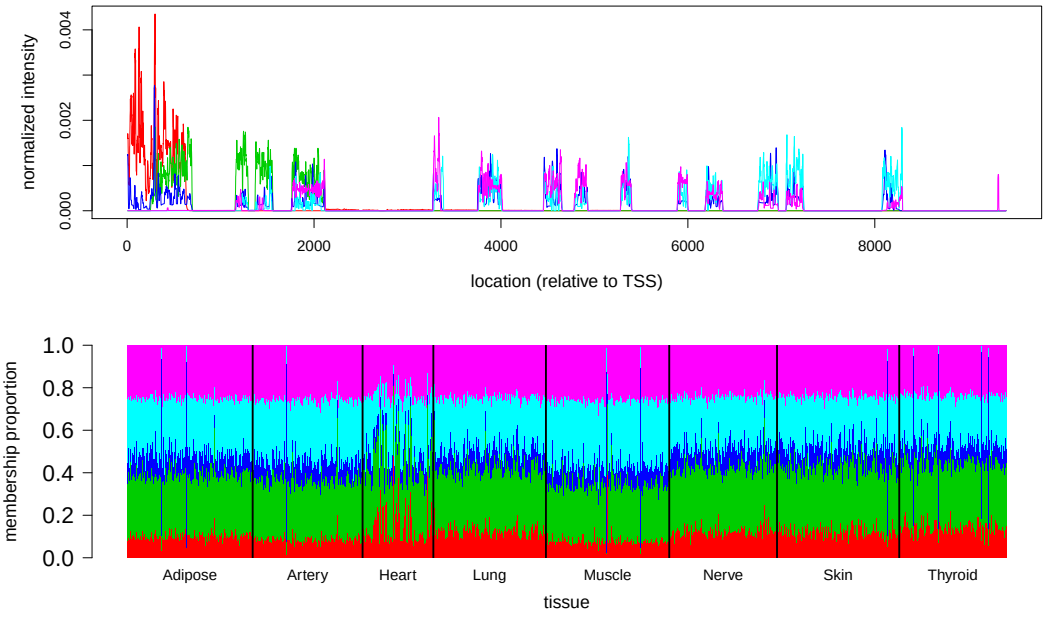
(b)

Continued on next page

– continued from previous page

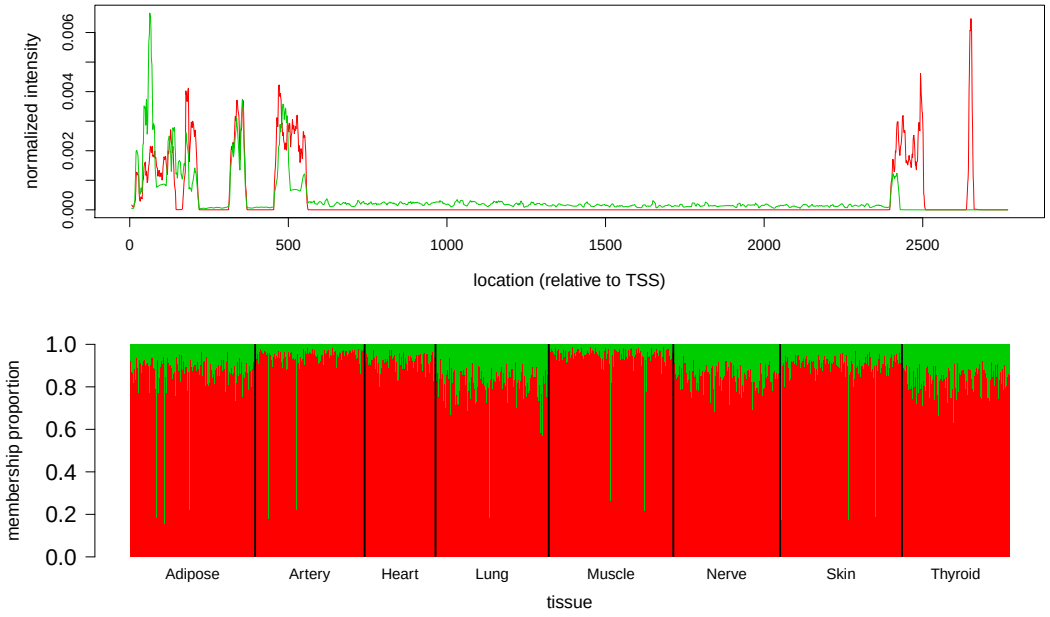


(c)

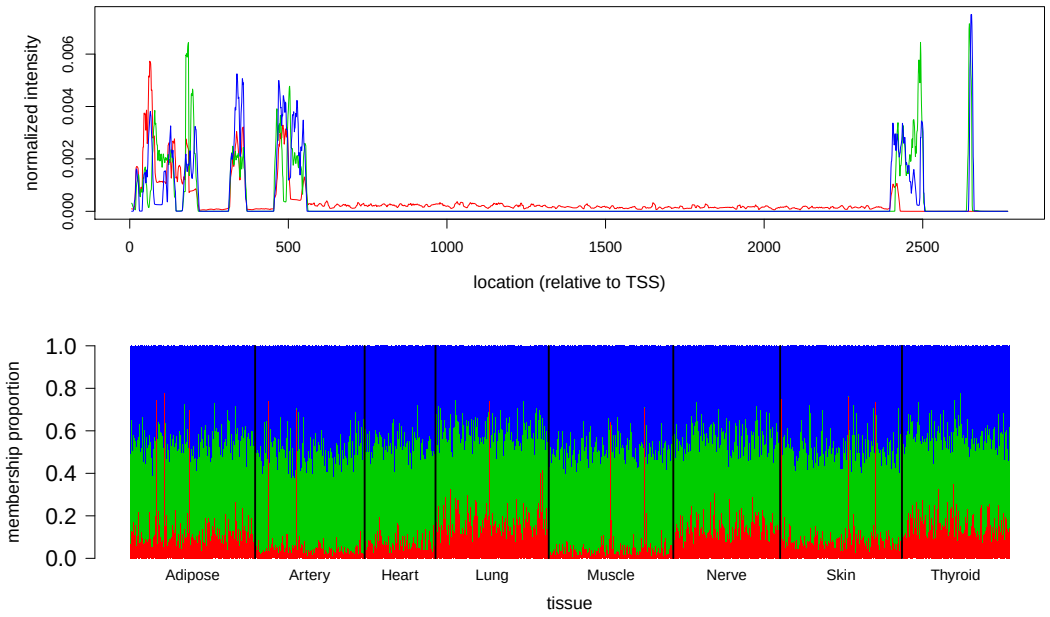


(d)

Figure A.24: Results from running *Cluster-seq* on RNA-seq data from gene *RPS16* with $K = 2, 3, 4$ and 5, corresponding to panels (a), (b), (c) and (d) respectively. Top plot in each panel shows the estimated cluster means, while bottom plot in each panel shows the corresponding admixture plot.



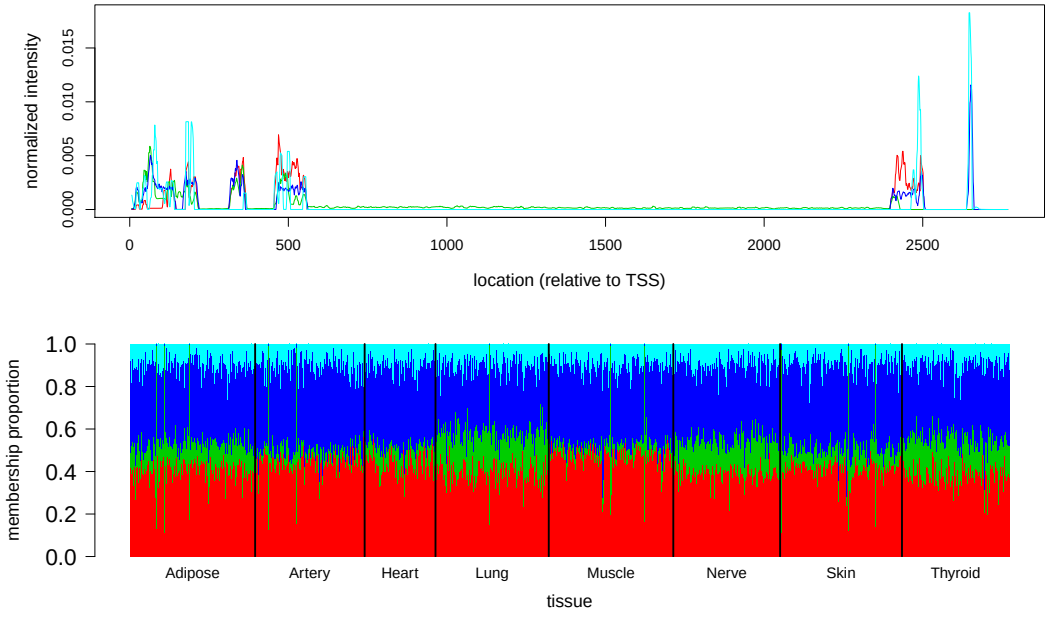
(a)



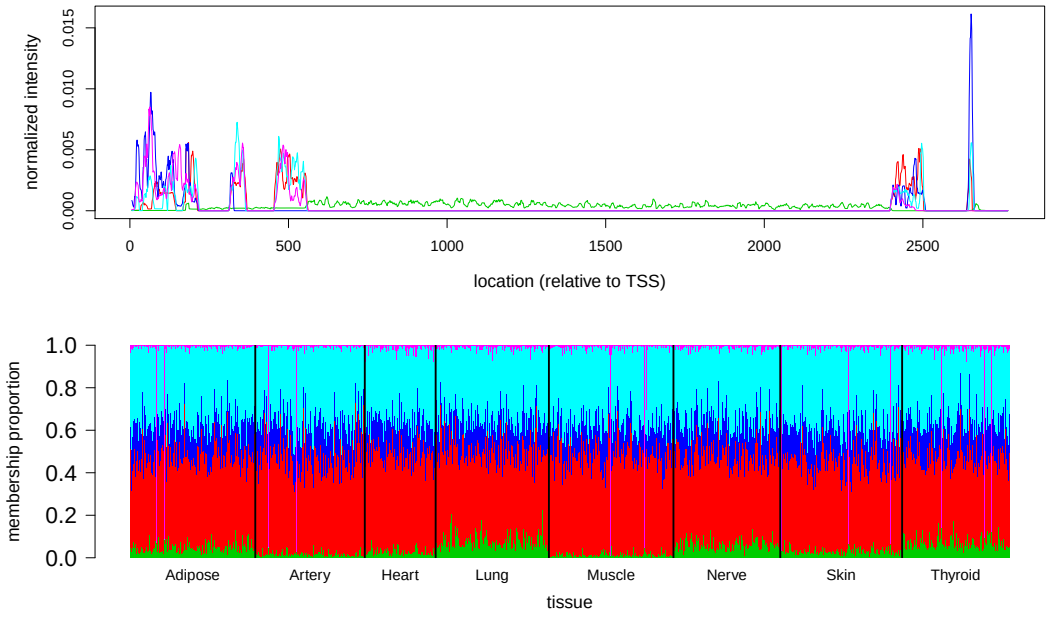
(b)

Continued on next page

– continued from previous page

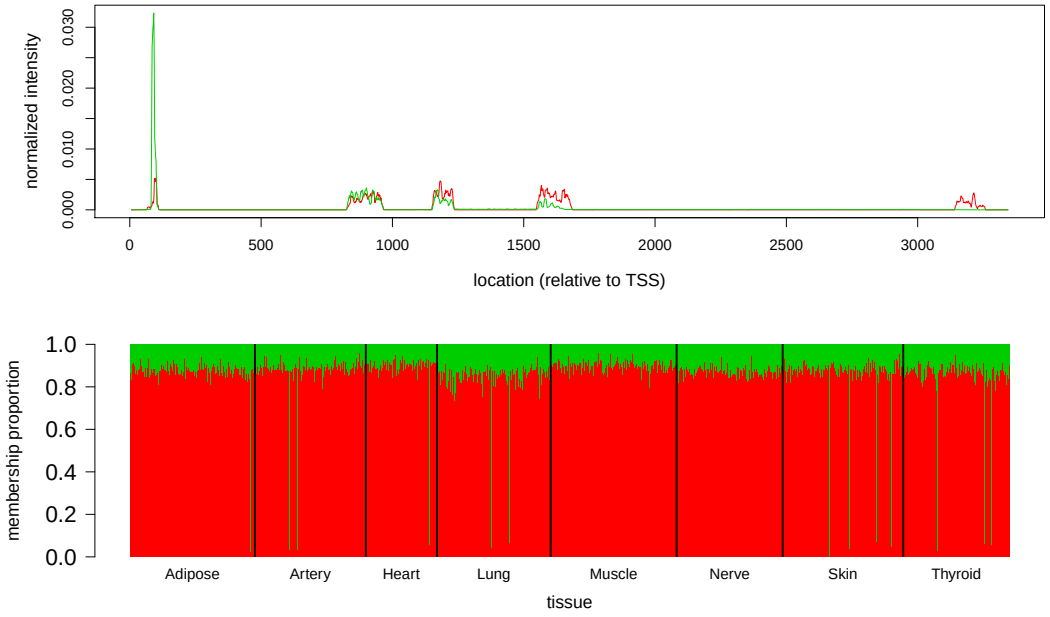


(c)

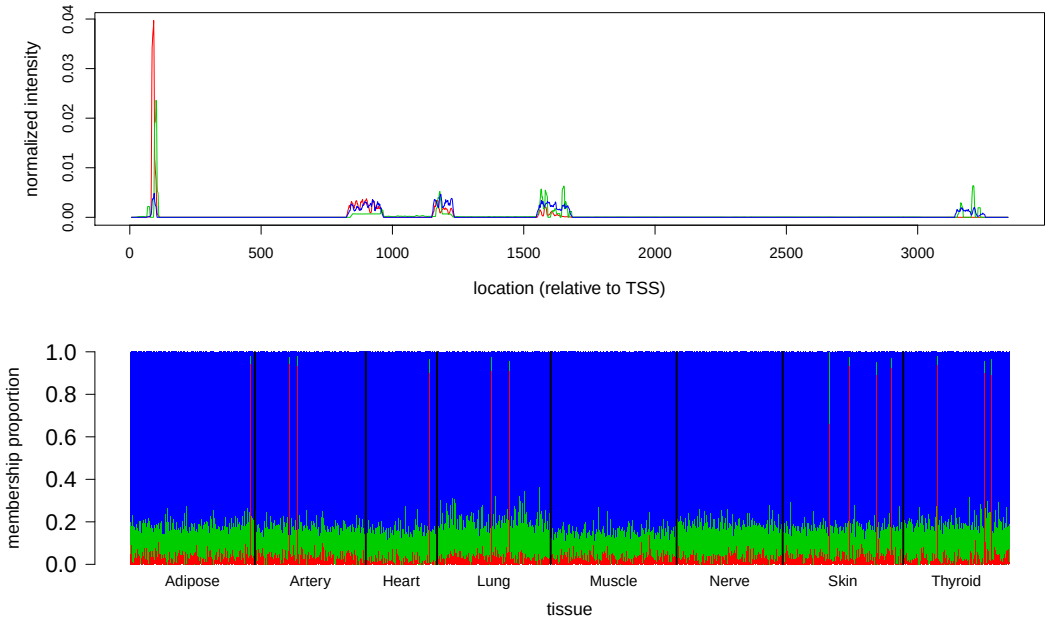


(d)

Figure A.25: Results from running *Cluster-seq* on RNA-seq data from gene *RPS11* with $K = 2, 3, 4$ and 5, corresponding to panels (a), (b), (c) and (d) respectively. Top plot in each panel shows the estimated cluster means, while bottom plot in each panel shows the corresponding admixture plot.



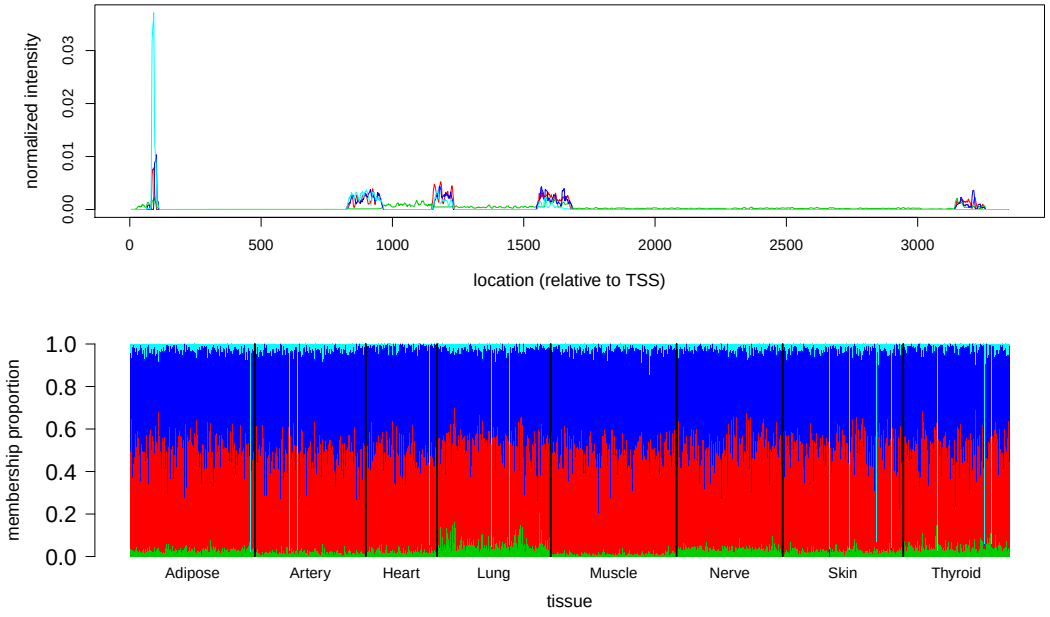
(a)



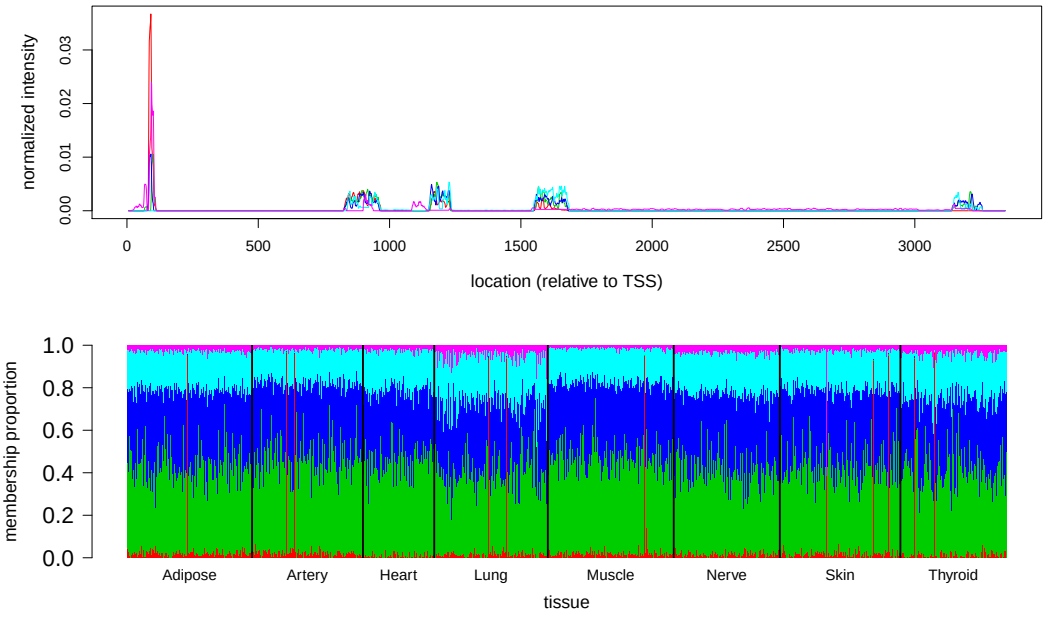
(b)

Continued on next page

– continued from previous page



(c)



(d)

A.10 Gaussian denoising with heteroskedastic errors using *smash*

A.10.1 Variance estimation for Gaussian denoising

With $\boldsymbol{Z}$ as defined in (5.7), we apply the wavelet transform W to $\boldsymbol{Z}^2$, and obtain the wavelet coefficients $\boldsymbol{\delta} = W\boldsymbol{Z}^2$. Note that $\mathbb{E}(\boldsymbol{\delta}) = (\boldsymbol{\gamma})$, where $\boldsymbol{\gamma} = W\boldsymbol{\sigma}^2$. We treat the likelihood for $\boldsymbol{\gamma}$ as if it were independent, resulting in

$$L(\boldsymbol{\gamma}|\boldsymbol{\delta}) = \prod_{j=0}^{J} \prod_{k=0}^{T-1} P(\delta_{jk}|\gamma_{jk}). \tag{A.102}$$

The likelihoods $L(\gamma_{jk}|\delta_{jk})$ are not normal, but we approximate the likelihood by a normal likelihood through matching the moments of a normal distribution to the distribution $P(\delta_{jk}|\gamma_{jk})$. That is,

$$P(\delta_{jk}|\gamma_{jk}) \approx N(\gamma_{jk}, \hat{\mathbb{V}}(\delta_{jk})) \tag{A.103}$$

so that

$$L(\gamma_{jk}|\delta_{jk}) \approx \phi(\delta_{jk}; \gamma_{jk}, \mathbb{V}(\delta_{jk})) \tag{A.104}$$

where ϕ is the normal density function, and $\mathbb{V}(\delta_{jk})$ is the variance of the empirical wavelet coefficients. Since these variances are unknown, we estimate them from the data and then proceed to treat them as known. More specifically, since $Z_t \sim$

$N(0, \sigma_t^2)$, we have that

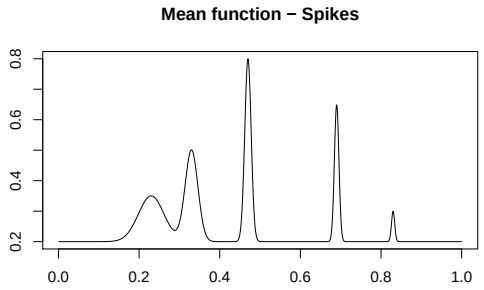
$$\begin{aligned} \mathbb{E}(Z_t^4) &\approx 3\sigma_t^4 \\ \mathbb{V}(Z_t^2) &\approx 2\sigma_t^4 \end{aligned} \tag{A.105}$$

and so we simply use $\frac{2}{3}Z_t^4$ as an unbiased estimator for $\mathbb{V}(Z_t^2)$. It then follows that $\hat{\mathbb{V}}(\delta_{jk})$ is given by $\sum_{l=1}^{T} \frac{2}{3} Z_l^4 W_{jk,l}^2$, and is unbiased for $\mathbb{V}(\delta_{jk})$. These will be the inputs to *ash*, which then produces shrunk estimates in the form of posterior means for the corresponding parameters. Although this works well in most cases, there are variance functions for which the above procedure tends to overshrink the wavelet coefficients at the finer levels. This is likely because the distribution of the wavelet coefficients is extremely skewed, especially when the true coefficients are small (at coarser levels the distributions are much less skewed since we are dealing a linear combination of a large number of data points). One way around this issue is to employ a procedure that jointly shrinks the coefficients $\boldsymbol{\gamma}$ and their variance estimates (implemented in the `jash` option in our software). The final estimate of the variance function is obtained from the posterior means via the average basis inverse across all the shifts.

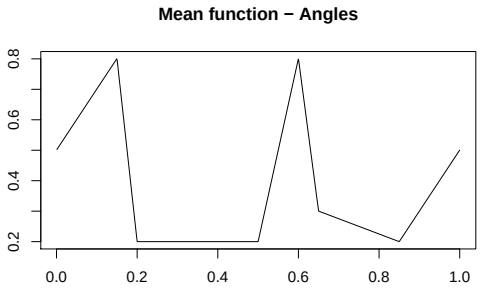
A.10.2 Simulation results

As noted in Section 5.2, we performed an extensive simulation study for Gaussian errors using a variety of test functions (7 mean functions and 5 variance functions, including constant variance), signal-to-noise ratios (SNR = 1 and 3) and sample sizes

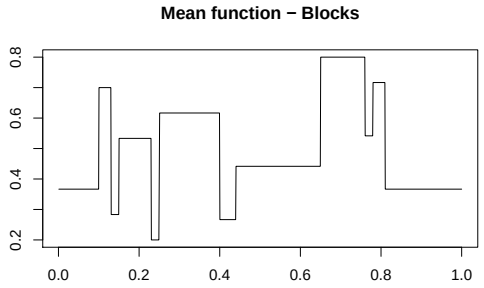
($T = 256, 512, 1024$). For details of where the results are saved and what information they contain, refer to README.md from the ashwave repo (https://github.com/stephenslab/ashwave). In particular, we include an RShiny app as an interactive way to present the results for $T = 1024$, the code for which can be found in the dscr-smash repo (https://github.com/zrxing/dscr-smash). For more detailed descriptions of what the repository does, instructions on reproducing the results in Section 5.2, as well as guidelines for extending the simulation study under the DSC framework refer to the README.md file from the repo. In particular, users can view the boxplots of MISEs for the methods and test functions they are interested in by checking the appropriate boxes. For more details on the RShiny display refer to graphs.Rmd in the dscr-smash repo. For reference, the mean and variance functions used in the simulation studies are shown below in Figures A.26 and A.27.



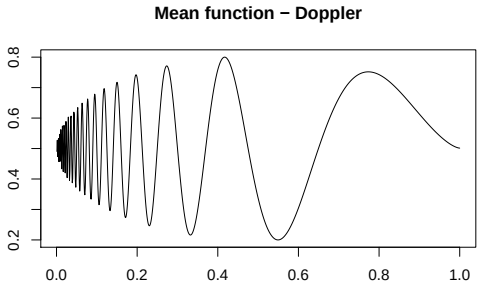
(a)



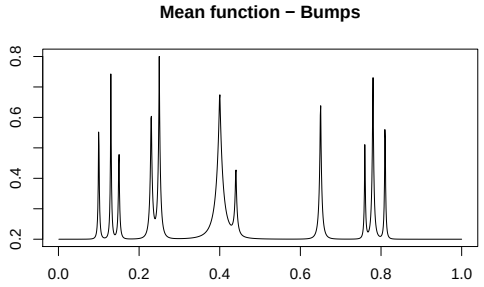
(b)



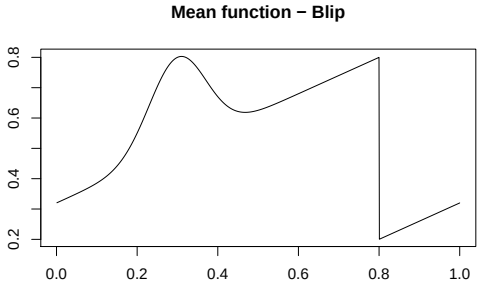
(c)



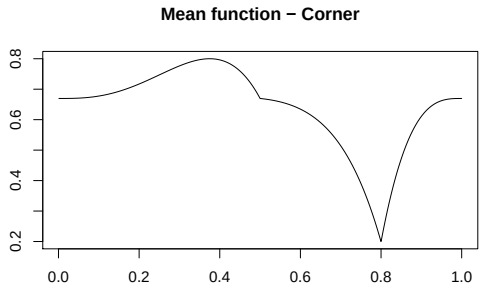
(d)



(e)

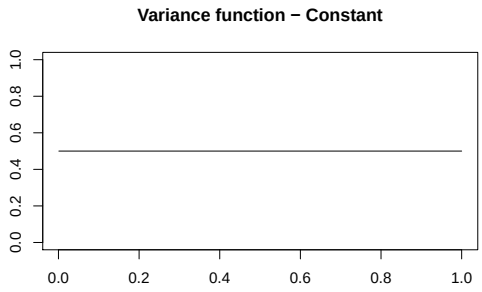


(f)

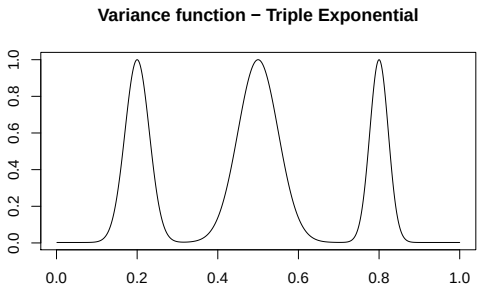


(g)

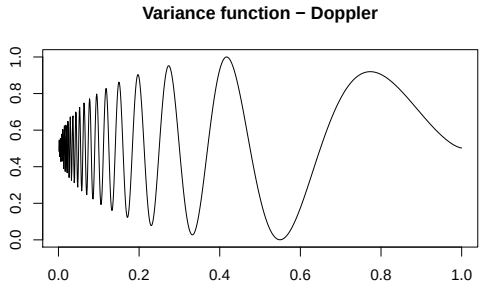
Figure A.26: The seven mean functions used in the Gaussian simulations, all scaled to be between 0.2 and 0.8.



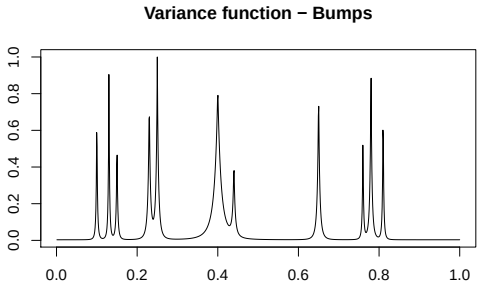
(a)



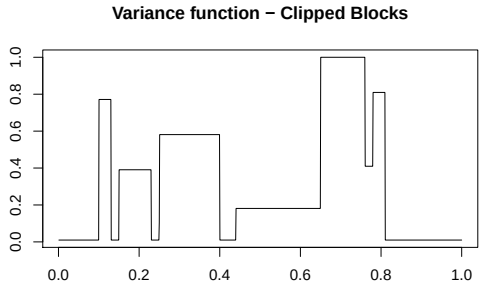
(b)



(c)



(d)



(e)

Figure A.27: The five variance functions used in the Gaussian simulations, which are rescaled in the simulations to achieve the desired signal to noise ratios.

In addition to the performance of the various methods, we also provide code in the ashwave repo (https://github.com/stephenslab/ashwave) generating a table that provide additional details for each method used, including the associated software, variance assumption, wavelet basis, shrinkage procedure used, and other relevant information.

A.11 GitHub repositories

- Results and code for *smash* simulations and figures/tables in Xing and Stephens (2016) - https://github.com/stephenslab/smash-paper
- Dynamical Statistical Comparison framework for conducting simulations for the Gaussian version of *smash* - https://github.com/zrxing/dscr-smash
- R package for *smash* - https://github.com/stephenslab/smashr
- R package for *Multiseq* - https://github.com/stephenslab/multiseq
- Using *Multiseq* to infer TF binding, and comparison of *Multiseq*-based inference with CENTIPEDE and msCentipede - https://github.com/zrxing/multiseq_analysis
- R code, simulation results and analysis results for *Cluster-seq* - https://github.com/zrxing/sequence_clustering