The analysis of RNA-Seq experiments using approximate likelihood

Daniel C. Jones

A dissertation submitted in partial
fulfillment of the requirements for the
degree of
Doctor of Philosophy
University of Washington
2020

Reading Committee:
Walter L. Ruzzo, Chair
Sreeram Kannan
Georg Seelig

Program Authorized to Offer Degree:
Paul G. Allen School of Computer Science & Engineering

University of Washington

Abstract

The analysis of RNA-Seq experiments using approximate likelihood
Daniel C. Jones
Chair of the Supervisory Committee:
Walter L. Ruzzo
Paul G. Allen School of Computer Science & Engineering

The analysis of mRNA transcript abundance with RNA-Seq is a central tool in molecular biology research, but often analyses fail to account for the uncertainty in these estimates, which can be significant, especially when trying to disentangle isoforms or duplicated genes. Preserving uncertainty necessitates a full probabilistic model of the all the sequencing reads, which quickly becomes intractable, as experiments can consist of billions of reads. To overcome these limitations, we propose a new method of approximating the likelihood function of a sparse mixture model, using a technique we call the Pólya tree transformation. We demonstrate that substituting this approximation for the real thing achieves most of the benefits with a fraction of the computational costs, leading to more accurate detection of differential transcript expression.

For Maddie. Let's have some more normal ones.

Contents

Index of mathematical notation

Likelihood Notation

$\mathbb{R}_+$	Positive real numbers
Δ^n	n-dimensional unit simplex, i.e., $\{x \in \mathbb{R}_+^{n+1} \mid \sum_{i=1}^{n+1} x_i = 1\}$
r	Set of reads (representation is left unspecified)
$r^{(i)}$	Set of reads belonging to the ith sample
x	Relative expression vectors
$x^{(i)}$	Relative expression vector (in Δ^{n-1}) for the ith sample.
z_{ij}	Latent indicator variables that is 1 iff read i was generated from transcript j
θ	Unspecified model parameters
m	Number of RNA-seq reads in a particular sample.
n	Number of annotation transcripts.
$p_j(\cdot)$	Transcript j's probability distribution over reads
$\mathcal{P}$	Normalized RNA-Seq likelihood function (equivalent to a posterior over expression u
ℓ_i	Length of fragment i
L_j	Length of transcript j
$\mathbb{L}_j$	Effective length of transcript j

Tree Notation

$\text{left}(i)$	Index of the left child of the node at index i
$\text{right}(i)$	Index of the left child of the node at index i
$\text{leaves}(i)$	Set of indexes of leaf nodes in the subtree rooted at i
$\text{internal}(i)$	Set of indexes of internal (i.e., non-leaf) nodes in the subtree rooted at i
$\mathcal{L}$	Vector giving the number of internal nodes in each internal node's left subtree
$\mathcal{R}$	Vector giving the number of internal nodes in each internal node's right subtree

Overview

This dissertation is organized into four chapters.

1. **The analysis of RNA-Seq experiments.** We give a brief overview of methodologies that have been brought to bear to analyze gene and transcript expression in RNA-Seq experiments over the last twelve years. The dominant probabilistic view of RNA-Seq treats it as a sparse mixture model, with fixed components, where relative mRNA abundance is represented by the mixture coefficients.

2. **Approximating the RNA-seq likelihood function.** Here we develop a new approximation to the RNA-Seq likelihood function. Though the focus will be on RNA-Seq, this approach is generalizable to other sparse mixture models. The approximation is based on a fully generalized family of stick breaking transformations we call Pólya tree transformations. We prove some useful properties of this class of transformations, and then go on to show the approximation is an extremely accurate despite using only a constant number of parameters per transcript.

3. **Applications of approximate likelihood.** Several applications of approximate likelihood are presented. Most importantly, we build a probabilistic model of differential expression, and show it improves on existing state of the art methods. We also examine simple pairwise correlation and classification tasks where approximate likelihood shows improvements over using point estimates or bootstrap sampling.

4. **Scaling up approximate likelihood analysis.** Finally, with the ultimate goal of scaling up inference to large numbers of cells in isoform level single cell datasets, we describe two possible improvements. First, we show an alternative heuristic for building Pólya tree transformations that allows for one size fits all transformations. Second, we explore the idea of quasi-conjugacy that allows efficient sampling from the posterior in models using a Dirichlet prior.

Some of the material preseted here previously appeared in the preprint:

> Jones, Daniel C., and Walter L. Ruzzo. "Polee: RNA-Seq analysis using approximate likelihood." bioRxiv (2020).

1

The analysis of RNA-seq experiments

1.1 Introduction

The increasing ability over the past two decades to perform large scale measurement of mRNA abundance has transformed how molecular biology research is done. Prior to the introduction of complementary DNA microarrays in the mid-1990s (Schena et al., 1995), studying the dynamics of gene expression was limited to overall RNA quantity or to a few specific RNA at a time. With this revolutionary expansion in the scope, microarrays presented new statistical challenges. Multiple-testing, noise, normalization, cross-hybridization, and reconciling multiple vendor's arrays were all issues tackled in the ensuing years by new software and statistical methods.

A similar technological leap was made a decade later with the advent of high-throughput RNA sequencing, or "RNA-seq" (Wang, Gerstein, and Snyder, 2009). The new technique, enabled in particular by new Illumina's new sequencers capable of sequencing millions of short (initially, 30-35nt) fragments at a time, promised more sensitivity, accuracy, and flexibility than microarrays. RNA-seq offers expression information at the nucleotide level without being chained to a particular set of probes or annotated genes. When new information becomes available (a better genome assembly or more accurate gene annotations, for example), RNA-seq data can be trivially reanalyzed.

As was the case with microarrays, the promise of higher fidelity transcriptomics came with a set of new computational and statistical challenges. Most of these challenges stem from one limitation: RNA-seq reads are short. Current sequencing technologies are capable of generating long reads with limited throughput (e.g., nanopore sequencing), or short reads with high throughput (e.g., sequencing by synthesis). When the goal is accurate quantification, as is usually the case in RNA-seq experiments, the latter is preferable. Though the reads have grown longer in the ensuing decade (from 30-35nt to 100-200nt on Illumina's platform), RNA-seq for the most part does not directly quantify full RNA molecules. The median human RNA transcript length is 794nt (Ensembl, 2018), much longer than the typical read.

The overcome this limitation, RNA molecules are fragmented (usually randomly, using sonication), and the ends of these fragments are sequenced with short reads. This approach complicates analysis because it introduces ambiguity. Typically transcripts have unique sequences, but they may differ from a paralog or another isoform of the same gene by only a few bases. When this is the case, a sequenced read may be compatible with multiple

transcripts, with little or no information to distinguish which it originated from. This moves the statistical inference problem from the realm of counting to that of *deconvolution*. Ambiguous reads must be disambiguated, mixed signals must be unmixed.

Statistical analysis of RNA-seq must first and foremost grapple with this deconvolution problem, but exactly how is not obvious and has led to a diversity of approaches with varying degrees of success.

1.2 The RNA-Seq quantification problem

There have been a large number of methods developed to analyze RNA-seq data in the last decade. This section will attempt to organize notable approaches to quantification into a simple taxonomy, explain the strengths and weaknesses, and describe where the method developed here fits in. RNA quantification is not the only application of RNA-seq. Most prominently, the technology has been used to discover and annotate new transcripts, either by de novo assembly (Robertson et al., 2010; Grabherr et al., 2011; Haas et al., 2013), or processing reads aligned to a reference genome sequence (Trapnell et al., 2010; Guttman et al., 2010; Pertea et al., 2015). But also other applications include detection of fusion transcripts (Kumar et al., 2016) and RNA editing (Peng et al., 2012). We will largely ignore these uses of RNA-seq to focus squarely on quantification. For the most part, we will assume that a suitable reference genome sequence and transcript annotations are available, but will note methods that do not have this restriction.

The first broad distinction among quantification methods is between those that avoid deconvolution and those that embrace it. A fundamental assumption of most RNA-seq analyses is that transcript expression is proportional in expectation to the number of reads observed from that transcript [1]. As Mortazavi et al. (2008) notes, read counts were observed to be "linear across a dynamic range of five orders of magnitude in RNA concentration." Yet reads are often of ambiguous origin and cannot be trivially assigned to transcripts. This is particularly the case for reads that overlap multiple isoforms of the same gene. Estimating transcript expression thus necessitates either explicitly assigning these ambiguous reads to transcripts, or otherwise implicitly considering the space of possible assignments.

But transcript expression is not the only possible quantity of interest, and many methods have found success in posing an alternative inference problem that avoids dealing with read ambiguity altogether. These methods are often simpler and more efficient than deconvolution methods and so for good reason remain a popular approach to analyze RNA-seq experiments.

1 We discuss what exactly "proportional" means, and some complications to this assumption in later sections.

1.2.1 Methods that avoid deconvolution

The problem of deconvolution, or read disambiguation, can be avoided if we restrict ourselves only to unambiguous reads or if we consider only transcriptomic features like genes or splice junctions that are potentially distinct enough as to avoid serious issues with ambiguous reads.

Two early methods that adopt this approach are edgeR (Robinson, McCarthy, and Smyth, 2010) and DESeq (Anders and Huber, 2010). Both methods presuppose gene-level read counts can be produced unambiguously from RNA-seq reads. In transcriptomics the precise definition of a "gene" is elusive, but it is commonly used to describe a set of transcripts whose genomic positions overlap significantly. For example, the popular Ensembl annotations define a genes largely by clustering annotated transcripts (Ensembl, 2018). With this definition, most reads ought to be able to be assigned to genes with ease. Some will remain ambiguous due to repeats and paralogs, but discarding these is hardly catastrophic.

All methods that follow in the count-based lineage proceed from the assumption that the observed data is not reads, per se, but read *counts*. The reasonableness of this assumption depends very much on how features are defined, but with suitable notion of a "gene" or other feature, it can be a very successful approach. Assuming then that we observe counts (i.e. natural numbers) rather than aligned reads makes modeling the problem easier. In the simplest formulation, we can assume that the read count vector k is distributed according to a Multinomial distribution

$$k \sim \text{Multinomial}(m, x)$$

where m is the total number of reads, and x is a vector proportional to relative transcript or gene expression, summing to 1.

Often this model is approximated by instead assuming each element of k is Poisson distributed.

$$k_j \sim \text{Poisson}(\alpha x_j)$$

where α is some per-sample constant capturing sequencing depth. There are then a variety of approaches available for detecting changes in x_j between samples or conditions. In practice, the negative binomial distribution is more often used as it allows for modeling of *overdispersion*, in this case referring to phenomenon of replicate samples displaying more variability in read counts than can be explained under a naive i.i.d. sampling assumption. The negative binomial distribution can be more intuitively understood as the distribution resulting when the Poisson rate is itself a random variable distributed according to a gamma distribution. So the move from Poisson to negative binomial is just a relaxation of the unreasonably strong assumption that replicates are all identical.

Methods of inference range from various forms of shrinkage estimates (Robinson, McCarthy, and Smyth, 2010; Anders and Huber, 2010), to objective Bayesian approaches (Hardcastle and Kelly, 2010), to fully Bayesian approaches spanning a million cells (Lopez et al., 2018b).

Count-based approaches to non-gene features

Count-based approaches need not focus only on genes. Other features can be defined that allow for mostly unambiguous read assignments. DEXSeq (Anders, Reyes, and Huber, 2012) approaches the problem by considering exon usage. Of course, in the case of alternative splicing or alternative transcript start or termination sites, exons may overlap with one another and result in ambiguous reads. DEXSeq avoids this by partitioning such exons into sections so that each section is either entirely included or entirely excluded in every transcript, a process they describe as "flattening the gene-models." Reads can then be directly counted for these exon sections, and changes in expression, and in some cases, splicing analyzed.

Other count-based methods have more directly interrogated splicing and RNA processing dynamics. JunctionSeq (Hartley and Mullikin, 2016) and LeafCutter (Li et al., 2018b) are both methods that specifically focus on reads crossing splice junctions, with the goal of detecting changes in usage. The former builds off DEXSeq, extending its analysis to include junction counts along with exon segment counts. LeafCutter, on the other hand, focuses only on splice junction counts. This has the advantage of not relying on existing transcript annotations, which may be incomplete or nonexistent. It then seeks to detect changes in splice junction usage by considering the proportion of reads in comparison to overlapping, mutually incompatible introns using a Dirichlet-Multinomial model.

Finally, in a more unusual analysis, La Manno et al. (2017) compared intronic read counts to exonic read counts in single cell data to understand the rate of transcriptional regulation, which they term "RNA velocity." Most RNA-seq protocols enrich the library for spliced mRNA by selecting for polyadenylated RNA. Since splicing is understood to occur very rapidly after (or concurrently with) transcription, most RNA are already spliced by the time they are polyadenylated. Yet enough polyadenylated RNA are not spliced to produce an appreciable number of intronic reads in highly expressed transcripts. Comparing the number of intronic reads to exonic reads be used as a proxy to measure the rate of transcription versus degradation.

Advantages and limitations of sidestepping deconvolution

Methods that avoid deconvolution have serious limitations, but also major advantages in terms simplicity and tractability. They, often very cleverly, shirk the issue of assigning ambiguous reads, and by doing so avoid dealing with cumbersome likelihood or network-flow functions and can write a model in terms of standard families of distributions. Inference with these models is often very fast. Whereas most deconvolution-based methods necessitate careful implementations in lower-level programming languages, count-based methods often have more freedom to write straightforward code in high-level languages.

The significant downside of these approaches is that they discard information. Even when considering broad, simple features like genes, reads with uncertain origin are thrown out or naively assigned, limiting the ability to detect differences in expression between paralogs, or other genes with similar sequences. In the case of exon or splice junction analysis, reads that are not contained in the exon or do not cross the junction, respectively, can still be informative but go ignored. This can be abated with deeper sequencing — we need not be miserly with reads when they are plentiful — but in a certain sense these methods will always be suboptimal.

The second way in which count-based analysis of gene expression loses information is in their nebulous conception of gene expression. If we accept the definition of a gene as a set of related transcripts, then a natural definition of gene expression is the sum of the expression of each of the gene's transcripts. Relying on gene counts without deconvolution can hide differences in expression. Two samples may have the same read count for gene but express different isoforms. Since read count is conflated with transcript length (and other factors that bias RNA-seq), the identical read count could imply very different expression. Not only can ignoring isoform expression hide changes in gene expression, it can even manifest as differential expression in the opposite direction. Indeed, estimating gene expression by deconvolution has been shown to be more accurate (Soneson, Love, and Robinson, 2015).

1.2.2 Methods that embrace deconvolution

Many methods have addressed the deconvolution issue directly. We can broadly group these as either probabilistic or non-probabilistic approaches. The former remains the most common (for good reason, we will argue), but the latter should not go unremarked upon.

In many respects, isoform deconvolution is a simpler problem than demixing, deconvolution, and source separation problems in other fields. For example, topic models in the natural language processing literature sometimes follow a similar form. Documents are treated as a bag of words (analogous to sequenced reads), topics (analogous to transcripts) are categorical distributions over words, and inference attempts to recover the the topic, or mixture of topics, that best explain a document (analogous to an RNA-seq sample). Yet topic models do not typically assume a fixed collection of topics, and instead use a Dirichlet process or other non-parametric approaches to consider an unbounded number of topics learned from the data. On the other hand, in transcript quantification, the set of transcripts is often, but not always assumed to be known. Similarly, it has this advantage over blind source separation problems in signal processing, which are solved without foreknowledge of the sources.

Despite this apparent simplicity, there are a few issues which make RNA-seq quantification an interesting problem. First, unlike words in bag-of-words topic models, reads have a more complex relationship with tran-

scripts. RNA-seq is a fairly elaborate laboratory procedure. Fragmentation, reverse transcription, size selection, amplification, and other steps leave their mark on the distribution of data. There is significant room in RNA-seq to improve quantification by considering these effects.

Second, in superficially similar problems there is often no goal other than decomposition of the signal into parts (e.g., sources or topics). RNA-seq experiments are not just about measuring expression, but understanding biological processes that drive and are driven by expression. These processes have an order and structure that we should hope to model. Deconvolution on its own is not enough. Our goal should be to accurately asses and understand changes in expression.

Nonprobabilistic transcript abundance estimation

Probabilistic approaches to transcript quantification are dominant to the extent that it would be easy to overlook other approaches. Prominent in this category are method that borrow ideas, or at least terminology from network flow. The earliest example is FluxCapacitor (Montgomery et al., 2010). It starts from the assumption that sequencing should generate reads uniformly at random from across a transcript. A set of isoforms is then represented as a graph where nodes are 5' and 3' ends of exons, and edges represent introns and exons. Each edge is shared by some set of isoforms. A linear program is formed to find isoform expression values that explain the number of reads observed at each edge with the contribution from each isoform being fixed according to its expression. Though the terms "flow" and "flux" are used to describe the approach, it is not a network flow approach in any conventional sense. Lin et al. (2012) use integer programming in a somewhat similar manner to FluxCapacitor, but with a more elaborate model encompassing multiple samples.

Later Tomescu et al. (2013) presented the quantification as a true network flow problem. There the graph consists of nodes representing exon segments with edges between segments that appear consecutively together in at least one isoform. They then use network flow methods to find a set of paths through this graph with accompanying expression values that best explain the read coverage according to a cost function. Bernard et al. (2014) presents a somewhat similar method, but show that solving their network flow problem is equivalent to optimizing a penalized likelihood function, in some ways bridging the gap between probabilistic and non-probabilistic approach.

Some of these methods have advantages, particularly when in comes to performance. The most significant downside is that each is an optimization method with no innate way of quantifying uncertainty. Methods like generating bootstrap samples could be applied, but in practice it's not clear they are efficient enough to re-run the optimization many times for each sample, and the issue is given little consideration. Compared to probabilistic methods, it is considerably harder to incorporate a more subtle model

of RNA-seq than includes factors like sequence bias and fragment length probability.

RNA-Seq as a mixture model

Though hardly unanimous, the dominant approach to the transcript quantification problem is to treat transcripts as inducing distinct probability distributions over reads (or read pairs). The experiment as a whole can then be thought of as a mixture model, in which the goal is to infer relative transcript expression (i.e., mixture coefficients). More explicitly, given a set r of m reads, and n annotated transcripts, we define a probability function p_j over possible reads, for every transcript j. The likelihood for a relative expression vector $x \in \Delta^{n-1}$ (here Δ^{n-1} is the open unit $(n-1)$-simplex, i.e., the set of all vectors of length n, with positive entries summing to 1)[2] is

$$P(r|x) = \prod_{i=1}^{m} \sum_{j=1}^{n} x_j p_j(r_i) \tag{1.1}$$

This basic model was first considered by Jiang and Wong (2009). Their analysis differs only in that the $p_j(r_i)$ factors were simply 0 or 1, to indicate compatibility between the read and transcript. Hiller et al. (2009) explores a similar formulation and notes that it is largely analogous to one that had been used successfully to detect splicing changes using microarrays years earlier (Li and Wong, 2001). compatibility/non-compatibility with transcripts, later work by Li et al. (2010), Katz et al. (2010), and Trapnell et al. (2010) offered a more careful analysis in which degrees of compatibility are considered as captured by the $p_j(r_i)$ factors.

Maybe the most consequential choices in this model are around the question of how best to define each transcript read distribution p_j, a question we will return to shortly.

In a maximum likelihood framework, estimates of relative transcript expression are easy enough to make from Equation 1.1 by gradient descent or expectation maximization. From a Bayesian perspective, we would like to be able to approximate the posterior distribution, with respect to some prior $P(x)$ on relative expression

$$P(x|r) \propto P(r|x)P(x)$$

This is amenable to Gibbs sampling, variational inference, among other approaches.

Mixture models have a very long history (e.g., Karl Pearson noted their applicability in 1894 in cases where "the units grouped together in the measured material are not really homogeneous" (Pearson, 1894)). By now, Gibbs sampling and expectation maximization are the textbook approaches

2 An $n-1$ simplex consists of n numbers, but is fundamentally a $n-1$ dimensional space, so this notation may initially seem confusing, but ultimately makes some of the mathematical results less so.

for posterior estimation and maximum likelihood, respectively. Many approaches to RNA-seq quantification use these algorithms quite successfully. They work by introducing latent indicator variables representing assignments of reads to transcripts and applying expectation maximization or the Gibbs sampler. If z_{ij} is 1 if read i originates from transcript j, and 0 otherwise, where $\sum_{j=1}^{n} z_{ij} = 1$, then the likelihood can be written as

$$P(r|x) = \sum_{z \in M_{m \times n}(\{0,1\})} P(r|z)P(z|x) \tag{1.2}$$

where $M_{m \times n}(\{0,1\})$ is the set of m by n matrices over $\{0,1\}$.

To see the equivalence, note that without observing the read, the probability of a read i deriving from transcript j is simply the transcript expression

$$p(z_{ij} = 1|x) = x_j$$

Also, we can write the probability of the reads conditioned on z as

$$P(r|z) = \prod_{i=1}^{m} \prod_{j=1}^{n} \left(p_j(r_i)\right)^{z_{ij}}$$

Then

$$\sum_{z} P(r|x)P(z|x) = \prod_{i=1}^{m} \sum_{z_{i\cdot}} \prod_{j=1}^{n} \left(x_j p_j(r_i)\right)^{z_{ij}} = \prod_{i=1}^{m} \sum_{j=1}^{n} x_j p_j(r_i)$$

The advantage of rewriting the likelihood in terms of z is two-fold. First, it enables easier inference, for example by iteratively find the maximum likelihood solution by computing expectations for each z_{ij} then setting the maximum likelihood x, according to expectation maximization. Second, allows us to explicitly assign and count reads, whereas optimizing Equation 1.1 by gradient ascent would arrive at the same answer without explicitly considering which reads derive from which transcript. Read counts are useful is some analyses, so this approach is more than just a convenience.

Transcripts as fragment distributions

Transcript j's distribution over reads p_j is where the detailed work of modeling RNA-Seq happens. We might reasonably suppose that fragments are uniformly distributed within a transcript. If fragments are of a fixed length ℓ, then a transcript with length L_j has $\max(0, L_j - \ell + 1)$ possible fragments. Thus a uniform distribution over fragments would be written

$$p_j(r_i) = \begin{cases} \frac{1}{L_j - \ell + 1} & \text{if } r_i \text{ is compatible with transcript } j \\ 0 & \text{otherwise} \end{cases} \tag{1.3}$$

Transcripts that are too short to sequence (i.e., $L_j < \ell$), have no well defined distribution over reads, and so could be excluded, but might otherwise be included using an improper distribution p_j that assigns zero to over read.

Complicating this picture, fragments are not of fixed length, but follow some distribution, often controlled by a fragment size-selection step in RNA-Seq protocols, as well as fragmentation and priming probabilities. If we know the marginal distribution over fragment lengths $P(\ell)$, which in practice is easy to estimate if we have access to long, deeply-sequenced exons, we can define a transcript probability function that accounts for variable length fragments.

First, the fragment length for a read pair r_i can always be determined if we assume the fragment derives from a particular transcript t_j, since paired-end sequencing gives us the start and end position of the fragment. Define ℓ_{ij} to be this *implied fragment length* for read pair i and transcript j, which is undefined when the read is incompatible with the transcript. Of course, in single-end sequencing the implied fragment length is not so simple to determine and we must guess at the fragment length or sum over possible fragment lengths. Fortunately, single-end sequencing experiments are increasingly uncommon, in part because of this statistical handicap.

Defining a transcript read distribution accounting for unfixed fragment length is a simple generalization of Equation 1.3.

$$p_j(r_i) := \begin{cases} \frac{P(\ell_{ij})}{\sum_{\ell'=1}^{L_j} P(\ell')(L_j - \ell' + 1)} & \text{if } \ell_{ij} \text{ is defined} \\ 0 & \text{otherwise} \end{cases} \tag{1.4}$$

Some probabilistic methods define this probability function slightly differently, with a denominator $P(\ell \leq T_j)(L_j - \ell_{ij} + 1)$ Pachter (2011). In practice, this choice is mildly consequential, affecting the quantification of short transcripts, but it is not obvious if one version is more correct. We would argue that p_j, defined as it is above, more closely maps to actual RNA-Seq protocols where size selection occurs after fragmentation, whereas in a generative interpretation of the alternative denominator a fragment length is chosen, then a position is chosen conditioned on that length. Beyond this advantage, the math ends up being somewhat neater with the above definition.

More elaborate versions of transcript probability functions can be defined that account for various biases and technical effects. This will be discussed more in Section 1.2.2, but for now this definition is sufficient.

Effective length

The model presented in Equations 1.1 and 1.4 ignore the effect transcript length has on the number of fragments sampled from that transcript. Fragments are generating by randomly cleaving RNA molecules, so it stands to reason that a longer molecule will produce more fragments. As a result, without adjustment, the mixture model infers the mixture of fragment assignments, rather than the mixture of transcripts. This phenomenon is illustrated in Figure 1.1.

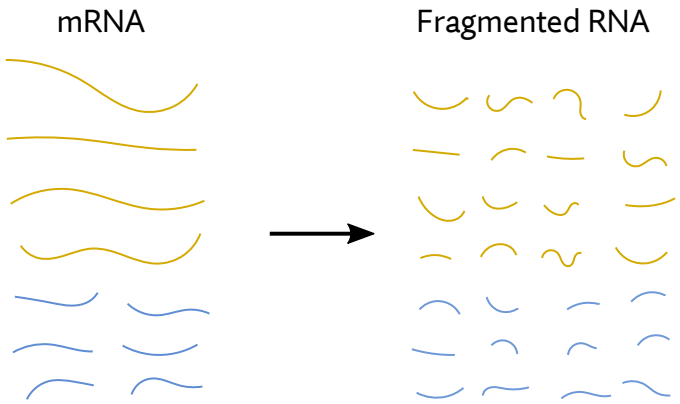

Figure 1.1: The the number of fragments is proportional in expectation the number of copies of a transcript, but also affected by its length. Here the first transcript (colored yellow) produces more fragments than the second transcript (colored blue), despite having fewer copies. Inferring the mixture of fragment assignments is not the same as inferring the mixture of transcripts, necessitating an adjustment for length.

What adjustment is necessary to convert between a mixture of fragments and a mixture of transcripts? In our simple model of fragments being uniformly distributed across transcripts, we posited there are $L - \ell + 1$ opportunities to produce fragments. By this reasoning, the expected number of fragments produced from a single mRNA should be proportional to $L_j - \ell + 1$. So dividing each x_j by $L_j - \ell + 1$ and renormalizing should correct for this effect in in expectation. We can extend this to the case of variable length fragments by taking the expectation with respect to the fragment length distribution. Jiang and Wong (2009) coined this the *effective length*

$$\mathbb{L}_j := E[L_j - \ell + 1] = \sum_{\ell=1}^{L_j} P(\ell)(L_j - \ell + 1) \tag{1.5}$$

Weighted by effective length, the probability of observing some read from transcript j, given the expression vector x, and without observing the read is,

$$P(z_{ij} = 1|x) = \frac{\mathbb{L}_j x_j}{\sum_{k=1}^{n} \mathbb{L}_k x_k} \tag{1.6}$$

The effective length aware likelihood is simply Equation 1.2 with this elaboration on the $P(z|x)$ factor. Expectation maximization and the Gibbs sampler can easily be adapted to include effective lengths (Turro et al., 2011).

In methods that avoid introducing explicit latent read assignments z, the easiest way to incorporate effective length is to consider it a transformation $T_{\mathbb{L}}$ applied to the transcript expression vector x, defined by

$$\left(T_{\mathbb{L}}(x)\right)_j = \frac{\mathbb{L}_j x_j}{\sum_{k=1}^{n} \mathbb{L}_k x_k}$$

$T_{\mathbb{L}}$ is a bijection and the determinant of the determinant of the Jacobian matrix is computable in $O(n)$, so a change of variables from transcript expression to weighted transcript expression, or vice versa, is relatively easy. Appendix A discusses some of the mathematical details of this transformation.

Transcript length is not the only transcript-specific factor which affects the rate at which reads are sampled. Section 1.2.2 discusses modeling other technical biases to arrive at more accurate estimates of x.

Counterintuitive effects of effective length

Effect length can have produce counterintuitive, though not necessarily incorrect results, particularly in the case of zero observed reads, or very short transcripts. Suppose two transcripts a, b have very different effective lengths: $\mathbb{L}_a \ll \mathbb{L}_b$. In an RNA-seq experiment with $m > 0$ reads, suppose none were compatible with either a or b. The maximum likelihood expression estimate (assuming some other transcripts did have compatible reads) will have $x_a = x_b = 0$, an intuitive result.

However, perhaps surprisingly, a will have higher likelihood of expression than b. Or in Bayesian terms, we will deem that a probably has higher expression than b That is, $P(x_a > x_b|r) > 0.5$. This seems odd, because neither a or b have any observed reads, so one might think there is no evidence to distinguish the two. Its useful then to think of RNA-seq as throwing darts (reads) at targets (transcripts) at random. After throwing m darts, neither a or b were hit by any, yet b was a larger target. Missing a large target m times is *more* evidence of its absence than missing a small target m times, explaining the result.

This result is most prominent and most surprising in Bayesian methods that produce posterior mean estimates. Often one will observe shorter transcripts with no observed reads to be estimated to have higher expression estimates than longer transcripts with no observed reads expression. This should not be taken as evidence that the method is wrong or broken, but rather that relying solely on point estimates can give a misleading or incomplete picture.

Units of expression: RPKM, FPKM, and TPM

So far we have discussed transcript expression as a length n vector summing to 1, that is $x \in \Delta^{n-1}$. With many transcripts each element will often be a small decimal number. For convenience these numbers are often converted to units of transcripts per million or *TPM* simply by scaling x by 10^6 (i.e., into a non-unit simplex). TPM was first used by Li and Dewey (2011), and has since become the de facto standard standard unit of expression for RNA-Seq expression and other settings where relative expression is measured.

Prior to the convergence on TPM, two other units were frequently used, and occasionally still are. First was reads per kilobase of transcript per mil-

lion mapped reads or *RPKM*, introduced by Mortazavi et al. (2008). RPKM was refined to *FPKM* by Trapnell et al. (2010) or "expected fragments per kilobase of transcript per million fragments mapped". The emphasis or fragments rather than reads better accommodates paired-end sequencing, but is otherwise similar to RPKM and shares most of the same issues.

The first major issue is that "kilobase of transcript" is ambiguous as to whether transcript length or effective length should be used. Effective length, as we have discussed, is clearly the correct choice, but RPKM and FPKM have often been computed using simple length. If effective length is used, then FPKM is proportional to TPM, however the constant of proportionality can differ between samples, complicating the already non-trivial task of comparing relative expression across samples.

To see this, imagine two samples, each with only two transcripts, with identical read count vectors $k_A = k_B = (10^6, 10^6)$, yet different effective length vectors $\mathbb{L}_A = (50, 50)$, $\mathbb{L}_B = (50, 450)$. The table below shows the corresponding TPM and FPKM vectors for both samples.

Sample	TPM	FPKM
A	$(5 \times 10^5, 5 \times 10^5)$	$(200, 200)$
B	$(9 \times 10^5, 1 \times 10^4)$	$(200, 22.2)$

FPKM is proportional to TPM in both samples, but the constant of proportionality is 2500 for sample A and 4500 for sample B. This is a consequence of scaling by effective length without re-normalizing, as is done with the effective length transformation $T_{\mathbb{L}}$, rendering FPKM (and RPKM) sensitive to the magnitude of effective lengths, rather than what actually matters: relative effective length.

Bayesian analyses

In a standard Bayesian treatment of the quantification problem, we are interested in the posterior distribution $P(x|r) \propto P(r|x)P(x)$. This necessitates choosing a prior distribution $P(x)$ over Δ^{n-1}. The typical choice is a uniform prior on the simplex, which is simply Dirichlet$(\mathbb{1})$. A few methods do use informative priors, for example Salmon (Patro et al., 2017) uses a weakly informative Dirichlet prior (α_0 such that $\alpha_{0j} = 0.001 \cdot \mathbb{L}_j$), which has the effect of slightly biasing expression towards longer transcripts, perhaps to avoid some of the counterintuitive effects effective length has on expression estimates, as discussed above. But, for the most part, there is little basis for constructing an informative prior when estimating expression in one sample.

Under such a uniform prior the maximum posterior point estimates are equivalent to maximum likelihood estimates, but Bayesian methods (e.g., RSEM (Li and Dewey, 2011)) can also generate posterior mean or median point estimates. These are more computationally expensive to produce, necessitating sampling or variational inference, but have some advantages. We argue in Jones et al. (2016) that posterior mean estimates of isoform

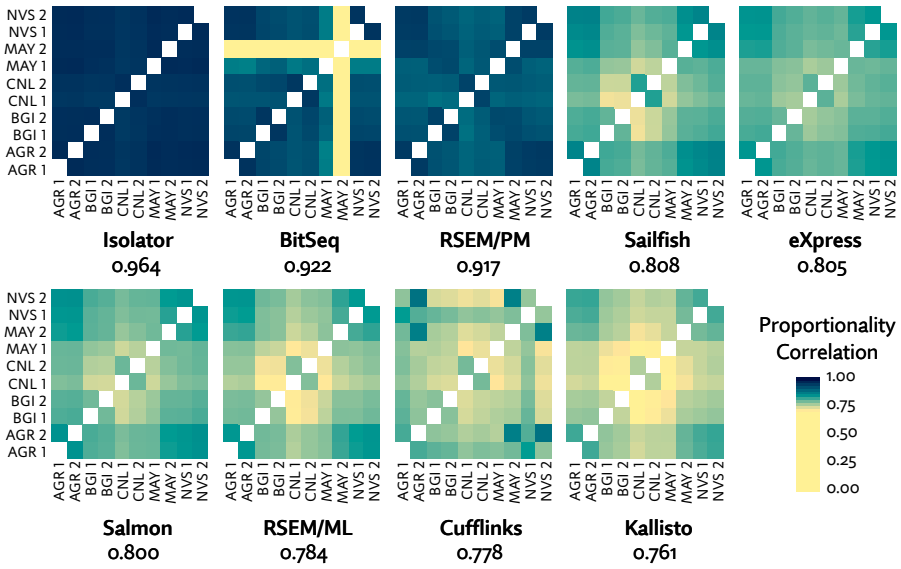

Figure 1.2: A heatmap showing pairwise proportionality correlation (Lovell et al., 2015) between transcript expression of technical replicates estimated using a variety of methods. Samples were sequenced on two flowcells each at five sites, from centrally prepared libraries. Flowcells are numbered arbitrarily 1 or 2 and sequencing sites are abbreviated with three letter codes: Australian Genome Research Facility (AGR), Beijing Genome Institute (BGI), Cornell University (CNL), Mayo Clinic (MAY), and Novartis (NVS). Median proportionality correlation is listed below each heatmap. Isolator, BitSeq, and RSEM/PM all use posterior mean estimates, all other methods are maximum likelihood or maximum a posteriori. Figure reproduced from Jones et al. (2016).

usage are less affected by small changes in coverage, as compared to maximum likelihood or maximum a posteriori estimates, and so preferable in some cases. Figure 1.2 shows how posterior mean estimates produce estimates with higher correlation between technical replicates, regardless of the implementation.

Where Bayesian analysis does depart even more significantly from maximum likelihood methods is in their treatment of models over multiple samples. Multi-sample models, for example, those developed in Huang and Sanguinetti (2017) and Li et al. (2018a), are discussed in Section 1.3. There may be little basis for informative priors on a single sample, but when multiple samples are involved there is a case to be made for models that shrink estimates towards global- or condition-wise means, or use sequence features to inform splicing predictions.

Bertrand's paradox in RNA-Seq analysis

Adopting a uniform Dirichlet$(\mathbf{1})$ prior on expression is a reasonable choice, justified through the "principle of indifference": without explicit justification to predict one alternative over another, our should assign equal prob-

abilities. A common critique of this reasoning turns out to relevant to Bayesian analyses of RNA-Seq. Bertrand's paradox (Shackel, 2007) is posed in various ways, but in one telling, we consider a factory that produces cubic boxes with side lengths between 0 and 1 meters, but otherwise of unknown size. In a Bayesian analysis, before observing any boxes, we should have a prior distribution over box sizes. But the principle of indifference can be used to justify a uniform prior over side lengths, or over the area of sides of the box, or over the volume of the box. Each imply a very different distribution. The principle of indifference is not a get out of jail free card for having to choose some subjective credal starting point.

In RNA-Seq this issue manifests in at least two ways. First, we might choose a uniform prior over expression values before or after transformation using the effective length transformation, which result in the different priors and thus different posterior estimates, particularly for short transcripts. The quantity of interest is usually the expression values after adjusting for length, but it can be easier to implement a prior over unadjusted values. Methods often do not clarify which is being used.

Secondly, we could choose a uniform prior over transcript expression, or over gene expression and isoform usage (i.e., a composition of subcompositions). The former is perhaps simpler or more straightforward but results in the potentially surprising outcome where we award higher belief to a gene's expression merely for having more annotated isoforms. There is not an obvious resolution to "the problem of the priors" here, nor in Bayesian epistemology in general. In most cases this plays a minor role and only for low expression transcripts, but does play some role and tends to go unremarked upon.

Modeling and incorporating bias and technical effects

Models discussed up to this point have tended to assume that reads are sampled uniformly at random. Any superficial glance at the data shows that this is far from true (Mortazavi et al., 2008; Dohm et al., 2008). Indeed, we should not reasonably expect it to be true. Many steps commonly used in RNA-seq protocols have biases that have been well documented prior to their use in sequencing. RNA is often fragmented using sonication, which does not cleave uniformly at random. As demonstrated by Grokhovsky (2006), exactly where the RNA molecule is cleaved is sensitive to fragment size, pH, ionic strength, and temperature. After reverse transcription, cDNA is often amplified by PCR, but amplification efficiency is well known to be sensitive to the sequence being amplified, particularly to GC content. One study suggests reduced amplification efficiency of high GC content sequences is likely due to disruption of priming and elongation by secondary structure, in a addition to high GC molecules being less easily denatured (Dutton, Paynton, and Sommer, 1993). Read coverage across the transcript can vary according the library preparation. Wang, Gerstein,

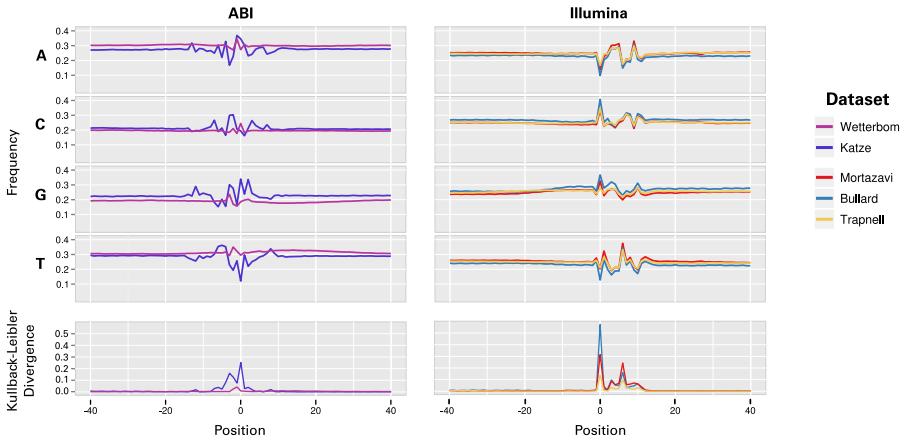

Figure 1.3: Sequence specific bias observed in a number of datasets. Nucleotide frequencies are plotted relative to the start (labeled position 0) of each mapped read, respecting strand, and grouped by platform (Illumina or ABI SOLiD). The plotted datasets were taken from Wetterbom et al. (2010), Jones et al. (2012), Mortazavi et al. (2008), Bullard et al. (2010), and Trapnell et al. (2010). The sequence is taken from the genomic context surrounding the read, so that -40 to -1, for example, fall outside the read sequence itself. The symmetrized Kullback-Leibler divergence is used to summarize the difference in nucleotide frequency compared to a fixed estimate of background nucleotide frequencies made by sampling many positions near mapped reads. Under the assumption that reads are sampled uniformly from transcripts, each of the plots should be essentially flat. Figure reproduced from Jones et al. (2012)

and Snyder (2009) noted striking differences depending on whether RNA or cDNA is fragmented, each choice biased in its own way.

Hansen, Brenner, and Dudoit (2010) first noted that reads additionally show very strong sequence specific bias in the leading few nucleotides, and plausibly hypothesized that this is due to reverse transcription initialized with random hexamer primers, a step common in RNA-seq. Varying binding affinities and non-uniform hexamer mixtures may cause certain fragments to be preferably reverse transcribed, though they find that computationally predicted binding affinities did not explain the bias. Figure 1.3 shows examples of the severity and prevalence of this source of bias.

Critically, this bias effects both ends of the fragment, consistent with the notion that the bias is driven by fragmentation and cDNA synthesis by random priming. For example, in an example dataset a fragment is more likely to begin and end with C than A by a wide margin, and the nucleotide bias at the start and end of the fragment acts multiplicatively (Figure 1.4). As a consequence, bias needs to be addressed at the fragment level, rather than the read level, and single-end sequencing is at a severe disadvantage as only one end of the fragment is observed.

Figures 1.5 and 1.6 demonstrate two other sources of bias: positional bias and fragment GC bias. Positional bias refers to broad trends in non-

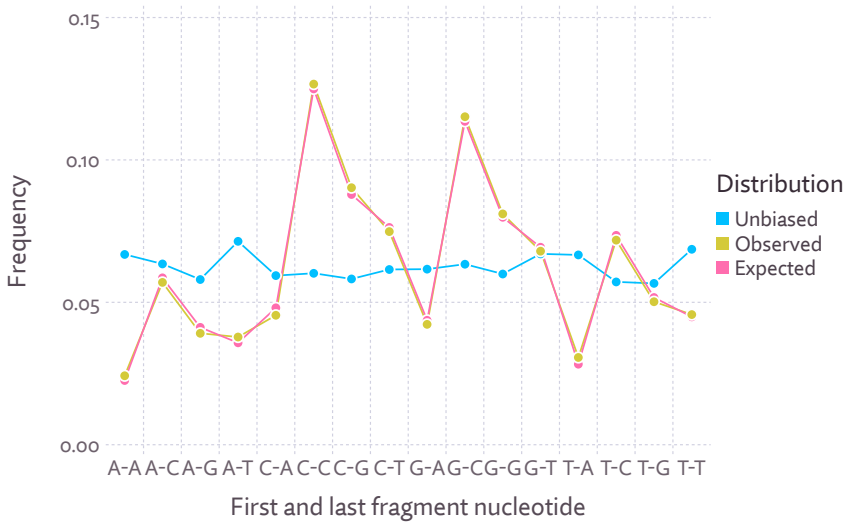

Figure 1.4: The frequency of pairs nucleotides flanking sequenced fragments at their 5' and 3' ends in one example dataset taken from the Sequencing Quality Control project (SEQC/MAQC-III Consortium, 2014) with accession number SRR897059. "Observed" plots the observed frequencies, "Unbiased" shows the distribution expected if fragments were sampled uniformly at random in proportion to a transcripts expression. "Expected", which matches "Observed" very closely, shows frequencies that would be expected if 5' and 3' biases were independent and thus multiplicative, an assumption that seems to hold.

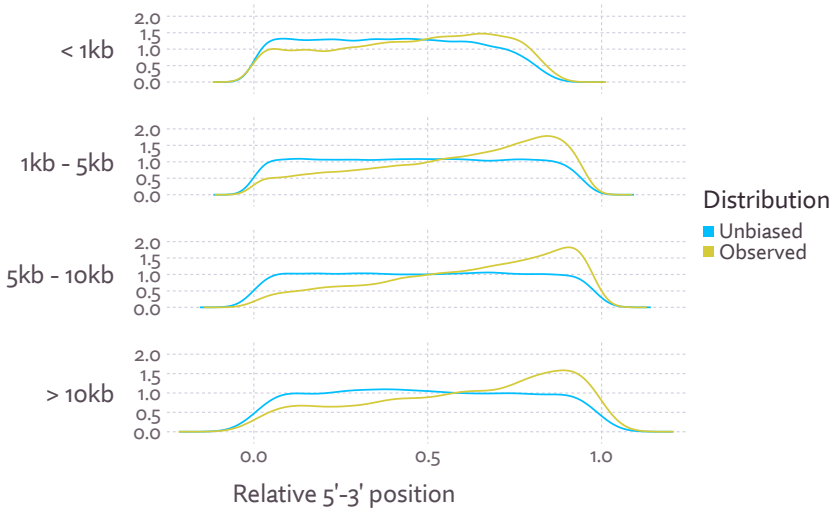

Figure 1.5: Positional bias plotted for an example dataset (described in the caption of Figure 1.4). Kernel density estimates of the distribution of relative 5' to 3' position of fragments is shown, separated into four bins by transcript length. The "Unbiased" lines show the same data with the reads shuffled as to place them uniformly at random in the same transcripts. The skew of the "Observed" data towards 1.0, indicates a 3' bias common to RNA-Seq protocols with a poly(A) selection step.

uniform coverage across transcripts. For example, a common positional bias is increased read coverage towards the 3' end of transcripts, particularly on long transcripts, plausibly caused by selection for RNA molecules with poly(A) tails, a step common in many protocols. Fragment GC bias often manifests as observing fewer low- or high-GC fragments than expected. As noted, this type of GC bias in PCR amplification is well documented. Love, Hogenesch, and Irizarry (2016) additionally observes biases caused by long runs of G or C nucleotides, apart from the overall GC content.

Modeling bias

Hansen, Brenner, and Dudoit (2010) sought to correct biased read counts by estimating the distribution of 7-mers at both the beginning of the read (where the bias is observed) and the end (where it is not observed). Read counts are then corrected, in expectation, by scaling them by the ratio of unbiased to biased 7-mer probability. Li, Jiang, and Wong (2010) capture the same effect using a more sophisticated regression tree model. While these works focused on weighting read counts to correct for bias, Roberts et al. (2011) first incorporated bias into the standard probabilistic model of transcript expression (Equation 1.1). Their model for bias is conceptually similar to that of Hansen, Brenner, and Dudoit (2010), but using a sparser

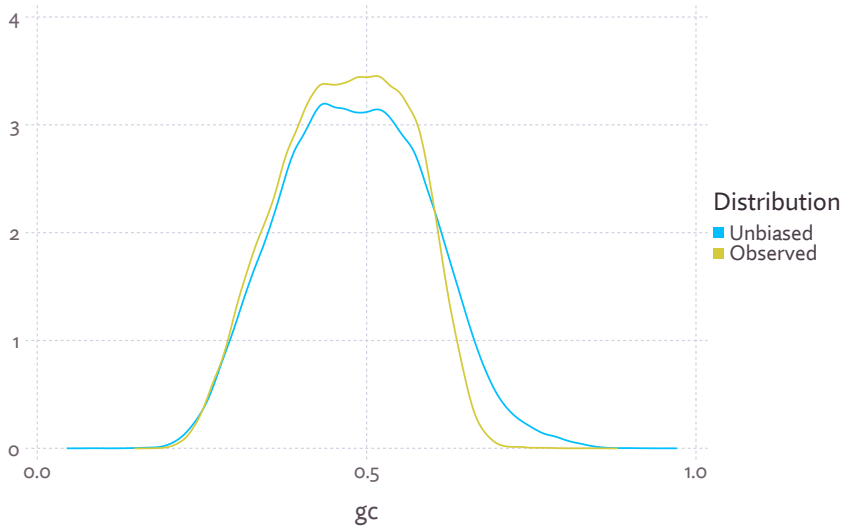

Figure 1.6: Fragment GC bias plotted for an example dataset (described in the caption of Figure 1.4). "Observed" shows the distribution of the proportion of G and C nucleotides within observed fragments, while "Unbiased" shows this same distribution after shuffling the fragments uniformly at random within their assigned transcripts. In this data we see a depletion of fragments with very high GC content, consistent with effects from PCR amplification.

variable-order Markov model to capture sequence probabilities. They also model broader coverage biases across transcripts.

We proposed a separate method of capturing sequence bias which finds a sparse representation of sequence probabilities by heuristically optimizing over graphical model structures Jones et al. (2012). In comparison, the sparse graphical model used by Roberts et al. (2011) is fixed based on prior datasets, and so does not necessarily generalize well. Additionally, we show that optimizing over structures in this way has has very low probability of falsely correcting for bias where there is none, which is less clearly the case with fixed-structure models. Zhang et al. (2017) demonstrated that a small improvements are achievable using recurrent neural networks.

Love, Hogenesch, and Irizarry (2016) has built perhaps the most holistic model of RNA-Seq bias to date, incorporating sequence bias at fragment ends, fragment GC bias, positional bias, and presence of uninterrupted GC runs. Their method, *alpine*, works by optimizing model parameters in a GLM framework, as to most accurately predict coverage patterns using the selected features.

Correcting bias

Assuming bias can be reasonably estimated, how should models of RNA-Seq incorporate that information? Early work (Hansen, Brenner, and Dudoit, 2010; Li, Jiang, and Wong, 2010; Jones et al., 2012) focused or reweighing read counts. It is worth discussing the rationale for weighting read counts, before delving into the full probabilistic models.

The read count for transcript j can be defined as

$$k_j = \sum_{i=1}^{m} z_{ij}$$

where z_{ij} is a 0/1 indicator that is 1 iff read i originates from transcript j. The expectation of these indicator variables is simply the effective length weighted transcript expression: $E[z_{ij}] = P(z_{ij} = 1) = x'_j$. So a transcript's expected read count is the its effective length adjusted expression scaled by the total number of reads in the experiment

$$E[k_j] = E\left[\sum_{i=1}^{m} z_{ij}\right] = mx'_j$$

Under ideal, unbiased circumstances, observing some potential confounders for a read would have no bearing on this math. Let $c(f)$ be the confounding features (e.g., GC content, position along its transcript) corresponding to a particular fragment f. If f_{ij} is the fragment implied by read i originating from transcript j, then an experiment is unbiased if z_{ij} and $c(f_{ij})$ are independent, for all i, j. Observing the potential confounders for a read tells us nothing about whether such a read will be generated. On the other hand, if the experiment is biased, the read count will be a biased estimate of the transcript expression, which is to say $E[k_j|c] \neq mx'_j$

With Bayes' theorem we can write expectation for z_{ij}, conditioned on $c(f_{ij})$ as

$$\begin{aligned} E[z_{ij}|c(f_{ij})] &= P(z_{ij} = 1|c(f_{ij})) \\ &= \frac{P(c(f_{ij})|z_{ij} = 1)P(z_{ij} = 1)}{P(c(f_{ij}))} \\ &= x'_j \frac{P(c(f_{ij})|z_{ij} = 1)}{P(c(f_{ij}))} \end{aligned}$$

If we define the *bias* for a fragment to be

$$b(f_{ij}) := \frac{P(c(f_{ij})|z_{ij} = 1)}{P(c(f_{ij}))} \tag{1.7}$$

This definition of bias makes some intuitive sense. The denominator represents the background probability of the confounding bias features, while the numerator gives the probability conditioned on the features corresponding to a sequenced read. In an unbiased setting, this ratio should be 1, with no confounding feature under- or over-represented among sequenced positions.

It is easy to show that dividing z_{ij} by its bias yields an unbiased estimator we will call k'

$$k'_j := \sum_{i=1}^{m} z_{ij}/b(f_{ij})$$

$$E[k'_j|c] = E\left[\sum_{i=1}^{m} z_{ij}/b(f_{ij})|c\right] = \sum_{i=1}^{m} E\left[z_{ij} \mid c_{ij}\right]/b(f_{ij}) = \sum_{i=1}^{m} x'_j = mx'_j$$

So the justification for weighting reads by the reciprocal of the bias becomes clear. A potentially biased estimator becomes unbiased. Yet read weighting approach has some downsides. First, while k_j is a natural number and thus can be modeled by a Poisson or Negative-Binomial distribution, k'_j is a non-negative real without an obvious choice of modeling distribution.

An alternative approach is to estimate transcript-level bias b_j, and model counts as, for example,

$$k_j \sim \text{Poisson}(b_j \lambda_j)$$

where λ_j then represents some unbiased measure of expression proportional to x'_j.

Roberts et al. (2011) and Pachter (2011) describe how to incorporate bias estimates into a full probabilistic model of transcript abundance. We describe a simplified variation of that approach here, elaborating on Equation 1.1. The key insight is that bias has two effects on the model as presented so far:

1. The effective length $\mathbb{L}_j$ for each transcript j.

2. The shape of the fragment distribution $p_j(\cdot)$ for each transcript j.

We proceed by simply weighting each fragment by its bias in Equations 1.5 and 1.6. Let F_j be the set of all possible fragments of transcript j, then we can redefine effective length to account for bias

$$\tilde{\mathbb{L}}_j := \sum_{f \in F_j} P(\ell = |f|) b(f) \tag{1.8}$$

where $|f|$ is the length of fragment f.

Similarly, transcript probability functions can be redefined,

$$p_j(r_i) = \begin{cases} P(\ell = \ell_{ij}) b(f_{ij}) / \tilde{\mathbb{L}}_j & \text{if } \ell_{ij} \text{ is defined} \\ 0 & \text{otherwise} \end{cases} \tag{1.9}$$

Note that when $b(\cdot)$ is 1 for all fragment (i.e. the experiment is unbiased), these equations reduce to Equations 1.5 and 1.6, respectively.

As with Equation 1.6, there are variations of this approach, which yield slightly different models, but have no bearing on the methodology developed in following sections. We have attempted to present the most straightforward version, but also the version which we ultimately implement.

1.2.3 Approximations

When sequencing is deep or an experiment has many samples, aligning reads to a genome or transcriptome and inferring transcript expression can be both be computationally burdensome. This encouraged some to explore approximations. A great deal of success has been seen in avoiding explicit read alignment, and determining the compatibility between reads and transcripts by other means. Additionally faster approximations of the likelihood function have been introduced that quantize the likelihood assignments in order to approximately factor it into a smaller number of factors.

Approximating read alignment

A precursor step to transcript and gene quantification using RNA-Seq reads has typically been alignment, either to a reference genome sequence or transcript sequences. A number of specialized short read aligners have been built to perform this task with increasing efficiency. The latest generation of short read aligners are quite efficient indeed (see for example STAR (Dobin et al., 2013) or HISAT (Kim, Langmead, and Salzberg, 2015)), but still represent a major bottleneck in analysis.

Bypassing the alignment process was first explored by Patro, Mount, and Kingsford (2014) in a method called Sailfish. We say that read i is *compatible* with transcript j iff $p_j(r_i) > 0$. Sailfish is able to determine compatible pairs of transcripts and reads directly, rather than first using a short read

aligner, then comparing these alignments to annotated transcripts. To do this it builds an index of k-mers present in the transcript sequences, using a minimal perfect hash function. Reads are then "shredded", or deconstructed into their constituent k-mers, which are hashed with this index function to determine the set of transcripts each k-mer is compatible with. With this compatibility matrix in hand, inference works much like other probabilistic method, using expectation maximization, but substituting k-mers for reads.

The method Sailfish introduced dramatically speeds up transcript quantification, by avoiding alignment, but it discards a significant amount information by shredding reads into k-mers. Bray et al. (2016) introduced Kallisto, a refined approximate alignment method that loses much less information. Kallisto again indexes k-mers present in the transcript sequences, but preserves the connectivity between those k-mers in a de Bruijn Graph structure.

As they were originally defined, de Bruijn graphs refer to the graph formed with $|A|^k$ nodes representing every length k string of over some alphabet A. The graph contains an edge between two nodes iff if one string can be formed from the other by deleting one letter from the left or right end and adding another letter to the opposite end. For example, there would be an edge between *abab* and *babb*. In the context of nucleotide sequence analysis, de Bruijn graphs refer to a subgraph of the de Bruijn graph over nucleotide k-mers, formed by only including nodes representing k-mers present in the transcript sequences. A sequence then represents a path through the de Bruijn graph.

With an annotated version of this data structure, Kallisto is able to determine if a read is compatible with a transcript by checking that it has a complete path through the transcript de Bruijn graph, and that the path is a subpath of one or more transcripts. This offers a major advantage over shredding reads and pretending k-mers are independent observations. The authors of Sailfish implemented their own variation of this idea is Salmon (Patro et al., 2017), as well an RapMap (Srivastava et al., 2016), a standalone implementation of these ideas that can be used to produce efficient approximate alignment for use with other tools.

Approximating the likelihood function

Faster analysis can also be achieved through approximating the likelihood function. Though hardly prohibitively expensive to evaluate on its own, Gibbs sampling or or variation may necessitate thousands of evaluations. This becomes a particular issue when multi-sample models are considered. Little work has been done on this front, with the the notable exception of Zakeri et al. (2017), who developed an approximation in which Equation 1.1 can be effectively factored, by treating as equivalent reads than have similar $p_j(r_i)$ values for every transcript j.

The likelihood function can be factored when two or more reads share the same transcript probabilities. For example, if $p_j(r_1) = p_j(r_2)$ for all transcripts j, then

$$\prod_{i=1}^{2}\sum_{j=1}^{n} x_j p_j(r_i) = \left(\sum_{j=1}^{n} x_j p_j(r_1)\right)^2$$

More generally, the set of reads can be partitioned into a set of equivalence classes $\mathcal{C}$, and for any class $\mathcal{F} \in \mathcal{C}$, any pair of reads $r_a, r_b \in \mathcal{F}$ share the same transcript probabilities: $\forall j, p_j(r_a) = p_j(r_b)$. Equation 1.1 can then be rewritten

$$P(r|x) = \prod_{i=1}^{m}\sum_{j=1}^{n} x_j p_j(r_i) = \prod_{\mathcal{F}\in\mathcal{C}}\left(\sum_{j=1}^{n} x_j p_j(r_{\mathcal{F}})\right)^{|\mathcal{F}|}$$

where we use $r_{\mathcal{F}}$ to represent an arbitrary read in the equivalence class $\mathcal{F}$.

In the worst case, every read is in its own equivalence class, but if there are a large number of equivalent reads, then the set of reads can be partitioned into a smaller number of equivalence classes, and the likelihood can be computed more efficiently with fewer factors. Unfortunately, with more sophisticated modeling of RNA-Seq, fewer non-identical reads with identical probabilities will be observed. In a detailed model encompassing various biases, two reads are unlikely to have exactly the same probability for all transcripts, unless they represent multiple copies of the same fragment. Zakeri et al. (2017) demonstrated that even with more elaborate RNA-Seq models, there are many nearly-compatible reads. With little loss to accuracy these near-compatible reads can be collapsed into approximate equivalence classes, speeding up inference significantly.

Ultimately, we will propose to take the idea of likelihood approximation further, developing a much more aggressive approximation of the likelihood function, but great gains can be made even with the straightforward clustering or quantization.

1.3 The two-step style of RNA-Seq analysis

Relative transcript expression for an individual RNA-Seq sample is not typically interesting on its own. RNA-Seq experiments are nearly always concerned with detecting transcriptional changes between groups of samples. From a Bayesian perspective, we would like to build a model of transcriptional changes among k samples consisting of sets of reads $r^{(1)}, \ldots, r^{(k)}$, with model parameters θ (e.g., effect sizes, pooled means, or latent space encodings), then consider the posterior distribution $P(\theta|r^{(1)}, \ldots, r^{(k)}) \propto P(r^{(1)}, \ldots, r^{(k)}|\theta)P(\theta)$.

If we adopt the likelihood function in Equation 1.1, then $r^{(i)}$ is independent of all other variables when conditioned on $x^{(i)}$. In typical models expression vectors $x^{(i)}$ will also be mutually independent when conditioned on the model parameters θ, so we can write this posterior distribution in

terms of the latent expression vectors $x^{(1)}, \ldots, x^{(k)}$, treating them as nuisance parameters.

$$\begin{aligned} & P(\theta | r^{(1)}, \ldots, r^{(k)}) \\ & \propto \int_x P(r^{(1)}, \ldots, r^{(k)} | x^{(1)}, \ldots, x^{(k)}) \\ & \quad P(x^{(1)}, \ldots, x^{(k)} | \theta) P(\theta) dx \\ & = \int_x \prod_{s=1}^{k} P(r^{(s)} | x^{(s)}) P(x^{(s)} | \theta) P(\theta) dx \end{aligned} \tag{1.10}$$

This is all very straightforward but presents some practical problems. To evaluate the likelihood functions, some information about each unique read must be stored (typically in a sparse matrix where entry i, j corresponds to the probability assigned to the ith read by the jth transcript distribution). This translates to hundreds of megabytes to several gigabytes per sample. Estimating the posterior for moderately large experiments requires either a great deal of memory or cycling read data in and out of memory, as is done in stochastic gradient methods.

In practice, this kind of textbook model is often short-circuited. Point estimates are made for transcript expression vectors x, usually by maximum likelihood

$$x^{(i)*} = \arg\max_{x^{(i)}} P(r^{(i)} | x^{(i)})$$

Then these are plugged into the full model, forming an alternative posterior distribution

$$P(\theta | x^{(1)*}, \ldots, x^{(k)*})$$

This model adopts the (false) assumption that these expression values are observed, substituting them for what is actually observed: the reads. Non-Bayesian models have the same issues, and often resort to the same two-step approach. When estimates are treated as observations, the uncertainty of these estimates is flushed from the analysis, artificially inflating the certainty of the end result. This two-step approach can be thought of as an approximation of the desired model, but one that captures none of the uncertainty of read assignments.

Gelman (2016) describes the way in which statistics is used carelessly or unscrupulously to transmute randomness into the perception of certainty as "uncertainty laundering." The practice of using count-based analysis on estimated counts is a specific and pernicious form of this phenomenon.

1.3.1 Dealing with estimation uncertainty

Some methods have been developed to account for this estimation uncertainty. In prior work, we built a joint model that avoids the two step approach, directly conditioning on reads (Jones et al., 2016). This produced accurate expression estimates but only scaled to a relatively small number

of samples. Similarly, MSIQ (Li et al., 2018a) has shown that more accurate estimates can be produced in a joint model, as estimates can be pooled and shrunk towards similar samples. They implement a full Bayesian hierarchical model over all reads and transcripts in the dataset, using a Gibbs sampler for inference. It is unclear how well this method scales, which goes conspicuously unremarked upon, while the largest dataset the method is demonstrated on consisted of 10 samples and a subset of the annotated transcripts.

Other full probabilistic models have emphasized the importance of fully accounting for uncertainty when studying splicing in single-cell data, where coverage per-cell is often low. BRIE (Huang and Sanguinetti, 2017), like MSIQ implements a Bayesian hierarchical model which attempts to overcome the limitations in coverage by regressing on sequence features, effectively shrinking estimates among splicing features that share similar sequencing motifs, exon lengths, conservation scores, etc. The method does not do full inference, but relies on an ad-hoc approximation: isoform expression is randomly sampled for each cell using Metropolis-Hastings, then global regression coefficients are maximized. In this way the method is reminiscent of stochastic expectation-maximization, but without any its theoretical guarantees (Celeux and Diebolt, 1985). BRIE was demonstrated on a larger dataset than MSIQ — 96 cells — but this is still quite small compared to other modern single-cell experiments. It is unclear how well it would scale to larger experiments, and how easily the ad-hoc approximate inference could be adapted to other types of models.

BitSeq (Glaus, Honkela, and Rattray, 2012) implements a two step model in which transcript expression is sampled using MCMC and these samples used in the second step as "pseudo-data". MMSEQ (Turro, Astle, and Tavaré, 2014) incorporates MCMC samples by fitting a Normal distribution to the samples and using that distribution in the differential expression model, while Swish (Zhu et al., 2019) introduced a way of using MCMC samples directly in nonparametric tests of differential expression. In later work, variational inference was added to BitSeq (Hensman et al., 2015), but observing an underestimation of variance, the authors express reservations about using it to call differential expression. Using a generalized Dirichlet distribution (Papastamoulis et al., 2014) was shown to help reduce this issue.

In sleuth (Pimentel et al., 2017), variance is estimated from bootstrap samples of maximum likelihood estimates. Incorporating these variance estimates into regression models, they show substantial improvements in accuracy when calling differential expression, particularly at the transcript level. IsoDE2 (Mandric et al., 2017) instead used bootstrap samples to directly compute confidence intervals over fold changes. Bootstrap methods do have limitations. Estimates of variance are guaranteed to converge asymptotically to the true value with enough reads, but this leaves lightly sequenced loci with potentially unreliable estimates.

Whether using the bootstrap or MCMC, generated samples have limited applications. For example, powerful probabilistic programming languages have become an increasingly popular way to implement models, but efficient inference is usually performed using variational inference or some form of Hamiltonian Monte Carlo which rely on directly evaluating and differentiating the likelihood function. Variational methods of approximating the likelihood produce just such a function, but are highly contingent on the distribution family used to make the approximation. An insufficiently flexibly family will systematically underestimate variance, inflating false positives.

Many avenues have been explored in the last decade since RNA-seq entered the scene, but the straightforward goal of incorporating deconvolution directly into larger inferential problems has yet to be fully realized at scale. The attempts have seen promising results, but run up against the inherent computational complexity of the problem. The standard form of the RNA-seq likelihood function is simply expensive to compute when large numbers of samples are involved. Even the most efficient implementations have to store a conditional probability for every intersection between a read and transcript, which can equal hundreds of megabytes per sample. As modern RNA-seq moves towards analysis of many thousands of cells, this approach has no hope of keeping up, hence the focus on count-based methods in many contemporary single-cell experiments.

In the following sections we consider carefully the space of possible approximations to the RNA-seq likelihood, ultimately arriving at one that is able to capture the likelihood function with surprising fidelity using only three parameters per transcript. With this approximation in hand, inference in full probabilistic models over billions of reads becomes a possibility on even inexpensive computers. We show that bread and butter analyses like differential expression, regression, and dimensionality reduction become considerably more robust when used with the full likelihood of the model, and that splicing dynamics can be explored in previously impractical ways. Far from a specialized trick to massage the feasibility of one narrowly defined model, likelihood approximation marks a suitable way forward in any RNA-seq quantification problem that has heretofore been forced to choose between tractable inference or a suboptimal model.

2

Approximating the RNA-seq likelihood function

In this section we develop a novel approximation for the RNA-Seq likelihood function. Because the likelihood function for a full experiment factors into per-sample likelihood functions (as in Equation 1.10), this approximation can be built one sample at a time. Once fit, evaluating and sampling from the approximation is orders of magnitude faster than using the likelihood function. Substituting this approximation for the real thing can make inference on full probabilistic models, with billions of reads, tractable on even modest computers.

2.1 Approximating likelihood with variational inference

Variational inference is usually presented as a means of estimating an otherwise intractable posterior distribution. Given a distribution function p, and a family of distributions $q_\phi(\cdot)$ parameterized by ϕ, we fit q to p, given some data y, by choosing ϕ to minimize the Kullback-Leibler (KL) divergence,

$$\arg\min_{\phi} D_{\mathrm{KL}}(q_\phi(\theta)||p(\theta|y))$$

A key feature of this method of inference is that q does not depend directly on y, so that once ϕ is optimized, the approximate probability can be evaluated without the data, only retaining the typically much smaller parameter vector ϕ. Though this is the most common approach to variational inference, it's far from the only. For example, other divergence functions have been proposed (e.g., Rényi divergences (Li and Turner, 2016)), and sometimes approximations are used that are explicitly a function of y, as is the case in variational autoencoders.

This suggests a solution to the issue of building large joint models. Instead of using variational inference to approximate a posterior distribution, we can use it to separately approximate the likelihood of each sample. By substituting an approximation $q_\phi(x)$ for each sample's likelihood function $P(r|x)$, we can capture the likelihood with some fidelity without having to keep the RNA-Seq reads in memory. This would allow us to build a very large model, encompassing hundreds or thousands of samples, that can be run on laptops or meager servers.

Of course, the likelihood is not a distribution over expression vectors x but over reads r, so the KL divergence is not well-defined here. But in models making use of the likelihood function, multiplicative constants are generally irrelevant, so our approximation only has to be proportional to the likelihood. To bring the machinery of variational inference to bear, in-

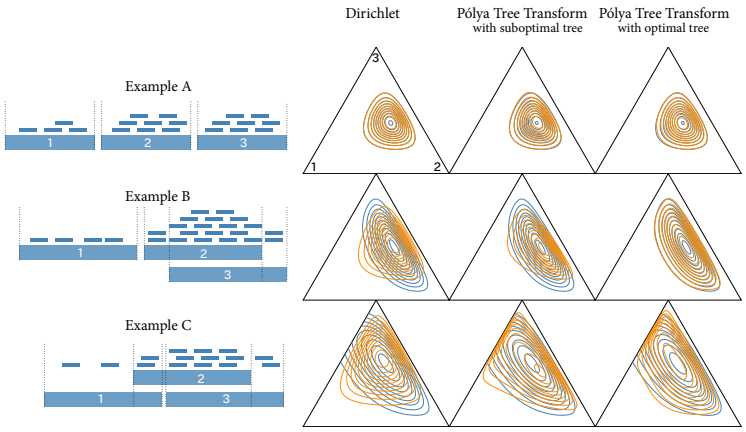

Figure 2.1: Three contrived RNA-Seq examples are shown on the left. On the right, in blue, contours of the their likelihood functions are plotted, and in orange, various approximations made by minimizing KL divergence. The approximation we propose, based on what we call the Pólya tree transformation, is shown in the second and third columns. This transformation is defined in terms of a tree. Choosing the optimal tree plays a large role in how well the approximation fits. In example A, every read can be unambiguously attributed to a single transcript. In this easy case, a Dirichlet distribution is perfectly proportional, and the approximation, regardless of the tree, is a near perfect fit. In examples B and C, reads cannot by unambiguously assigned. Here, the proposed approximation remains a good fit, provided the right tree is selected. Using the approximation in place of the exact likelihood function can enable dramatically more efficient inference.

stead of approximating likelihood directly, we approximate a normalized likelihood function, mathematically equivalent to a posterior distribution under a uniform prior. We will denote with $\mathcal{P}(x|r)$ this normalized likelihood function, defined simply as

$$\mathcal{P}(x|r) := \frac{P(r|x)}{\int_{x \in \Delta^{n-1}} P(r|x)} \tag{2.1}$$

In Figure 2.1, some illustrative examples of the approach we develop in the following sections are shown. We can see that capturing the dependence structure of $\mathcal{P}$ requires careful selection of the distribution family being used, but it is often possible to do so very accurately. In simple cases lacking any read ambiguity, the likelihood is proportional to a Dirichlet distribution, but this model is inadequate for cases where reads are compatible with multiple transcripts. The model we propose can perfectly capture the cases of zero read ambiguity (see Theorem 2.2.11 in Section 2.2.4), but is strictly more expressive, and able to capture more complex dependence structures while being similarly efficient (i.e. linear time and space in the number of transcripts).

2.1.1 Practical benefits to likelihood approximation

Our approach to approximation is unusual: factoring the likelihood, then separately fitting a proportional approximation to each factor. Though it could be used in other settings, it is particularly well suited for RNA-Seq, compared to other possible approaches.

The obvious alternative for tractable inference is simply to use variational inference on the posterior we are actually interested in. That is, if we have a model of, say, differential expression, with parameters θ, we want to approximate the intractable posterior $P(\theta|r)$. For a large experiment, all of the reads r will not fit in memory, but this problem is amenable to stochastic variational inference, or SVI (Hoffman et al., 2013). In SVI, batches of data are subsampled to update estimates of "local" latent parameters (transcript expression estimates $x^{(i)}$ for each sample i), before updating "global" latent parameters (θ).

Because SVI algorithms must cycle many gigabytes of sequencing data in and out of memory to repeatedly compute stochastic gradients, they are likely much less efficient than likelihood approximation. Two additional considerations further increase the latter's desirability.

First is reusability. Large RNA-Seq experiments can be rich with insight, and lend themselves to multiple analyses. Differential expression, differential splicing, clustering, and dimensionality reduction are separate tasks that might be carried out on the same data, each with its own model, each representing a separate inference task. For these tasks, likelihood approximations must be made only once, after which they can be reused over and over. Amortized over every iteration of every analysis typically run on a dataset, likelihood approximation is far more efficient than other approaches to tractable inference.

Second, likelihood approximation can obviate some of the cumbersome data transfer and storage issues with RNA-Seq. High throughput sequencing produces huge datasets, which must be stored and transferred to collaborators, which can mean waiting on long downloads or exchanging hard drives. Our approach to approximated likelihood, on the other hand, summarizes all expression information for a sample using only a few megabytes per sample. The 1461 brain samples produced by the GTEx project (GTEx Consortium, 2013), are reduced to 7.8GB of likelihood approximation data. The most compact exact representation of the likelihood function for this experiment would be about 1.5TB, often a prohibitively large amount of data to keep in memory.

2.1.2 Minimizing KL divergence

In some cases there is an analytical solution to the best fitting surrogate distribution $q_\phi(\cdot)$. Other models can be optimized using iterative approaches analogous to expectation maximization (Bishop, 2006a). In complex or exotic models, some form of gradient descent is typically used.

A standard result in variational inference (Jordan et al., 1999) is that minimizing the KL divergence is equivalent to maximizing the *evidence lower bound (ELBO)*, defined as

$$\text{ELBO}(\phi) = E_{\theta \sim q_\phi}[\log p(\theta|y) - \log q_\phi(\theta)]$$

where y is the observed data being conditioned on to produce the posterior distribution $p(\theta|y)$.

As an objective function, the ELBO makes some intuitive sense. We would like to choose q to have high expected log-probability ($E[\log p(\theta|y)]$), but also high entropy ($-E[\log q_\phi(\theta)]$) to avoid q being overfit to p's mode. When ∇_ϕELBO has a tractable closed-form solution, standard gradient descent methods apply. If we lack this, but do have a means of efficiently drawing samples from q_ϕ, the problem is still amenable to *black box variational inference* (Ranganath, Gerrish, and Blei, 2014), which is a stochastic gradient descent algorithm in which each iteration uses a Monte Carlo estimate of ∇_ϕELBO made by drawing S samples $\theta_1, \ldots, \theta_S$ from q_ϕ and computing

$$\begin{aligned}
\nabla_\phi \text{ELBO} &= \nabla_\phi E_{\theta \sim q_\phi}[\log p(\theta|y) - \log q_\phi(\theta)] \\
&= E_{\theta \sim q_\phi}[\nabla_\phi \left(\log p(\theta|y) - \log q_\phi(\theta)\right)] \\
&\approx \frac{1}{S} \sum_{i=1}^{S} \nabla_\phi \left(\log q_\phi(\theta_i)\right) \left(\log p(\theta_i|y) - \log q_\phi(\theta_i)\right)
\end{aligned}$$

The interchange of the the gradient and expectation here follows from the dominated convergence theorem (the conditions are slightly technical, but apply in typical circumstances).

This a relatively high variance estimate, so optimizing ϕ may require many iterations of stochastic gradient descent or a large number of Monte Carlo samples. Kingma and Welling (2014) propose a less general but lower variance — and thus more efficient — estimate of ∇_ϕELBO they call *stochastic gradient variational Bayes (SGVB)*.

This approach makes use of the *reparameterization trick* (Salimans and Knowles, 2013). If we can write the approximating distribution function in terms of a deterministic transformation T_ϕ like

$$q_\phi(\theta) = |\det J_{T_\phi}| q_0(T_\phi(\theta))$$

where J_{T_ϕ} is its Jacobian matrix evaluated at θ, and q_0 is some unparameterized distribution function, then a more direct estimate is available. An easy example example is the Normal distribution. If $q_{\mu,\sigma}$ is the univariate Normal distribution function, we can let q_0 be the standard normal Normal$(0, 1)$ distribution function and let $T_{\mu,\sigma}(z) = \mu + \sigma z$.

Drawing S samples $\theta_1, \ldots, \theta_S$ from q_0 we have

$$\begin{aligned}
\nabla_\phi \text{ELBO} &= E_{\theta\sim q_\phi}[\nabla_\phi \left(\log p(\theta|y) - \log q_\phi(\theta)\right)] \\
&= E_{\theta\sim q_0}[\nabla_\phi \left(\log p(T_\phi(\theta)|y) - \log q_0(\theta) - \log|\det J_{T_\phi}|\right)] \\
&= E_{\theta\sim q_0}[\nabla_\phi \left(\log p(T_\phi(\theta)|y) - \log|\det J_{T_\phi}|\right)] \\
&\approx \frac{1}{S}\sum_{i=1}^{S} \nabla_\phi \left(\log p(T_\phi(\theta_i)|y) - \log|\det J_{T_\phi}|\right)
\end{aligned}$$

The second step follows because $E_{\epsilon\sim q_0}[\log q_0(\epsilon)]$ is constant and so it can be disregarded.

Another consequence of this approach is that constants of proportionality become irrelevant and we can directly fit an approximation to a likelihood function. Returning to the problem at hand, there is no need to normalize the RNA-Seq likelihood function to find an approximately proportional q_ϕ. Suppose we are fitting a distribution family q_ϕ to the normalized likelihood $\mathcal{P}(r|x)$ where $x \in \Delta^{n-1}$ are expression values and r are the observed reads, and there is a constant of proportionality c where $\mathcal{P}(r|x) = cP(r|x)$. Then

$$\begin{aligned}
\arg\max_\phi \text{ELBO}(\phi) &= \arg\max_\phi E_{x\sim q_\phi}[\log \mathcal{P}(r|x) - \log q_\phi(x)] \\
&= \arg\max_\phi \log c + E_{x\sim q_\phi}[\log P(r|x) - \log q_\phi(x)] \\
&= \arg\max_\phi E_{x\sim q_\phi}[\log P(r|x) - \log q_\phi(x)]
\end{aligned}$$

So there is no need in practice do the integration necessary to compute the normalizing constant, we only need to be be able to evaluate and differentiate the likelihood function $P(r|x)$ and it can be fit directly. The problem is amenable to any general purpose stochastic gradient descent techniques. In the work presented here we use Adam (Kingma and Ba, 2014).

2.2 Designing an approximation

Stochastic gradient variational Bayes (SGVB) gives us a tool for fitting a parameterized distribution to a likelihood function to produce something that is approximately proportional. The success of this approach depends on choosing a family of distribution functions (q_ϕ, in our notation) that is sufficiently flexible to accurately mimic the likelihood function, but the SGVB algorithm's reliance on the reparameterization trick also limits our choices.

We need to choose a q_ϕ that can be expressed as a parameterized transformation T_ϕ of an unparameterized distribution function q_0. That is,

$$q_\phi(x) = |\det J_{T_\phi}| q_0(T_\phi(x))$$

where J_{T_ϕ} is the Jacobian matrix for T_ϕ.

The question remains how to choose q_0 and the transformation T to satisfy the following requirements

1. Samples can be efficiently drawn according to q_0.
2. T_ϕ's domain is Δ^{n-1} (i.e. a degree $n-1$ simplex)
3. T_ϕ and its gradients can be efficiently computed.
4. $\det J_{T_\phi}$ is nonzero.
5. $\log\det(J_{T_\phi})$ and its gradient can be efficiently computed.

What is "efficiently computable" of course depends on the size of the input. In typical RNA-Seq experiments in well annotated model organisms, known transcripts (n) are on the order of 10^5. The number of generated reads (m) can vary a great deal, but is commonly on the orders of 10^7 to 10^8 in bulk RNA-Seq. As a consequence, when choosing how to approximate the likelihood function, we cannot stray far from linear or log-linear time complexity in m and n without the optimization quickly becoming intractable.

In the following sections we will discuss a number of possible choice for the approximation q_ϕ before introducing our approach, the Pólya tree transformation, a new class of a transformations that will let us construct efficient and flexible approximations.

2.2.1 Common distributions on the simplex

Since RNA-Seq measures compositional, not absolute, expression, a vector x of transcript relative expressions length n is on a simplex:

$$x \in \Delta^{n-1} = \left\{ y \in \mathbb{R}^n \middle| y_i > 0 \wedge \sum_{i=1}^{n} y_i = 1 \right\}$$

Naturally, we would look first to common distribution families on the simplex. By far the most common is the Dirichlet distribution. The Dirichlet distribution, parameterized by just a length n concentration vector is not suitably flexible to be considered. It is easy to produce plausible examples which can not be satisfactorily approximated. In Figure 2.2 we show one such case. Furthermore, there is not a simple way to express a Dirichlet distribution as a deterministic transformation of a fixed distribution. We may be able to minimize the KL divergence by other means, but SGVB is inapplicable. For these two reasons, this obvious option is easily dispensed with, and we have to seek out something less common.

A more flexible distribution over arbitrary simplexes is the multivariate logistic-normal distribution (Atchison and Shen, 1980), which can be induced by transforming a normal distributed vector by an additive logistic transformation.

$$X \sim \text{LogitNormal}(\mu, \Sigma) \Leftrightarrow \begin{array}{l} X_i = \frac{\exp(Y_i)}{1+\sum_{i=1}^{n-1}\exp(Y_i)} \ \forall i < n \\ X_n = \frac{1}{1+\sum_{i=1}^{n-1}\exp(Y_i)} \end{array}$$

Figure 2.2: An example of an RNA-Seq likelihood function that cannot be reasonably approximated with a Dirichlet distribution. The leftmost plot is the true likelihood, consisting of five reads, and three transcripts, two of which are indistinguishable. The three remaining plots are densities from possible Dirichlet parameterizations (specifically $(2, 2, 1)$, $(3, 3, 1)$, and $(5, 5, 1)$). It is clear that the ambiguity along the horizontal dimension, along with the relative certitude along the vertical axis, is not a feature that can be captured by a Dirichlet distribution.

where

$$Y \sim \text{Normal}(\mu, \Sigma)$$

It is clear that the logistic-normal distribution with parameters $\phi = (\mu, \Sigma)$ can be expressed as a deterministic transformation of a fixed distribution, and so we can bring the machinery of SGVB to bear. Specifically, if the fixed distribution q_0 is a standard multivariate Normal, $z \sim q_0 = N(0, I)$, then we can form a vector $x \in \Delta^{n-1}$ that is logistic-normal distributed by destandardizing z, setting $y = \mu + Az \sim N(\mu, \Sigma)$, where A is a matrix such that $AA^T = \Sigma$, then forming x by applying the additive logistic transformation. By this process we can also efficiently generate samples from the distribution. Using a full covariance matrix would be easily become intractable, but if Σ is diagonal (or, perhaps just sparse) gradients can be computed and parameters updated efficiently.

Examples like the one in Figure 2.2 can be more closely fit with a logistic-normal distribution, but as we will see, with only a diagonal covariance matrix it is unfortunately a poor fit to the task of approximating transcript expression likelihood, though this does leave open the possibility of using some sort of sparse covariance model.

2.2.2 Transformations onto the simplex

The logistic-normal distribution is an example of a more general strategy of transforming another distribution to form a distribution over Δ^{n-1}. The transformation used there is far from the only way of mapping other spaces onto the simplex. The compositional data analysis literature has long been concerned with how best to transform compositional data out of the simplex to bring it into the reach of standard statistical methods.

A number of $\Delta^{n-1} \to \mathbb{R}^{n-1}$ have been proposed, some with inverses. When used to transform a normal distribution, for example, these can induce useful simplex distribution families. Three such bijective transformations are explored here as possible candidates to define a suitable distribution. Other transformations, like the softmax transformation ubiquitous in

deep learning, or the centered log-ratio transformation (Aitchison, 1986a), are not bijections. Transforming a normal distribution onto the simplex with softmax will induce some distribution, but the non-invertible transformation yields a Jacobian determinant of zero, blocking any easy path to computing the density function.

Different transformations of the same underlying distribution yield different distributions. This is an obvious statement, but a powerful concept in variational inference. Because of the size of RNA-Seq datasets, when choosing an approximating distribution we are limited to a number of parameters more or less linear in the number of transcripts and reads. For example, estimating a full covariance matrix is out of the question. Though a sparse covariance estimation scheme is possible, transformations offer another way to arrive at a richer set of distributions while controlling the number of parameters that need to be estimated.

In mean-field variational inference, each latent variable is treated as mutually independent in the approximation. Rezende and Mohamed (2015) introduced the idea extending mean field variational inference by transforming the pairwise independent random variables using a series of bijections rendering them dependent in complex ways, a technique called *normalizing flows*. These bijections are typically parameterized and optimized over along with the parameters of the base distribution. Varying the number of steps in the flow provides a trade-off between number of parameters and richness of the resulting family of distributions. Follow up work has proposed a number of simple families of transformations that are tractable yet expressive (Kingma et al., 2016; Tomczak and Welling, 2016).

The success of normalizing flows should instruct us to carefully consider the space of possible transformations, as they can have a drastic affect on how well we can capture the true distribution. As we will see, our choice in transformation from real numbers to the simplex gives us some control over the larger conditional independence structure of the resulting random vector. This is important to RNA-Seq where most transcripts have few reads in common, but some share many. Our priority to choose a transformation that exploits this particular form of sparsity.

In the sections that follow, we will explore a number of $R^{n-1} \mapsto \Delta^{n-1}$ transformations from the compositional data analysis literature. Afterwards an alternative transformation will be presented, and we will compare the ability of each of these transformations to fit the likelihood function.

Additive, multiplicative, and centered log-ratio transformations

Aitchison (1986a) defines a number of transformations between real numbers and the simplex. These can be used to transform the a multivariate distribution on the reals to a simplex, inducing new distribution families. We have already seen one such transformation in the context of the stan-

dard logistic-normal distribution: the *additive log-ratio transformation*

$$\text{alr}(x) = \left(\log \frac{x_i}{x_n}; i = 1, \dots, n-1 \right) \text{ for } x \in \Delta^{n-1} \tag{2.2}$$

The multivariate logistic-normal distribution can be restated in terms of the alr: if $alr(X) \sim Normal(\mu, \Sigma)$, then X is logistic-normal distributed.

The alr is typically defined, as it is here, with the divisor x_n, but the vector can be permuted to make any element the divisor. In some settings it makes sense to choose a particular element as the reference for interpretability; for example we might choose x_n to be the expression of a housekeeping gene. When searching for the best fitting approximation, there is not an obvious choice.

The second useful transformation defined by Aitchison is the *multiplicative log-ratio transformation*

$$\text{mlr}(x) = \left(\log \frac{x_i}{1 - \sum_{j=1}^{i} x_j}; i = 1, \dots, n-1 \right) \text{ for } x \in \Delta^{n-1} \tag{2.3}$$

This transformation can be best understood using a sequential stick-breaking metaphor. If a stick is broken into n pieces, and $x_1, \dots, x_n$ give the size of each piece, in proportion to the whole, then $\text{mlr}(x)$ gives the log-ratio between each piece and the remaining length of the stick, if the stick were broken one piece at a time. While the alr is dependent on what element is chosen as the denominator, the mlr is sensitive to the complete ordering of the vector x. Every permutation defines a different bijection between $\mathbb{R}^n$ and Δ^n. We will revisit this notion of stick-breaking shortly in Section 2.2.2.

Lastly, another transformation that occurs frequently in the compositional data analysis literature is the *centered log-ratio transformation*

$$\text{clr}(x) = \left(\log \frac{x_i}{g(x)}; i = 1 \dots, n-1 \right) \text{ for } x \in \Delta^{n-1} \tag{2.4}$$

where $g(x) = \left(\prod_{i=1}^{n} x_i \right)^{1/n}$ is the geometric mean of x. This transformation is not a bijection, so is not directly useful to us. [1] However, it will be useful in defining the isometric log-ratio transformation.

Aitchison geometry and the isometric log-ratio transformation

The last, and most modern transformation we will consider is the *isometric log-ratio* or *ilr* transformation (Egozcue et al., 2003), which was developed to correct some of the shortcomings of the more common alr, mlr, and clr. Specifically, while these take compositional data out of the simplex, the result is often not especially interpretable. The ilr on the other

1 To see this note that $\sum_{i=1}^{n-1} clr(x)_i = \sum_{i=1}^{n-1} \log \frac{x_i}{g(x)} = 0$, so only maps onto vector in $\mathbb{R}^n$ that sum to 0.

hand, as the name implies, is distance preserving, giving it some geometric coherence the other transformations lack.

There have been a number of distance metrics defined on the simplex, and the correct choice for this task in the past been the subject of some debate. Watson and Philip (1989) explored some possibilities, motivated by measuring variability in geological mineral composition measurements. They suggest projecting the simplex on to the surface of $(n-1)$-dimensional sphere and measuring the angle between transformed points. Aitchison (1992) pointed out a number of shortcomings in this approach, describing desirable properties of a distance metric, such as invariance to index permutation, that it failed to meet. Aitchison goes on to propose five distances that do meet these criteria, ultimately recommending the use of the sum of the squared difference between all log ratios.

$$d_\Delta(x, x') = \sum_{j=1}^{n} \sum_{i=1}^{j} \left(\log \frac{x_i}{x_j} - \log \frac{x'_i}{x'_j} \right)^2 \tag{2.5}$$

This distance metric is simple, relatively interpretable, and conforms to each of the recommended properties. Yet, despite the lucidity of Aitchison's analysis, there is no consideration of computational issues; d_Δ is $O(n^2)$ compared to the $O(n)$ angle-based distance that Watson and Philip (1989) proposed, making it intractable in high dimensionality.

Despite this shortcoming, the adoption of d_Δ as a more or less standard distance on the simplex led to the development of a more rigorous simplical geometry. Billheimer, Guttorp, and Fagan (2001) show that the simplex is a vector space with vector addition defined as the *perturbation* operator

$$x \oplus x' = \mathcal{C}(x_1 x'_1, \dots, x_n x'_n) \tag{2.6}$$

where $x, x' \in \Delta^{n-1}$ and $\mathcal{C}$ is a normalization function

$$\mathcal{C}(x) = \left(\frac{x_i}{\sum_{j=1}^{n} x_j}; i = 1, \dots, n \right) \tag{2.7}$$

and scalar multiplication defined as the *powering* operation

$$a \odot x = \mathcal{C}(x_1^a, \dots, x_n^a) \tag{2.8}$$

where $a \in \mathbb{R}$. They go further, showing it is also a Hilbert space (i.e., a complete inner product space) with the inner product

$$\langle x, x' \rangle_\Delta = \langle clr(x), clr(x') \rangle \tag{2.9}$$

This choice of inner product reproduces Aitchison's proposed distance d_Δ

$$\| x \ominus x' \|_\Delta = \sqrt{\langle x \ominus x', x \ominus x' \rangle_\Delta} = d_\Delta(x, x') \tag{2.10}$$

Here $x \ominus x' = x \oplus (-1 \odot x')$.

This idea was also developed concurrently by Pawlowsky-Glahn and Egozcue (2001). The simplical geometry they introduce has come to be referred to as *Aitchison geometry* (Pawlowsky-Glahn and Buccianti, 2011).

With Aitchison geometry in hand, some interesting results become apparent. Because Δ^{n-1} is a vector space there are, of course, simplical bases. A standard result in linear algebra (see for example Lang (1987)) tells us that given a basis $e_1, \dots, e_{n_1}$ for Δ^{n-1}, the transformation $T : \mathbb{R}^{n-1} \mapsto \Delta^{n-1}$ defined as

$$T(a) = (a_1 \odot e_1) \oplus \cdots \oplus (a_{n-1} \odot e_{n-1})$$

is an isomorphism (and thus a bijection) of $\mathbb{R}^{n-1}$ onto Δ^{n-1}. The *isometric log-ratio transformation* is defined as the inverse $ilr(x) = T^{-1}(x)$.

Egozcue et al. (2003) shows us a how to derive an explicit formula. Given a simplical basis $e_1, \dots, e_{n-1}$, define a matrix V with columns $\text{clr}(e_1)$, ... $\text{clr}(e_{n-1})$, then ilr can be written simply as

$$\text{ilr}(x) = V^T \log(x) \tag{2.11}$$

where the log is taken element-wise.

Choosing a basis for ilr

The ilr is always defined with respect to an orthonormal basis. There are infinitely many bases we might choose, and no obvious analog to the Euclidean standard basis. Egozcue and Pawlowsky-Glahn (2005) suggests a method of choosing a basis called *sequential binary partitions* with the goal of maximizing interpretability.

If $x \in \Delta^{n-1}$ is thought of as a composition of n parts, we can recursively split it into two subcompositions until we arrive at the individual $x_1, \dots, x_n$. The pattern of splits corresponds to a full binary tree with n leaves and $n-1$ internal nodes. Every possible such tree corresponds to a different basis. To see how, for internal node i, let $\text{leaves}(\text{left}(i))$, $\text{leaves}(\text{right}(i))$ be the sets of indexes of the leaf nodes in i's left and right subtrees respectively. Then assign each internal node i a value

$$y_i = \log \frac{\prod_{j \in \text{leaves}(\text{right}(i))} x_j}{\prod_{k \in \text{leaves}(\text{left}(i))} x_k} \tag{2.12}$$

We can represent this as matrix multiplication

$$y = V^T \log(x)$$

where V is an n by $n-1$ matrix where each column V_j of V corresponds to an internal node in our tree and is formed by

$$V_{ij} = \begin{cases} -1 & \text{if } i \in \text{leaves}(\text{left}(j)) \\ 1 & \text{if } i \in \text{leaves}(\text{right}(j)) \\ 0 & \text{otherwise} \end{cases}$$

The similarity to Equation 2.11 suggests that with some adjustment this could be made a valid ilr transformation. Indeed, the only adjustment necessary is to scale the non-zero entries of V to form an orthogonal matrix V' defined as

$$V'_{ij} = \begin{cases} -\sqrt{\frac{N_r}{N_r(N_r+N_\ell)}} & \text{if } i \in \text{leaves}(\text{left}(j)) \\ \sqrt{\frac{N_\ell}{N_\ell(N_r+N_\ell)}} & \text{if } i \in \text{leaves}(\text{right}(j)) \\ 0 & \text{otherwise} \end{cases}$$

where $N_\ell = |\,\text{leaves}(\text{left}(j))|$ and $N_r = |\,\text{leaves}(\text{right}(j))|$.

With this scaling, the columns $V_1, \ldots, V_{n-1}$ corresponds to $\text{clr}(e_1), \ldots, \text{clr}(e_{n-1})$ with $e_1, \ldots, e_{n-1}$ forming a simplical basis.

The advantage of choosing a basis by sequential binary partitioning is that the value associated with every internal node can be interpreted as a *balance* between two subcompositions, whereas an arbitrary basis has the nice properties ascribed by Aitchison geometry but no coherent interpretation in Euclidean space.

In some domains the data conforms to a very natural binary tree. A particularly elegant example is in the analysis of species composition in metagenomics. Any set of species conform to phylogenetic tree. Choosing the ilr basis according to this tree results in every balance variable having a natural interpretation as measuring the relative abundance of two subtrees of the phylogeny (Silverman et al., 2017).

In RNA-Seq, there is no obvious binary-tree hierarchy of transcripts but interpretability is ultimately less important than goodness of fit, so we are free to optimize over bases by exploring the space of full binary trees with n leaves.

The ilr transformation is defined in terms of an orthonormal basis, but any basis can serve to define a bijection. In fact, the additive log-ratio transformation (Equation 2.2) can be formulated as a transformation made according to an oblique basis (Egozcue and Pawlowsky-Glahn, 2005). In principle this lets us relax the constraints on the transformation and optimize over a larger space of possibilities.

Stick breaking transformations

To revisit the stick-breaking metaphor we used to describe the mlr transformation, it is often easiest to consider generating a vector $x \in \Delta^{n-1}$ by starting with a stick of length 1 and breaking it $n-1$ times in sequence.

Let $y_i \in (0,1)$ represent the proportion of the remaining stick to break off on the ith break. Then $x \in \Delta^{n-1}$ can be produced from $y \in (0,1)^{n-1}$ with

$$x_i = y_i \prod_{k=1}^{i-1}(1-y_k) = y_i \left(1 - \sum_{k=1}^{i-1} x_k\right) \quad \forall\ 1 \leq i < n \tag{2.13}$$

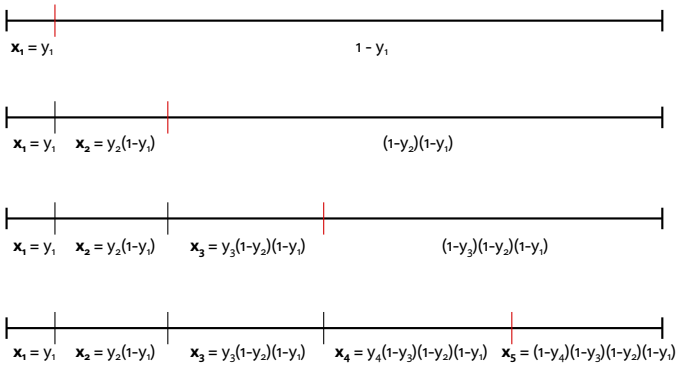

Figure 2.3: An example of the sequential stick breaking transform of a vector $y \in (0,1)^{n-1}$ to $x \in \Delta^{n-1}$, with $n = 5$ here. Each y_i can be thought of as breaking some proportion of a remaining stick off to form a new x_i.

$$x_n = \prod_{k=1}^{n-1}(1 - y_i) = 1 - \sum_{k=1}^{n-1} x_k$$

The equivalence between the product and sum form shown here may not be immediately obvious, but is easy to show by induction (Halmos, 1944), or by considering that $1 - \sum_{k=1}^{i-1} x_k$ is the length of the stick remaining after $i - 1$ breaks. This sequential stick breaking process is illustrated in Figure 2.3.

Stick-breaking metaphors like the one used to describe the mlr transformation have a long history. In an early example, Halmos (1944) explores a stick breaking distribution in which each y_i is uniformly distributed in $(0, 1)$ using the motivation of distributing a pile of gold dust to n beggars in sequence. The possibility of y_i being drawn independently from arbitrary distributions is also briefly considered. In recent literature, stick breaking occurs most commonly in descriptions of the Dirichlet process, which can be formulated as an infinite stick breaking procedure in which the breaks y_i are Beta distributed variables (Sethuraman, 1994). The resulting stick sizes are then used to weight draws from a base distribution. When $n \to \infty$ in Equation 2.13, and each y_i is i.i.d. Beta$(1, \theta)$ distributed, for some θ, the resulting distribution family is commonly denoted GEM(θ), after Griffiths, Engen, and McCloskey (see Pitman (2002)). This notion of a stick breaking prior was generalized by Ishwaran and James (2001) to, among other things, include finite stick breaking distributions.

To restate the relationship between mlr (Equation 2.3) and stick breaking: it is easy to see that the mlr transformation is equivalent to Equation 2.13 under a logit transformation (where logit$(p) = \log(p/(1-p))$).

$$\begin{aligned}
\text{logit}(y_i) &= \text{logit}\left(\frac{x_i}{1-\sum_{j=1}^{i-1} x_j}\right) \\
&= \log\left(\frac{x_i/(1-\sum_{j=1}^{i-1} x_j)}{1-x_i/(1-\sum_{j=1}^{i-1} x_j)}\right) \\
&= \log\left(\frac{x_i}{1-\sum_{j=1}^{i} x_j}\right) \\
&= \text{mlr}(x)_j
\end{aligned}$$

Unlike other aspects of Aitchison geometry, there has been little further discussion of mlr in the compositional data analysis literature. It lacks the elegant interpretation under Aitchison geometry that ilr and (to an extent) alr have, which has led to ilr being adopted as the overwhelmingly preferred methodology (Scealy and Welsh, 2014). Despite this, heterodox applications of mlr do appear in the literature (Bracci, Bull, and Grynpas, 1998; Stewart and Field, 2011).

Stick breaking in the Dirichlet distribution and Dirichlet process

A stick breaking procedure also occurs in some procedures for generating samples from a Dirichlet distribution Gelman et al. (2013b). Specifically, if $X \sim \text{Dirichlet}(\alpha)$ for $\alpha \in \mathbb{R}_+^n$ we can induce the same distribution by defining $n-1$ random variables

$$Y_i \sim \text{Beta}(\alpha_i, \sum_{j=i+1}^{n} \alpha_j) \; \forall \; 1 \leq i < n$$

then applying the stick breaking transformation from Equation 2.13 so that

$$X_i = Y_i(1 - \sum_{j=1}^{i-1} Y_i) \; \forall \; 1 \leq i < n$$

$$X_n = 1 - \sum_{i=1}^{n-1} Y_i$$

From one perspective, the Dirichlet process can be thought of as an infinite dimensional generalization of the Dirichlet distribution (Teh, 2010), so unsurprisingly it also has a stick breaking interpretation (Sethuraman, 1994).

If random probability measure G is distributed according to a Dirichlet process with parameter $\alpha \in \mathbb{R}_+$ and base distribution H (that is, $G \sim \text{DP}(\alpha, H)$), then we can equivalently define G in terms of the following infinite stick breaking procedure. For $i \in \{1, \dots\}$ let the breaking proportions be

$$\beta_i \sim \text{Beta}(1, \alpha)$$

and realizations from the base distribution H

$$\theta_i^* \sim H$$

then the stick pieces are defines as

$$\pi_i = \beta_i \prod_{j=1}^{i-1}(1-\beta_j) \tag{2.14}$$

and G assigns probabilities according to

$$G(x) = \sum_{i=1}^{\infty} \pi_i \delta_{\theta_i^*}(x)$$

where $\delta_{\theta_i^*}$ is the Kronecker delta that assigns 1 when $x = \theta_i^*$ and 0 otherwise. Note that Equation 2.14 clearly match our earlier definition of stick breaking definition in Equation 2.13. A simple way to think about this is that realizations from Dirichlet process are infinite mixture of point mass distributions, with the mixing proportions chosen from a Beta-distributed infinite stick breaking process.

Other uses of stick breaking

Prior to Aitchison's seminal work, and continuing sporadically and independently of it, a stick breaking procedure similar to mlr has appeared under the name *continuation ratios* (Fienberg, 2007). Typically continuation ratios are used in multinomial regression models with categories that have some inherent ordering, for example, ordinal categorical ratings of the effectiveness of analgesic drugs (Cox, 1988).

Similar transformations also arise in the probabilistic inference literature. Betancourt (2012) presents a method of reparameterizing the support of the Dirichlet distribution to more easily generate samples using Hamiltonian Monte Carlo. A related idea is explored by Khan et al. (2012) in the context of latent Gaussian models in which a stick breaking transformation of latent Gaussian variables is proposed as a more tractable and convenient alternative to the softmax function. The probabilistic programming system Stan (Carpenter et al., 2016) implements an essentially identical model as a general purpose variational approximation to distributions on the simplex as part of its Automatic Differentiation Variational Inference approach (Kucukelbir et al., 2017).

The model as used in Stan uses $n-1$ dimensional multivariate normal vector $Z \sim \text{Normal}(\mu, \Sigma)$ first transformed to lie in $(0,1)^{n-1}$ by separately applying the logistic function to each element yielding a vector y where

$$y_i = \text{logistic}(Z_i) = \frac{1}{1+\exp(-Z_i)} \;\forall\; 1 \leq i < n$$

The implementation in Stan adds centering constants to the y_is, which we exclude here for simplicity. After the logistic transformation a transformation like Equation 2.13 is used to induce a distribution over Δ^{n-1}. In

Stan this transformation and resulting distribution family is presented as something of a black box intended as a general purpose mechanism for capturing distributions over compositional data.

Khan et al. (2012) brings up an important issue which is ignored in Stan's use of stick breaking. They point out that the model represents a kind of decision boundary between every category i and the $n - i$ categories that follow it in the process, so a particular ordering may fit the data poorly if no such boundary naturally exists. Zhang and Zhou (2017) take up this issue in a more serious way, demonstrating classification problems with as few as three categories that show dramatic differences in performance depending the permutation of those categories in the stick breaking process. They go on to propose models in which category permutations are inferred along with regression coefficients when performing multinomial logistic regression.

Our problem is not dissimilar: a different stick breaking order will induce a different distribution over the simplex, but in choosing an approximating distribution for RNA-Seq likelihood, there is not an obvious natural ordering of transcripts. There are $n!$ orderings (in which n tends to be on the order of 10^5 for the human or mouse genome). Zhang and Zhou (2017) argue that this space of orderings is well behaved (i.e. in the sense that similar permutations will have similar performance) and can thus be effectively explored by MCMC without having to consider anything like the full set of permutations. However, the most categories considered in their benchmarks is 11. Results from exploring a space of $11!$ permutations may not generalize to exploring a space of $10^5!$ permutations.

The second key insight that is sometimes neglected is that there are other ways of breaking a stick. This is an idea we will explore in the next section.

2.2.3 Hierarchical stick-breaking

Stick breaking transformations are defined with respect to a permutation or order in which the breaks occur. But rather than breaking pieces off the stick and setting them aside, we might keep and recursively break both of the resulting pieces. This can be thought of as *hierarchical stick breaking*, as opposed to common *sequential stick breaking*. To define a transformation onto Δ^{n-1}, we must always end up with n pieces, so $n-1$ total breaks must still be made. Under these restrictions, breaks in a hierarchical stick-breaking scheme must occur according to a full binary tree with n leaves (i.e., where every node is either a leaf or has two children). This idea is illustrated in Figure 2.4.

While sequential stick breaking is defined in terms of an index permutation, hierarchical stick breaking is in terms of a tree, specifically a full binary tree with n leaves, in addition to a label permutation. Sequential stick breaking is in fact a special case of hierarchical stick breaking, occurring when the tree is "degenerate", in the sense that every internal node

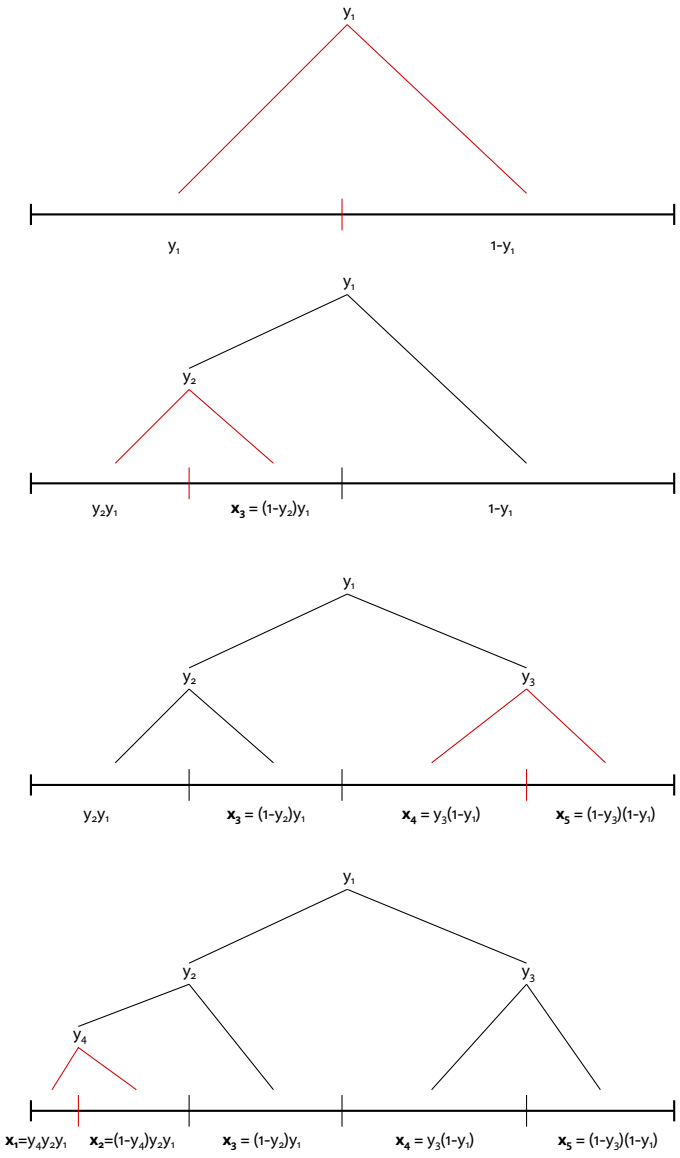

Figure 2.4: Hierarchical stick breaking is another means of transforming a vector in $(0,1)^{n-1}$ to a simplex Δ^{n-1}. It generalizes sequential stick breaking (Figure 2.3) by varying the order in which the breaks occur. By ordering breaks in a particular way, certain covariance structures can be more accurately captured.

except for one has exactly one leaf node child, forming essentially a linear list as shown in Figure 2.5.

This notion of hierarchical stick-breaking bears some resemblance to the hierarchical softmax transformation (Goodman, 2001), a technique used in some natural language models, but differs critically in that hierarchical softmax is not bijective and is used purely as a means of accelerating inference. More closely related are Pólya tree distributions which were introduced in Ferguson (1974) as generalization of the Dirichlet process which he described first in Ferguson (1973). A more detailed analysis was taken up by Lavine (1992), Lavine (1994), and Mauldin, Sudderth, and Williams (1992)

A Pólya tree distribution in a nonparametric model where realizations are continuous probability measures over some space Ω. They are defined in terms of a binary tree representing an infinite recursive partitioning of Ω. We can represent a path through the tree as binary string where the bit at the ith position represents the branch taken at that level. The sets making up the recursive partition can then be written as $B_\emptyset, B_0, B_1, B_{00}, B_{01}, \ldots$, where for any binary string s, $B_s = B_{s0} \cup B_{s1}$ and $B_{s0} \cap B_{s1} = \emptyset$.

The the distribution works by allocating probability mass at each branch of the tree according to independent Beta distributed variables $Y_\emptyset, Y_0, Y_1, Y_{00}, Y_{01}, \ldots$, each with a distribution

$$Y_s \sim \text{Beta}(\alpha_{s0}, \alpha_{s1})$$

A random probability measure $\mathcal{P}$ distributed according to the Pólya tree distribution then assigns probability according to

$$\mathcal{P}(B_{\epsilon_1,\ldots,\epsilon_m}) = \left(\prod_{i=1;\epsilon_i=0}^{m} Y_{\epsilon_1 \ldots \epsilon_{i-1}} \right) \left(\prod_{i=1;\epsilon_i=1}^{m} (1 - Y_{\epsilon_1 \ldots \epsilon_{i-1}}) \right)$$

Figure 2.6 gives an illustration of the first few levels of a Pólya tree distribution where $\Omega = (0, 1]$. As with Dirichlet processes, they are conceptually infinite, but in practice limited to some finite to depth for practical applications.

Unlike the other notions of stick breaking, the breaks here are not random but selected ahead of time. The random process follows these infinite deterministic breaks to allocate probability mass to smaller and smaller partitions yielding a continuous distribution in the limit. The necessity of determining the breaks in advance is cumbersome, which prompted the development of randomized Pólya tree distributions (Paddock et al., 2003), which use a random partitioning scheme resembling the hierarchical stick breaking scheme described in the previous section.

To summarize, Pólya tree distributions generalize Dirichlet processes. The Dirichlet process and the Dirichlet distribution can both be understood in terms of sequential stick breaking, while the Pólya tree distribution can be understood in terms of hierarchical stick breaking. Appealing simply to an aesthetic desire for symmetry, we might suspect there is a

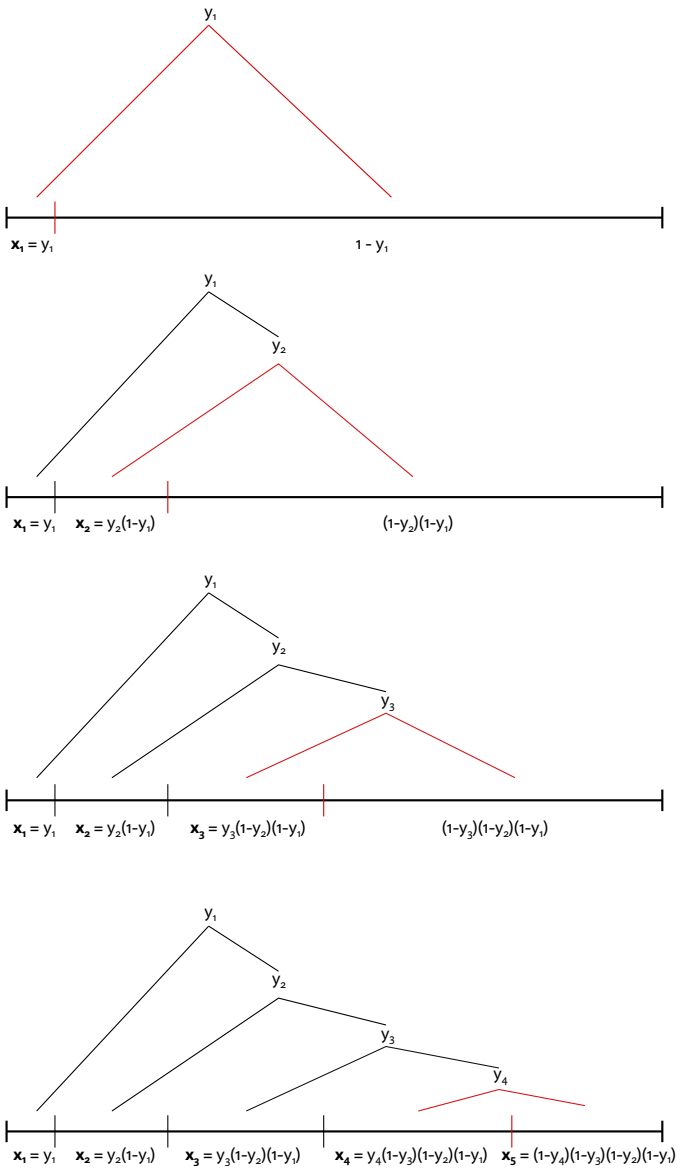

Figure 2.5: Sequential stick breaking can defined as special case of hierarchical stick breaking in which a degenerate binary tree is used (i.e., one where every internal node has exactly one leaf, except for the final deepest internal node). This correspondence is illustrated with $n = 5$ here.

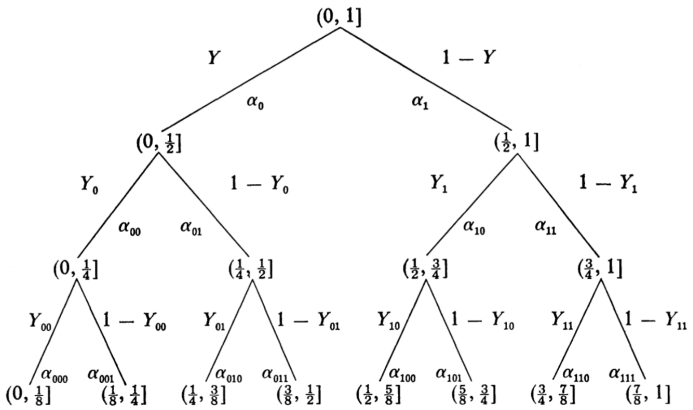

Figure 2.6: An example from Ferguson (1974) of the first three levels of a Pólya tree distribution over random probability measures on $(0, 1]$. The space is recursively partitioned in half and probability mass distributed among the partitions according to random Beta variables $Y, Y_0, Y_1, Y_{00}, Y_{01}, \ldots$., where the subscripts are binary strings indicating paths through the tree (0 for the left fork, 1 for the right).

generalization of the Dirichlet distribution that can be described in terms of hierarchical stick breaking. As we will see, there is such a distribution family.

The Pólya tree transformation

We saw that the Dirichlet distribution can be described as a sequential stick breaking transformation applied to Beta distributed random variables with specific parameterizations. Applying similar stick breaking transformations to less restricted family of distributions yields a broader family of distributions on the simplex. Inspired by the Pólya tree distribution, we can also generalize the stick breaking transformation in Equation 2.13 to consider the space of hierarchical stick breaking transformations. We call this more general class of transformations *Pólya tree transformations*.

The Pólya tree transformation is best represented as a full binary tree, with $2n - 1$ nodes, n of which are leaves. To simplify notation somewhat, assume these are assigned indices so that the root node is labeled 1, internal nodes $1, \ldots, n-1$, and leaf nodes $n, \ldots, 2n-1$. Additionally, we assume internal nodes are numbered so it has larger index than any of its ancestors, for example, according to a pre-order traversal. Intuitively, we can think of the ith node as representing the ith break in a stick-breaking process.

The tree can then be defined by functions giving an internal node's left and right children respectively.

left(i): index of node i's left child

right(i): index of node i's right child

The transformation T from $(u_1, y) \in (\mathbb{R}_+, (0,1)^{n-1})$ to $x \in \mathbb{R}_+^n$ is defined in terms of *intermediate values* $u_1, \dots, u_{2n-1}$ for each node, which have the following relation,

$$u_{\text{left}(i)} = y_i u_i \tag{2.15}$$
$$u_{\text{right}(i)} = (1 - y_i) u_i \tag{2.16}$$

The result of the transformation is then simply the intermediate values from the leaf nodes:

$$x_i = u_{i+n-1}, \text{ for } i = 1, \dots, n \tag{2.17}$$

In our use case, we are operating on the unit simplex, in which case $u_1 = 1$. In these instances we will leave the $u_1 = 1$ implicit and write the transformation as $T : (0,1)^{n-1} \mapsto \Delta^{n-1}$.

When implemented, u_i values are computed by traversing the tree with any top-down traversal from the root. In the stick-breaking metaphor, intermediate values can be thought of as sizes of intermediate sticks after some number of breaks are performed. The inverse transformation can be computed by traversing the tree up from its leaves, as is done in a post-order traversal.

If we choose a less large family of distributions to govern the breaks, and select a transformation from this larger space of stick breaking transformations, we have the opportunity of finding a better approximation for the RNA-Seq likelihood function, not to mention many other problems dealing with compositional data..

2.2.4 Some mathematical properties of the Pólya tree transformation

In this section we will prove a number of useful properties of Pólya tree transformations. Most critically, deriving the Jacobian determinant (Theorems 2.2.4 and 2.2.5) and showing that Dirichlet distributions can always be written in terms of any Pólya tree transformation (Theorems 2.2.11 and 2.2.13).

Lemma 2.2.1. *Let $x = T(y, u_1)$ be the output for a Pólya tree transformation with inputs $y \in \mathbb{R}_+^{n-1}, u_1 \in \mathbb{R}_+$, and let u be the $n-1$ intermediate values.*

If leaves(i) is the set of indexes of leaf nodes in i's subtree, then

$$u_i = \sum_{j \in \text{leaves}(i)} u_j = \sum_{j \in \text{leaves}(i)} x_{j-n+1}$$

Proof. This simply says that each intermediate stick piece is the same length as the sum of the parts it is ultimately broken into, and is easy to see through induction.

Let i be the index of a leaf node. The subtree's only leaf is i, so the theorem is trivially true,

$$u_i = x_{i-n+1} = \sum_{j \in \text{leaves}(i)} x_{j-n+1}$$

Next consider any internal node i where the theorem is true of its children. That is, where

$$u_{\text{left}(i)} = \sum_{j \in \text{leaves}(\text{left}(i))} x_{j-n+1}$$
$$u_{\text{right}(i)} = \sum_{j \in \text{leaves}(\text{right}(i))} x_{j-n+1}$$

Then,

$$\begin{aligned} u_i &= y_i u_i + (1 - y_i) u_i \\ &= u_{\text{left}(i)} + u_{\text{right}(i)} \\ &= \sum_{j \in \text{leaves}(\text{left}(i))} x_{j-n+1} + \sum_{j \in \text{leaves}(\text{right}(i))} x_{j-n+1} \\ &= \sum_{j \in \text{leaves}(i)} x_{j-n+1} \end{aligned}$$

By mathematical induction, the theorem is true of all $i = 1, \ldots, 2n-1$. □

Theorem 2.2.2. *$T : (\mathbb{R}_+, (0,1)^{n-1}) \mapsto \mathbb{R}_+^n$ is a bijection.*

Proof. For any $u_1, u_1' \in \mathbb{R}_+$, and $y, y' \in (0,1)^{n-1}$, where $(u_1, y) \neq (u_1', y')$, let $x = T((u_1, y))$ and $x' = T((u_1', y'))$.

Suppose $u_1 \neq u_1'$, then by Lemma 2.2.1

$$\sum_{j=1}^{n} x_j = u_1 \neq u_1' = \sum_{j=1}^{n} x_j'$$

and therefore $x \neq x'$.

Suppose $u_1 = u_1'$, and let i be the index of the first $y_i \neq y_i'$. Since nodes are numbered so that ancestors have smaller indices than their children, for every ancestor j of i, $y_j = y_j'$, therefore $u_i = u_i'$. All intermediate values are positive (this follows from u_1 being positive and elements of y residing on an open $(0,1)$ interval), therefore

$$u_{\text{left}(i)} = y_i u_i \neq y_i' u_i = y_i' u_i' = u'_{\text{left}(i)}$$

And by Lemma 2.2.1

$$\sum_{j \in \text{leaves}(\text{left}(i))} x_{j-n+1} \neq \sum_{j \in \text{leaves}(\text{left}(i))} x'_{j-n+1}$$

so $x \neq x'$.

As $x \neq x'$ in either case, T is injective.

To show that T is surjective as well, consider any $x \in \mathbb{R}^n_+$.

Equations 2.15 and 2.16 can be rewritten as

$$u_i = u_{\text{left}(i)} + u_{\text{right}(i)} \tag{2.18}$$

$$y_i = \frac{u_{\text{left}(i)}}{u_{\text{left}(i)} + u_{\text{right}(i)}} \tag{2.19}$$

First, following Equation 2.17, let $u_{i+n-1} = x_i$ for $i = 1, \ldots, n$, thus defining leaf node intermediate values. The remaining intermediate values u_i, for $i = 1, \ldots, n-1$ are then defined according to Equation 2.18, and y_i for $i = 1, \ldots, n$ according to Equation 2.19.

Importantly, since elements of x are positive, it follows that all intermediate values are positive, so the denominator of Equation 2.19 is never zero and $y_i \in (0, 1)$.

As Equations 2.15, 2.16 are satisfied if Equations 2.18, 2.19 are, and u_1, y are defined by the latter, it follows that $T((u_1, y)) = x$ and T is surjective, and thus bijective. □

Corollary 2.2.3. *If $u_1 = 1$ is fixed, then the resulting transformation is a bijection between $(0, 1)^{n-1}$ and $\Delta^{n-1} = \{x \in \mathbb{R}^n_+ | \sum_{i=1}^n x_i = 1\}$.*

Proof. A restriction of a bijection remains a bijection so we need only show that the image is Δ^{n-1}. It follows directly from Lemma 2.2.1 that the codomain of the restriction is Δ^{n-1}. Furthermore, any $x \in \Delta^{n-1}$ has a (u_1, y) where $T((u_1, y)) = x$, and again by Lemma 2.2.1, $u_1 = 1$, so (u_1, y) is in the restricted domain, and thus the image is Δ^{n-1}. □

For the purposes of approximating the RNA-Seq likelihood function, we only care about this special case mapping onto the simplex.

For the approximation to be useful for computing probability densities of transformed variables, the determinant of the Jacobian must also be efficiently computable, which we show here.

Theorem 2.2.4. *Let $(u_1, y) \in (\mathbb{R}_+, (0, 1)^{n-1})$, and J_T be the Jacobian matrix of $T((u_1, y))$. Further, let u_i for $i = 2, \ldots, 2n-1$ be the intermediate values as defined in Equations 2.15 and 2.16. Then*

$$|\det(J_T)| = \prod_{i=1}^{n-1} u_i$$

Or in words, the absolute value of the Jacobian determinant is simply the product of the internal node intermediate values.

Proof. One way to derive the determinant for J_T is to separate the transformation T into the composition of a number of simpler transformations. The stick breaking metaphor suggests as natural decomposition: each break

can be treated as its own transformation. If T is thought of as a sequence of $n-1$ breaks, it can be represented as

$$T = T_{n-1} \circ ... \circ T_1$$

where T_i is the ith break and has the associated break proportion y_i. This would let us write

$$|\det(J_T)| = \prod_{i=1}^{n-1} |\det J_{T_i}|$$

where J_{T_i} is the Jacobian matrix for T_i.

The single break transformation

$$T_i : (\mathbb{R}_+^i, (0,1)^{n-i}) \mapsto (\mathbb{R}_+^{i+1}, (0,1)^{n-i-1})$$

can be thought of as taking i sticks lengths, and $n-i$ remaining breaking proportions, and breaking one of the sticks according to the first remaining proportion. Equivalently, T_i reflects taking one step in the tree traversal, computing $u_{\text{left}(i)}, u_{\text{right}(i)}$ from u_i, y_i, and then discarding these latter two values.

Incidentally, because only two values are replaced, it is not hard to see that each of these transformations is also a bijection between different n dimensional spaces.

To give a concrete example of T being decomposed into single break transformations, consider a balanced tree with $n = 4$ leaves, and the input $u_1 = 100, y = (0.2, 0.6, 0.9)$. In the following table, each row gives the input to T_i, and the subsequent row its output. The final result is a vector in $\mathbb{R}_+^n$.

i	$\mathbb{R}_+^i$	$(0,1)^{n-i}$
1	100	$0.2, 0.6, 0.9$
2	$20, 80$	$0.6, 0.9$
3	$12, 8, 80$	0.9
	$12, 8, 72, 8$	

Because T_i only replaces u_i, y_i with $u_{\text{left}(i)}, u_{\text{right}(i)}$, the Jacobian matrix is the n by n identity matrix except for a 2 by 2 submatrix so that,

$$\begin{aligned}
|\det(J_{T_i})| &= \begin{Vmatrix} \frac{\partial u_{\text{left}(i)}}{\partial u_i} & \frac{\partial u_{\text{left}(i)}}{\partial y_i} \\ \frac{\partial u_{\text{right}(i)}}{\partial u_i} & \frac{\partial u_{\text{right}(i)}}{\partial y_i} \end{Vmatrix} \\
&= \begin{Vmatrix} y_i & u_i \\ 1-y_i & -u_i \end{Vmatrix} \\
&= |-y_i u_i + y_i u_i - u_i| \\
&= u_i
\end{aligned}$$

And therefore

$$|\det(J_T)| = \prod_{i=1}^{n-1} |\det J_{T_i}| = \prod_{i=1}^{n-1} u_i$$

□

Writing the Jacobian determinant in terms of intermediate values yields a simple, conveniently computed form, but we can also write the determinant in terms of the input to the transformation. This will come in handy in Section 4.2.1.

Corollary 2.2.5 (Alt. Jacobian determinant). *Let* $(u_1, y) \in (\mathbb{R}_+, (0,1)^{n-1})$, *and* J_T *be the Jacobian matrix of* $T(u_1, y)$. *Then*

$$|\det(J_T)| = u_1^{n-1} \prod_{i=1}^{n-1} y_i^{|\text{internal}(\text{left}(i))|} (1-y_i)^{|\text{internal}(\text{right}(i))|}$$

where $|\text{internal}(\text{left}(i))|$ *and* $|\text{internal}(\text{right}(i))|$ *give the number of internal nodes in node* i*'s left and right subtrees, respectively.*

Proof. Every intermediate value for $i > 1$, with a parent node j is either $u_i = u_j y_j$, if i is j's left child, or $u_i = u_j(1-y_j)$ if it is j's right child. Carry this forward on u_j, we can see that u_i is determined by its path to the root and is the product of u_1 and y_k or $(1-y_k)$ for every ancestor k.

To be specific, let ℓ, r be indicator function defined as

$$\ell(i,j) = \begin{cases} 1 & \text{if } i \text{ is in } j\text{'s left subtree} \\ 0 & \text{otherwise} \end{cases}$$

$$r(i,j) = \begin{cases} 1 & \text{if } i \text{ is in } j\text{'s right subtree} \\ 0 & \text{otherwise} \end{cases}$$

Then

$$u_i = u_1 \prod_{j=1}^{n-1} y_j^{\ell(i,j)} (1-y_j)^{r(i,j)}$$

The Jacobian determinant is then

$$\begin{aligned} |\det(J_T)| &= \prod_{j=1}^{n-1} u_j && \text{By Theorem 2.2.4} \\ &= u_1^{n-1} \prod_{i=1}^{n-1} \prod_{j=1}^{n-1} y_j^{\ell(i,j)} (1-y_j)^{r(i,j)} \end{aligned}$$

Now, $\ell(i,j) = 1$ for exactly $|\text{internal}(\text{left}(i))|$ values of $j \in \{1, \ldots, n-1\}$, and similarly $r(i,j) = 1$ is that case $|\text{internal}(\text{right}(i))|$ times. Therefore we can rewrite this product as

$$|\det(J_T)| = u_1^{n-1} \prod_{i=1}^{n-1} y_i^{|\text{internal}(\text{left}(i))|} (1-y_i)^{|\text{internal}(\text{right}(i))|}$$

□

A Dirichlet process is a special case of a Pólya tree distribution (Ferguson, 1974). Here we show a variation of this fact, that a Dirichlet distribution can be represented using any Pólya tree transformation, regardless of topology, if appropriately applied to Beta distributed random variables.

This is important to RNA-Seq because a multinomial distribution is often assumed when there is thought to be no read ambiguity. A multinomial distribution renormalized to form a distribution over probability vectors is a Dirichlet distribution. So this theorem says that if Pólya tree transformed Beta random variables are fit to the multinomial likelihood, nothing is lost except for a constant of proportionality. If the underlying distributions can reasonably mimic Beta distributions, then we can expect the fit to the multinomial to be very good.

Definition 2.2.6 (Hierarchical Beta distribution family). *For $n > 1$, let $T : (0,1)^{n-1} \mapsto \Delta^{n-1}$ be a Pólya tree transformation and $\alpha \in \mathbb{R}_+^n$.*

For $i \in \{1, \dots, n-1\}$, let

$$a_i = \sum_{j \in \mathit{leaves}(i)} \alpha_{j-n+1}$$

where leaves corresponds to T's tree topology. Put in words, if we associate α_j with the jth leaf node (which is node $j+n-1$ in our numbering scheme), a_i is the sum of α entries corresponding to the leaf node descendants of internal node i.

A random variable X is HierarchicalBeta(T, α) distributed iff $X \sim T(Y)$, where $Y = (Y_1, \dots, Y_{n-1})$ are independent random variables with

$$Y_i \sim \mathit{Beta}(a_{\mathit{left}(i)}, a_{\mathit{right}(i)})$$

Hierarchical Beta is simply the family of distributions induced by applying a Pólya tree transformation to Beta distributed stick breaking proportions with a specifically chosen parameterization scheme. Recall that a Pólya tree distribution is a (potentially infinite) hierarchical stick breaking process (Lavine, 1992; Lavine, 1994; Mauldin, Sudderth, and Williams, 1992) with Beta distributed breaks. Hierarchical Beta distributions are thus a specific subset of Pólya tree distributions.

Definition 2.2.7. *A* terminal stick break *transformation*

$$B_{ijk} : (\mathbb{R}_+^{n-1}, (0,1)) \mapsto \mathbb{R}_+^n$$

for $1 \leq i \leq n-1$ and $1 \leq j < k \leq n$ is defined by

$$x' = B_{ijk}(x, y)$$

where x' is formed from x by removing x_i and inserting yx_i and $(1-y)x_i$ such that they are x'_j and x'_k, respectively.

This is a bijection, and the inverse B_{ijk}^{-1}, we call an initial stick mend*. It removes x'_j and x'_k and inserts $x_i = x'_j + x'_k$, and additionally yields $y = \frac{x'_j}{x'_j + x'_k}$.*

In the proof of Theorem 2.2.4 we described how a Pólya tree transformation can be thought of as a series of single stick break transformations, which we used to derive the Jacobian determinant. In most trees there is a choice in the order of single stick breaks, equivalent to choosing one of multiple possible pre-order traversals of the tree, each of which results in a different representation of the same Pólya tree transformation. Regardless of the ordering, there is always some final break, and this break is a terminal stick break B_{ijk} for some i, j, k.

Some notation will be useful to decompose and build up Pólya tree distributions.

Definition 2.2.8. *If*

$$T_i : (\mathbb{R}_+^j, (0,1)^k) \mapsto (\mathbb{R}_+^{j+1}, (0,1)^{k-1})$$

is a single stick break transformation, as described in the proof for Theorem 2.2.4, which produces one more stick length by using the first breaking proportion, and if $k > 1$, we can drop the last unused breaking proportion to form a new transformation we denote

$$T_i^- : (\mathbb{R}_+^j, (0,1)^{k-1}) \mapsto (\mathbb{R}_+^{j+1}, (0,1)^{k-2})$$

Similarly, we can add another unused breaking proportion if $k \geq 0$, which we'll write as

$$T_i^+ : (\mathbb{R}_+^j, (0,1)^{k+1}) \mapsto (\mathbb{R}_+^{j+1}, (0,1)^{k})$$

This notation lets us peel off the final break of a Pólya tree transformation. As we saw, the last break is a terminal stick break B_{ijk} for some i, j, k. So any Pólya tree transformation T with $n > 2$ can be written as $B_{ijk} \circ S$, for a transformation

$$S : (\mathbb{R}_+, (0,1)^{n-1}) \mapsto (\mathbb{R}_+^{n-1}, (0,1))$$

If we dispense with S's unused breaking proportion, we have

$$S^- : (\mathbb{R}_+, (0,1)^{n-2}) \mapsto \mathbb{R}_+^{n-1}$$

which is itself a Pólya tree transformation. Similarly, we can add another breaking proportion with the T^+ notation to extend a Pólya tree transformation.

Lemma 2.2.9 (Hierarchical Beta breaking). *Let $X \sim \mathit{HierarchicalBeta}(T, \alpha)$, where T is a Pólya tree transformation. Let $\alpha \in \mathbb{R}_+^n$, and $Z \sim \mathit{Beta}(a, b)$ where $a + b = \alpha_i$ for some $1 \leq i \leq n$, and let B_{ijk} be a terminal stick break, then*

$$B_{ijk}(X, Z) \sim \mathit{HierarchicalBeta}(B_{ijk} \circ T^+, \alpha')$$

where α' is formed by removing α_i from α and inserting a and b such that $\alpha'_j = a$ and $\alpha'_k = b$.

Proof. The original Hierarchical Beta distribution is defined by the Pólya tree transformation T applied to $n-1$ Beta distributed random variables $Y_1, \dots, Y_{n-1}$, parameterized with α as they are in Definition 2.2.6.

Applying the terminal stick break B_{ijk} simply does one more break according to one more Beta distributed random variable Z. So the resulting distribution is defined by the the Pólya tree transformation $B_{ijk} \circ T^+$ applied to Beta random variables $Y' = (Y_1, \dots, Y_{n-1}, Z)$. It only remains to be seen that Y' follows the Hierarchical Beta parameterizations with α'.

Z is defined to trivially obey these rules, and restricting $a + b = \alpha_i$ and inserting a, b into α to form α' preserves the parameterization of the rest of the distribution, so we can say $B_{ijk}(X, Z) \sim \text{HierarchicalBeta}(B_{ijk} \circ T^+, \alpha')$. □

Lemma 2.2.10 (Dirichlet mending). *If $X \sim$ Dirichlet(α) and B_{ijk}^{-1} is an initial stick mend, then*

$$B_{ijk}^{-1}(X) \sim (\textit{Dirichlet}(\alpha'), \textit{Beta}(\alpha_j, \alpha_k))$$

where α' is formed from α by removing α_j and α_k and inserting their sum so that $\alpha'_i = \alpha_j + \alpha_k$.

Proof. This follows from two properties of the Dirichlet distribution: subcompositional invariance and amalgamation invariance (see, for example, Pawlowsky-Glahn, Egozcue, and Tolosana-Delgado (2015) p. 121). If $x \sim \text{Dirichlet}(\alpha)$, the former property says, in the narrow case needed here, that for any $1 \leq k_1 < k_2 \leq n$,

$$\left(\frac{x_{k_1}}{x_{k_1} + x_{k_2}}, \frac{x_{k_2}}{x_{k_1} + x_{k_2}} \right) \sim \text{Dirichlet}(\alpha_{k_1}, \alpha_{k_2})$$

or, equivalently,

$$\frac{x_{k_1}}{x_{k_1} + x_{k_2}} \sim \text{Beta}(\alpha_{k_1}, \alpha_{k_2})$$

The latter property says that if $\{1, \dots, n\}$ is partitioned into $k > 1$ nonempty sets $I_1, \dots, I_k$, then

$$\left(\sum_{i \in I_1} x_i, \dots, \sum_{i \in I_k} x_i \right) \sim \text{Dirichlet} \left(\sum_{i \in I_1} \alpha_i, \dots, \sum_{i \in I_k} \alpha_i \right)$$

For any $X \sim \text{Dirichlet}(\alpha)$, and initial stick mend B_{ijk}^{-1}, let $(X', Y) = B_{ijk}^{-1}(X)$.

From the definition of an initial stick mend, $Y = \frac{X_j}{X_j + X_k}$. The subcompositional invariance property then gives us

$$Y \sim \text{Beta}(\alpha_j, \alpha_k)$$

Since α' is formed by by summing two elements of α, by the amalgamation invariance property,

$$X' \sim \text{Dirichlet}(\alpha')$$

□

Theorem 2.2.11. *For any Pólya tree transformation* $T : (0,1)^{n-1} \mapsto \Delta^{n-1}$ *where* $n \geq 2$ *and any* $\alpha \in \mathbb{R}_+^n$

$$\textit{Dirichlet}(\alpha) = \textit{HierarchicalBeta}(T, \alpha)$$

In other words, the Dirichlet distribution family and the Hierarchical Beta distribution family are exactly the same, regardless of the tree topology used for the Hierarchical Beta.

Proof. The proof will proceed by induction over n.

(Base case) Let $n = 2$. For any $\alpha \in \mathbb{R}_+^2$ let $Y \sim \text{Beta}(\alpha_1, \alpha_2)$. Note that, because Dirichlet generalizes Beta, $(Y, (1-Y)) \sim \text{Dirichlet}(\alpha)$.

There is only one Pólya tree transformation $T : (0,1) \mapsto \Delta^1$, which takes the form $T(Y) = (Y, 1-Y)$ and by Definition 2.2.7, $T(Y) = (Y, 1-Y) \sim \text{HierarchicalBeta}(T, \alpha)$, so we have

$$\text{Dirichlet}(\alpha) = \text{HierarchicalBeta}(T, \alpha)$$

(Inductive step) Now assume the proposition is true for some $n \geq 2$.

For any $\alpha \in \mathbb{R}_+^{n+1}$ and any Pólya tree transformation $T : (0,1)^n \mapsto \Delta^n$, we can decompose T as $T = B_{ijk} \circ S$ for transformations S and B_{ijk} where B_{ijk} is a terminal stick break for some indexes i, j, k.

By Lemma 2.2.10, if $X \sim \text{Dirichlet}(\alpha)$ then

$$(X', Y) = B_{ijk}^{-1}(X) \sim (\text{Dirichlet}(\alpha'), \text{Beta}(\alpha_i, \alpha_j))$$

where α' is defined as it is in Lemma 2.2.10.

By the inductive hypothesis, X' is also Hierarchical Beta distributed for any Pólya tree transformation onto Δ^{n-1}, and since S^- is a Pólya tree transformation onto Δ^{n-1},

$$\text{Dirichlet}(\alpha') = \text{HierarchicalBeta}(S^-, \alpha')$$

Because B_{ijk} and B_{ijk}^{-1} are inverses,

$$B_{ijk}(X', Y) \sim \text{Dirichlet}(\alpha)$$

and by Lemma 2.2.9,

$$B_{ijk}(X', Y) \sim \text{HierarchicalBeta}(T, \alpha)$$

and so we have

$$\text{Dirichlet}(\alpha) = \text{HierarchicalBeta}(T, \alpha)$$

And by induction, this equality holds for all such α, T and any $n \geq 2$. □

This result can be strengthened to say that not only can any Dirichlet distribution be constructed using any Pólya tree transformation of the same dimensionality, but the Hierarchical Beta construction is the only way to do so. First, we need a simple result about random variables under bijections.

Lemma 2.2.12. *For any random variables $Y = (Y_1, \dots, Y_n)$ and $Y' = (Y'_1, \dots, Y'_n)$, and bijection T, if $T(Y)$ and $T(Y')$ are identically distributed, Y and Y' must also be identically distributed.*

Proof. To show the contrapositive, suppose $T(Y)$ and $T(Y')$ are not identically distributed. There is then some y where $P(T(Y) = y) \neq P(T(Y') = y)$. Since

$$P(T(Y) = y) = |\det J_T| P(Y = T^{-1}(y))$$

and

$$P(T(Y') = y) = |\det J_T| P(Y' = T^{-1}(y))$$

where J_T is the Jacobian matrix for T, it follows that

$$|\det J_T| P(Y = T^{-1}(y)) \neq |\det J_T| P(Y' = T^{-1}(y))$$
$$P(Y = T^{-1}(y)) \neq P(Y' = T^{-1}(y))$$

Therefore Y and Y' are not identically distributed either. □

Next, we can strengthen Theorem 2.2.11.

Corollary 2.2.13. *For any Pólya tree transformation T, $\alpha \in \mathbb{R}^n_+$, and random variables $Y = (Y_1, \dots, Y_{n-1})$. If*

$$T(Y) \sim \textit{Dirichlet}(\alpha)$$

then

$$Y_i \sim \textit{Beta}(a_{\textit{left}(i)}, a_{\textit{right}(i)})$$

for $i \in \{1, \dots, n-1\}$ where a is defined as it is in Definition 2.2.6.

Proof. For any Pólya tree transformation T and $\alpha \in \mathbb{R}^n$, we know from Theorem 2.2.11, that if $Y = (Y_1, \dots, Y_n)$ are Beta distributed with the Hierarchical Beta parameterization, then

$$T(Y) \sim \text{Dirichlet}(\alpha)$$

From Lemma 2.2.12, any other random variables $Y' = (Y'_1, \dots, Y'_n)$ where $T(Y') \sim \text{Dirichlet}(\alpha)$ must be identically distributed to Y, and so must also be Beta distributed with the same parameterization. □

To summarize these results: any Dirichlet distribution can be constructed using any Pólya tree transformation of the same dimensionality, and furthermore this construction is unique.

ilr and the Pólya tree transformation

It is worth considering how Pólya tree transformations relate to the isometric log ratio transformation under the sequential binary partitions scheme developed by (Egozcue and Pawlowsky-Glahn, 2005). There too the transformation is a kind of hierarchical partitioning, which can be represented as a binary tree. In fact, they share a similar form.

Consider an inverse Pólya tree transformation from the unit simplex $T^{-1} : \Delta^{n-1} \mapsto (0,1)^{n-1}$. If we compose this with an element-wise logit function, we have a transformation $\text{logit} \circ T^{-1} : \Delta^{n-1} \mapsto \mathbb{R}^{n-1}$, and for any x and $y = (\text{logit} \circ T^{-1})(x)$ the elements of y are.

$$
\begin{aligned}
y_i &= \text{logit} \frac{\sum_{j \in \text{leaves}(\text{right}(i))} x_j}{\sum_{k \in \text{leaves}(i)} x_k} \\
&= \log \frac{(\sum_{j \in \text{leaves}(\text{right}(i))} x_j)/(\sum_{k \in \text{leaves}(i)} x_k)}{1 - (\sum_{j \in \text{leaves}(\text{right}(i))} x_j)/(\sum_{k \in \text{leaves}(i)} x_k)} \\
&= \log \frac{\sum_{j \in \text{leaves}(\text{right}(i))} x_j}{\sum_{k \in \text{leaves}(\text{left}(i))} x_k}
\end{aligned}
$$

Comparing the definition here to the one in Equation 2.12, we see that it differs only in whether the x_js are multiplied or summed. The ilr with sequential binary partitions is the log ratio of products, while the logit inverse Pólya tree transformation is the log ratio of sums, both in terms of a full binary tree.

Where coordinates from ilr using sequential binary partitions are subcompositional balances measured as the log-ratio of products (multiplied by a constant to preserve orthonormality), balances in hierarchical stick breaking are simple proportions between sums of constituent parts. This has the advantage of producing more easily interpreted coordinates. If a pie is sliced into n pieces, a more intuitive notion of the balance between two groups of pieces is a comparison of sums (which group ate the most pie) rather than products (which group ate the larger sum of log piece sizes). But given that the interpretation of both methods revolve around balance of one kind or another, models that use sequential binary partitions and ilr (such as the phylogeny-based transformations defined by Silverman et al. (2017)) could quite easily be made to work with Pólya tree transformations, and there may be major advantages in doing so.

One disadvantage of the Pólya tree transformation is that there is no clear interpretation under Aitchison geometry. It does not correspond to any basis, much less an orthonormal one, which we can easily prove.

Theorem 2.2.14. *A logit inverse Pólya tree transformation is not necessarily represented as coordinates in some Aitchison geometry basis.*

Proof. Consider the logit inverse Pólya tree transformation $S : \Delta^2 \mapsto \mathbb{R}^2$ defined by

$$
S(x_1, x_2, x_3) = \left(\log \frac{x_1}{x_2 + x_3}, \log \frac{x_2}{x_3} \right)
$$

Consider the point $x = (1/2, 1/4, 1/4)$. From the definition of the transformation $S(x) = (0, 0)$.

Suppose there is a Aitchison geometry basis (e_1, e_2) corresponding to this transformation. Then $(0 \odot e_1) \oplus (0 \odot e_2) = (1/2, 1/4, 1/4)$. However,

$$\begin{aligned}(0 \odot e_1) \oplus (0 \odot e_2) &= \mathcal{C}(e_{11}^0, e_{12}^0, e_{13}^0) \oplus \mathcal{C}(e_{21}^0, e_{22}^0, e_{23}^0) \\ &= \mathcal{C}(1, 1, 1) \oplus \mathcal{C}(1, 1, 1) \\ &= (1/3, 1/3, 1/3) \oplus (1/3, 1/3, 1/3) \\ &= (1/3, 1/3, 1/3) \\ &\neq (1/2, 1/4, 1/4)\end{aligned}$$

This contradiction then proves that Pólya tree transformations cannot be represented as linear transformations in Aitchison geometry. □

Whether there is an alternative stick breaking geometry on the simplex remains unclear. From the proof above it is apparent that if there is one, it must rely on different definitions of addition and scalar multiplication.

2.2.5 Heuristics for tree-based transformations

Both the isometric log-ratio transformation with sequential binary partitions and the Pólya tree transformation are defined by with respect to full binary tree with n leaves. Ideally, this tree would be another parameter of our approximating distribution q, which would be optimized over, thus choosing the tree topology that minimized the KL divergence to the true normalized likelihood. Unsurprisingly, there are far too many possible trees for this to be feasible by any optimization method that considers every tree. This would require considering not just the $n!$ permutations of the stick breaking process, but also the $C_{n-1} = \frac{1}{n}\binom{2(n-1)}{n-1}$ (the $(n-1)$st Catalan number) possible full binary trees with n leaves, resulting in a family of $C_{n-1}n! = \frac{(2n-2)!}{(n-1)!}$ possible transformations.

Though Zhang and Zhou (2017) were able to effectively optimize over stick-breaking topologies, at most 11 labels were used, and only permutations were considered. We would have a much harder time sampling over the possible configurations of a Pólya tree transformation. To avoid an exhaustive exploration of the space of trees we must either find an adequate heuristic with which to choose a tree, prior to optimizing parameters, or pursue optimization metaheuristics (e.g., simulated annealing or genetic programming) that could explore a subset of the space in a guided way. The latter may be possible, but each topology is accompanied with an entirely new set of parameters that must be optimized. Optimizing both concurrently is unlikely to be practical.

Fortunately, there are reasonable approaches we may take to choosing a topology ahead of optimization. RNA-Seq is typically a sparse mixture model. Most reads are compatible with only a small number of annotated transcripts. Because of this, the problem displays subcompositional independence: knowing the mixture of isoforms expressed in one gene tells us

nothing about the mixture of isoforms expressed in another, if the genes share no reads. This suggests that the transformation might be oriented in a way to try to capture this structure.

Aitchison (1986a) discusses the concept of subcompositional independence, as well as several other notions of independence on the simplex. There, tests for independence are proposed, but they necessitate estimating the full covariance matrix, an intractable task for a large n. Instead we pursue the idea of capturing a similar subcompositional independence structure by using hierarchical clustering as a heuristic. Each transcript is represented by its set of compatible reads. We then cluster greedily, choosing the maximum Jaccard index (i.e., the size of the intersection divided by the size of the union) at each step. Transcripts, or sets of transcripts, that share a large proportion of their compatible reads have a have a higher Jaccard index, and thus their common ancestor is placed lower in the tree. In an ideal scenario, this constructs a tree that encodes a distribution family that has a similar subcompositional independence structure to that of the real likelihood function. Pseudocode for this algorithm is given in Algorithm 1. A simple illustration of the idea is shown in Figure 2.7.

`// Initial single-node subtrees`
$T \leftarrow ((i) \forall i \in 1, \dots, n)$;
`// Sets of compatible reads corresponding to each subtree in` S
$R \leftarrow (\{j | p_i(r_j) \neq 0\} \forall i \in 1, \dots, n)$;
while $|T| > 1$ **do**
  $u^*, v^* \leftarrow \arg\max_{\substack{u,v \in \{1,\dots,|T|\} \\ s \neq t}} \frac{|R_u \cap R_v|}{|R_u \cup R_v|}$;
  $T \leftarrow (T \backslash \{T_{u^*}, T_{v^*}\}) \cup \{(T_{u^*}, T_{v^*})\}$;
  $R \leftarrow (R \backslash \{R_{u^*}, R_{v^*}\}) \cup \{R_{u^*} \cup R_{v^*}\}$;
end

Algorithm 1: Building a transformation tree using a form of hierarchical clustering. At each step, the algorithm greedily chooses the pair of transcript sets that have the largest Jaccard index between their sets of compatible reads.

Computing the maximum Jaccard index at each iteration makes even this greedy algorithm $O(n^3)$, which is not feasible. It practice we sort the list of subtrees by the median genomic position among compatible reads. This will tend to sort transcripts that share many reads near each other. We then compare only transcripts that are within some neighborhood of size c within this sorted list, resulting in an algorithm that is $O(nc^2))$.

In cases of complete subcompositional independence, where no reads are shared between transcripts, the likelihood function in Equation 1.1 is proportional to a Dirichlet distribution. As shown in Theorem 2.2.11 and Corollary 2.2.13, any Pólya tree transformation can exactly fit any Dirichlet distribution of the same dimensionality, if applied to appropriately chosen Beta distributed random variables.

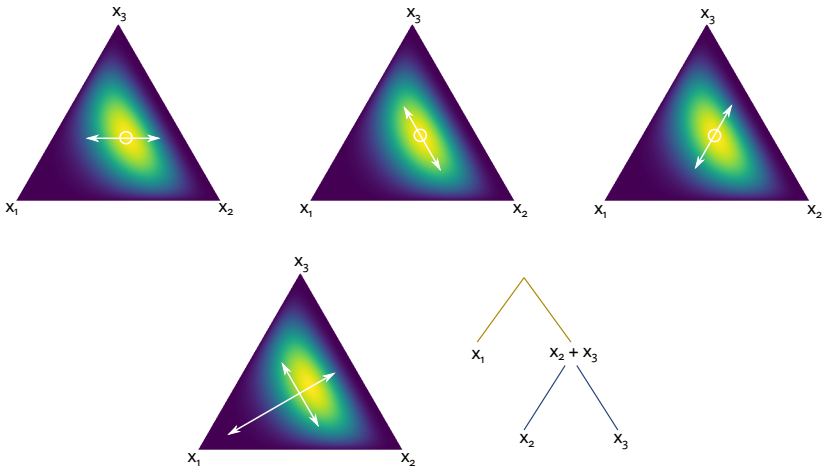

Figure 2.7: In the top row we consider each pair of dimensions, attempting to greedily find the pair that capture the most variance according the distribution with whole heatmap is underlain. Ultimately choosing the tree that more closely captures the subcompositional independence structure of that data.

In Section 4.1 we explore an alternative tree heuristic using the Jaccard index computed on sequence k-mers. Since this heuristic is not a function of the reads, a fixed topology can be used across samples, potentially speeding up inference.

2.2.6 Base distributions

Following the tactic of minimizing the KL divergence using the reparameterization trick to backpropagate gradients through random variables, we need to equip our transformation with a base distribution to induce new new class of distributions over the simplex. The Pólya tree transform has been described described here as a $(0,1)^{n-1} \to \Delta^{n-1}$ transformation. Most of the compositional data analysis transforms take the form $\mathbb{R}^{n-1} \to \Delta^{n-1}$. We can always use the logit function (or its inverse) to move between the two, so this distinction is insignificant. We would like then to find a distribution over $\mathbb{R}^{n-1}$ or $(0,1)^{n-1}$. To minimize the parameter space, each element will be considered independent (a common approach referred to as "mean field variational inference").

We are limited to distributions that lend themselves to the reparameterization trick. The two distributions we will consider are the normal (or logit-normal) distribution and the Kumaraswamy distribution (Jones, 2009), which is qualitatively similar to the Beta distribution but, unlike the Beta distribution, can be easily expressed as a transform of a uniform distribution.

Transformation family	Parameterization	Family size
Additive log-ratio	divisor element	n
Multiplicative log-ratio	permutation	$n!$
Isometric log-ratio	full bin. tree w/ n leaves	$C_{n-1}n!$
Pólya tree transformation	full bin. tree w/ n leaves	$C_{n-1}n!$

Table 2.1: Three families of transformations onto the simplex. Here C_{n-1} is $(n-1)$th Catalan number. Transformation families are all parameterized. Different parameterizations result in a different transformation. The number of possible parameterizations is given in the family size column.

Lastly we explore a further transformation of the normal distribution using a parameterized sinh-arcsinh transformation of the following form

$$U(Z;\alpha) = \sinh(\alpha + \operatorname{arcsinh}(Z))$$

This technique, explored by Jones and Pewsey (2009) along with a two-parameter version, provides an analytically convenient way to add a parameter controlling skewness to a distribution. Where $Z \sim \text{Normal}(0,1)$, we use

$$Y = \mu + \sigma U(Z;\alpha)$$

as our fully reparameterized distribution. We will refer to the resulting distribution as a *skew-normal* distribution, though other distribution families also go by this name (Azzalini, 1985; Hosking and Wallis, 2005).

2.3 Evaluating approximations

In the prior section we introduced the Pólya tree transformation along with two other bijective transformations suitable for constructing distributions over the simplex. Table 2.1 summarizes these. We are primarily interested in achieving goodness of fit. Before any empirical benchmarks, it is worth briefly exploring qualitative properties of the transformations.

The upper part of Figure 2.8 shows a Cartesian grid mapped to the Δ^2 using different parameterizations of each of the first three transformations discussed. The mlr transformation is generalized by Pólya tree transformations under a logit (or, in the opposite direction, logistic) function. On Δ^2, the set of mlr and logit Pólya tree transformations are the same. In higher dimensionality, there are more trees to choose from and Pólya tree transformations become a much larger family. We can see in these plots that each transformation distorts Euclidean space in a characteristic manner, and that this distortion becomes more extreme as we approach the edges of the simplex.

The mlr (and the logit Pólya tree transformation, which generalizes it), does not map the 0 vector to the middle of the simplex like the ilr and alr transformations. If fact, the Pólya tree transformation is only centered

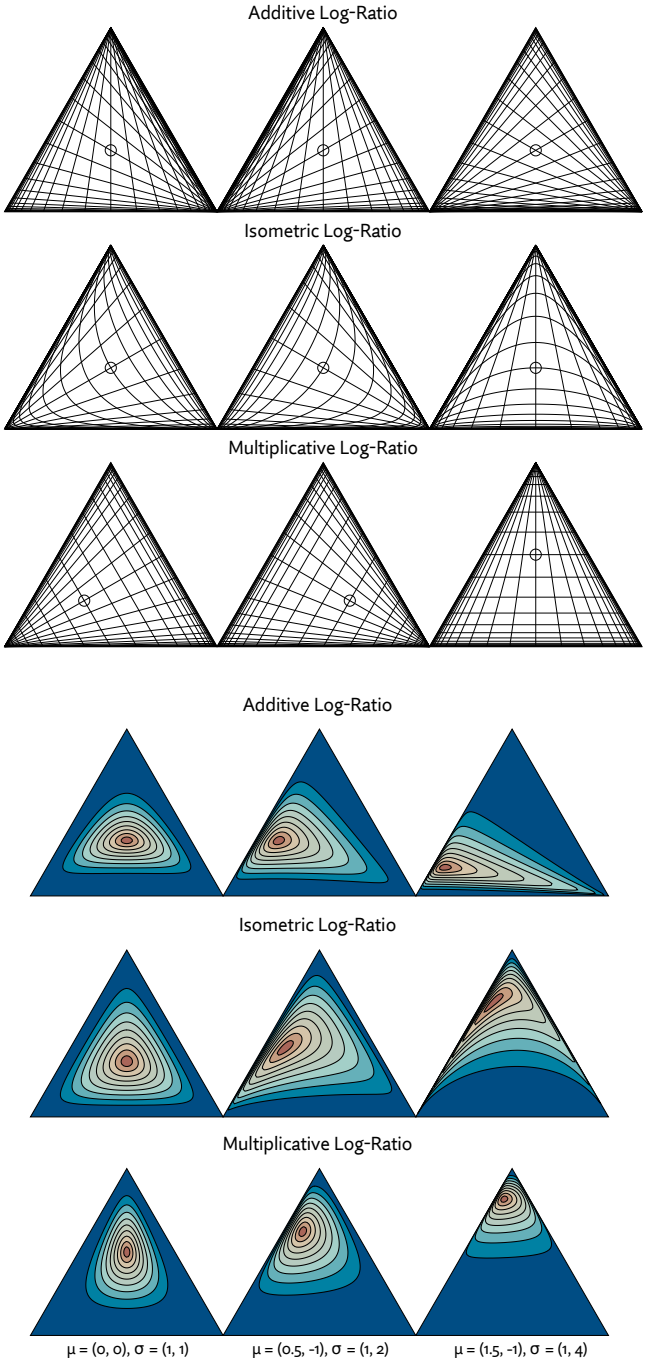

Figure 2.8: Above, Cartesian grid lines are transformed onto Δ^2 using three classes of transformations: additive log-ratio, isometric log-ratio, and multiplicative log-ratio. Grid lines are spaced evenly at a distance of 0.5 in Euclidean space and the point $(0, 0)$ is marked with a circle. Each transformation has variations shown in the columns. The variations are formed by choosing a different denominator, basis, or permutation for alr, ilr, and stick breaking, respectively. Below, various parameterizations of a 2-dimensional multivariate normal distribution, with a diagonal covariance matrix, are mapped onto the simplex with alr, ilr, and mlr (for each, the furthest right of the three variants shown on the left). The choice of transformation has a dramatic effect on the resulting distribution.

when n is a power of 2 and the tree is perfectly balanced. In higher dimensionality with a highly unbalanced tree, the transformation can be highly uncentered, mapping the zero vector near an edge of the simplex. We could correct for this, and indeed the stick breaking transformation used in Stan does so by adding a constant. Rather than mapping y onto the simplex, they compute y' as

$$y'_i = y_i + \log\left(\frac{1}{n-i}\right)$$

and map this onto the simplex. The same approach can be taken with Pólya tree transformations, but since we are heuristically fitting the topology of the transformation to the data, it matters less.

We can see in the bottom part of Figure 2.8 that each transformation distorts a normal distribution in a dramatically different way. The ilr distribution in particular very easily becomes non-concave. Since RNA-Seq in Equation 1.1 is always concave, this suggests that ilr may not be the best fit. Regardless, even if using normal distributed variables, it is clear we are dealing with very different distribution families after transformation.

2.3.1 A dataset for evaluating goodness of fit

To evaluate empirical properties of likelihood approximation we chose a publicly available mouse body map dataset (Li et al., 2017). The samples consists of 17 mouse tissues, taken from male and female mice, each 6 weeks old. There are multiple replicate animals as well replicate samples from the same animal. Samples were sequenced using the Illumina HiScan platform generating an average of roughly 10 million 101bp paired-end reads per sample. All samples are available under the Sequence Read Archive accession number PRJNA375882.

There is no single representative RNA-Seq dataset, but using an expansive body map project with many tissues and replicates allows for many possible comparisons with less concern for batch effects.

We use version 95 of the Ensembl gene annotations (Ensembl, 2018) and version 38 of the mouse genome assembly. The number of annotated transcripts (which we have denoted n in prior discussion) is 138,930. Reads were aligned to reference genomes (also obtained from Ensembl) using version 2.1.0 of HISAT (Kim, Langmead, and Salzberg, 2015).

2.3.2 Testing procedures

There are multiple ways to approach the question of how good an approximation is. We present two here. First, goodness of fit of marginal densities is measured using signed rank null hypothesis tests. Second, we attempt to more directly address how approximation affects downstream analysis by measuring error introduced to fold-change estimates.

In both cases, we compare the approximation to samples generated from a Gibbs sampler, which are treated as ground truth. There are more so-

phisticated ways to sample using the RNA-Seq likelihood function and a flat prior, and even more sophisticated implementations of Gibbs sampling, but this basic approach is not dissimilar to what exists in RSEM (Li and Dewey, 2011) or BitSeq (Glaus, Honkela, and Rattray, 2012), for example.

Gibbs samplers, indeed any method that samples from conditional distributions, run the risk of generating highly correlated samples when there is strong dependence between variables (as is the case when transcripts have many reads in common). To assess convergence we use the $\hat{R}$ statistic defined by Gelman and Rubin (1992), which gives the ratio of within- to between-chain variance. If multiple chains are used with random initializations drawn from a distribution that is overdispersed compared to the true likelihood, between-chain variance will tend to be greater than within-chain variance, but converge to 1 as the number of samples approaches infinity.

We use 8 chains, initialized to uniformly random points on the simplex. Each chain is run for 2000 "burn-in" iterations which are not recorded, then an additional 25,000 iterations are run, recording every 25th iteration to produce 1000 total samples for analysis. The $\hat{R}$ statistic is computed for every transcript, measuring the convergence of the expected relative transcript expression.

Figure 2.9 shows that after 25,000 iterations over 99.9% of transcripts are converged according to the simple rule of thumb $\hat{R} < 1.1$ suggested in Gelman et al. (2013a). Between 33 and 131 transcripts (i.e., less than 0.1%) remain unconverged by this criteria in the samples used. Unsurprisingly, groups of transcripts that share a very large number of reads would take a very large number number of iterations of the Gibbs sampler to fully converge. It is likely that methods that do use Gibbs samplers to estimate expression use insufficient samples to fully resolve every transcript. For example, RSEM uses 500 burn-in samples and generates 1000 total additional samples. Though the Gibbs sampler is a easily applicable to RNA-Seq, the very slow convergence of some transcripts suggests that, if sampling is desired, the problem might be better served by a more sophisticated method, like Hamiltonian Monte Carlo. To account for variance introduced by the Gibbs sampler, we ran it again using different random number generator seeds. Those measurements are included in the plots that follow as "Gibbs replicate".

We compare a number of approximations to RNA-Seq likelihood (Table 2.2). Each method in constructed as a transformation of $n-1$ independent random variables onto Δ^{n-1}. We use the two standard bijections defined in the compositional data analysis literature: the additive and isometric log-ratio transformations.

Both hierarchical stick breaking and ilr are defined in terms of a tree. With the former we try three strategies. First, a sequential or degenerate tree is used making the transformation equivalent to classical definitions of stick breaking. Under a logit transformation, this is equivalent to mlr. Second, we build a random tree by starting with n individual nodes, then join-

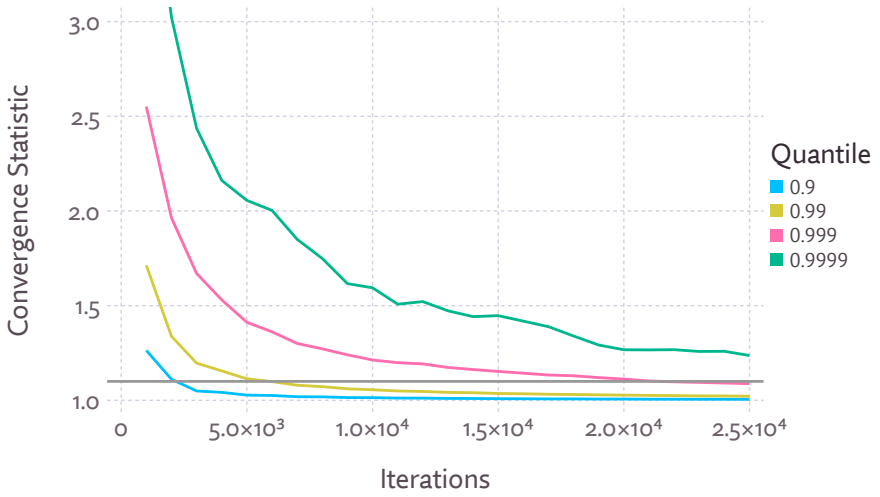

Figure 2.9: Upper quantiles of the convergence statistic $\hat{R}$, computed for every transcript as the Gibbs sampler is run. The horizontal gray line shows the rule of thumb convergence criteria $\hat{R} < 1.1$. After 25,000 iterations, approximately 99.9% of transcript expression estimates have converged. The slope of the 99.99% percentile line suggests that the remaining transcripts may require exponentially more iterations to converge.

ing random subtrees until one tree is left. With such a procedure the tree will be balanced in expectation, but otherwise arbitrary. Finally, we use the Jaccard index heuristic described in Section 2.2.5 with approximate hierarchical clustering to build the tree from the bottom up. With ilr, we only use the Jaccard index heuristic, as it was found to typically outperform the others.

As described in Section 2.2.6, we try three base distributions in conjunction with the stick breaking transformations: logit-normal, Kumaraswamy, and logit-skew-normal. With other transformations, alr and ilr, only the normal distribution is used. As will be shown, performance was found to be so lagging with these transformations that variations of the base distribution would be unlikely to make them worth considering.

Each method is optimized using stochastic gradient descent, using Adam (Kingma and Ba, 2014) to update ϕ estimates each iteration, and 500 iterations, with each iteration estimating gradients using 6 Monte Carlo samples of z.

2.3.3 The Pólya tree transform improves the goodness of fit to likelihood marginals

To evaluate the overall fit of our likelihood approximation we tested the marginal densities both at the transcript and gene level, where gene expression is defined as the sum of the expression of the gene's constituent

Distribution	Transformation	Tree Heuristic
normal	additive log-ratio (alr)	
normal	isometric log-ratio (ilr)	hierarchical clustering
normal	logit Pólya tree transformation	sequential
normal	logit Pólya tree transformation	random
skew-normal	logit Pólya tree transformation	hierarchical clustering
Kumaraswamy	Pólya tree transformation	hierarchical clustering

Table 2.2: Approximations evaluated, each defined as a transformation of $n-1$ independent variables from a base distribution. The hierarchical clustering heuristic is described in Section 2.2.5. Sequential and random refer to building a degenerate (i.e., linear) tree, and build a tree iteratively by joining random nodes, respectively.

transcripts. To evaluate the fit, we perform a Wilcoxon signed-rank test comparing 1000 samples from the Gibbs sampler to an equal number directly sampled from the approximation, producing a p-value for each transcript and gene.

A larger p-value implies less distinguishability between the approximation and true distribution, and thus a better fit. If our approximation is perfect, we should expect to see the p-values uniformly distributed across $(0, 1]$. Imperfect approximations will yield p-value distributions that are increasingly skewed towards smaller numbers. Figure 2.11 shows the results of this test using a mouse brain sample. To provide some intuition for the correspondence between p-value and fit, a number of examples with low p-values are plotted in Figure 2.12. We can see from that figure that a low p-value, that is, one customarily signifying "statistical significance", is not necessarily indicative of terribly poor fit.

We also see that the approximating distribution family matters a great deal. Distributions based on the Pólya tree transformation offer a dramatically better fit than the traditional compositional data analysis transformations alr and ilr. Moreover, the heuristic tree construction is a sharp improvement over a random tree topology. The common sequential stick-breaking approach (equivalent to the mlr transformation) is seen to be the worst approach to stick-breaking, perhaps in part because such a long chain of dependent breaks is numerically less stable and thus more difficult to fit to the target likelihood. As no attempt was made to optimize over permutations, it is also possible sequential stick breaking could be improved with the right permutation heuristic.

On this test, the approximate factorization approach proposed by Zakeri et al. (2017) also performs very well, offering a slight median improvement over the Pólya tree transformation. In Figure 2.10 we see that this comes at some cost. Though far more efficient in time and space than the unapproximated likelihood function, it does not match the low constant time and space of the Pólya tree transformation. Exact factorization (labeled

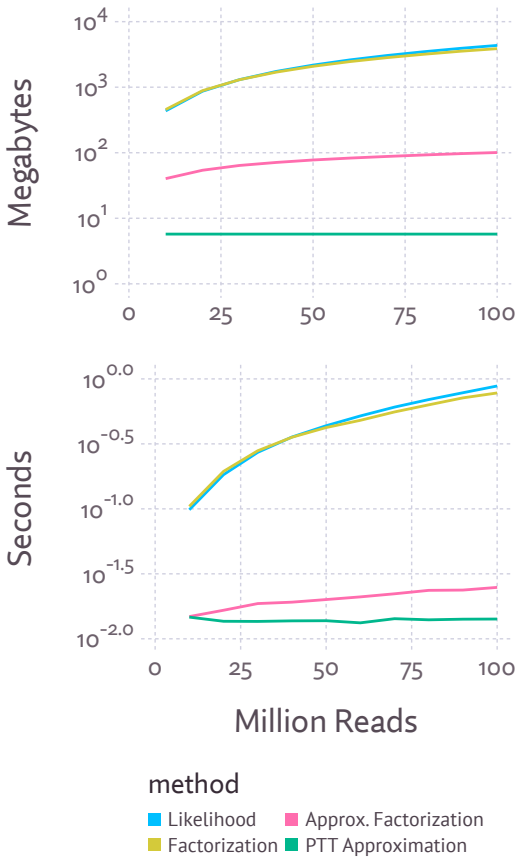

Figure 2.10: A comparison of the space and time needed to evaluate the likelihood function and its approximations. A GTEx sample (SRR1340508) was subsampled to, and its likelihood evaluated for, increasing numbers of reads. The upper plot shows the memory in megabytes necessary to evaluate the function, the lower plot, the time in seconds necessary to evaluate the function once.

"Factorization" in the figure), has a negligible effect, as few read pairs, even with 100 million reads, are completely identical.

The plot of p-value distributions on a Log10 scale gives a sense of what the worst case fit is for each method. The clustering heuristic improves a great deal on the worst case performance of sequential or random stick breaking. The choice of base distribution is also significant, with the Kumaraswamy distribution having a somewhat lower worst case than either the normal or skew-normal distributions. Marginalizing transcript expression to get gene expression, and then computing p-values at the gene level produces a picture very similar to the one at the transcript level. Compared to transcript expression, all the stick breaking approximations have slightly better fit, whereas alr and ilr have slightly reduced fit at the gene level.

This results hold across the other samples from Li et al. (2017) as well (Figure 2.13), but to ensure that the approximation is broadly effective and

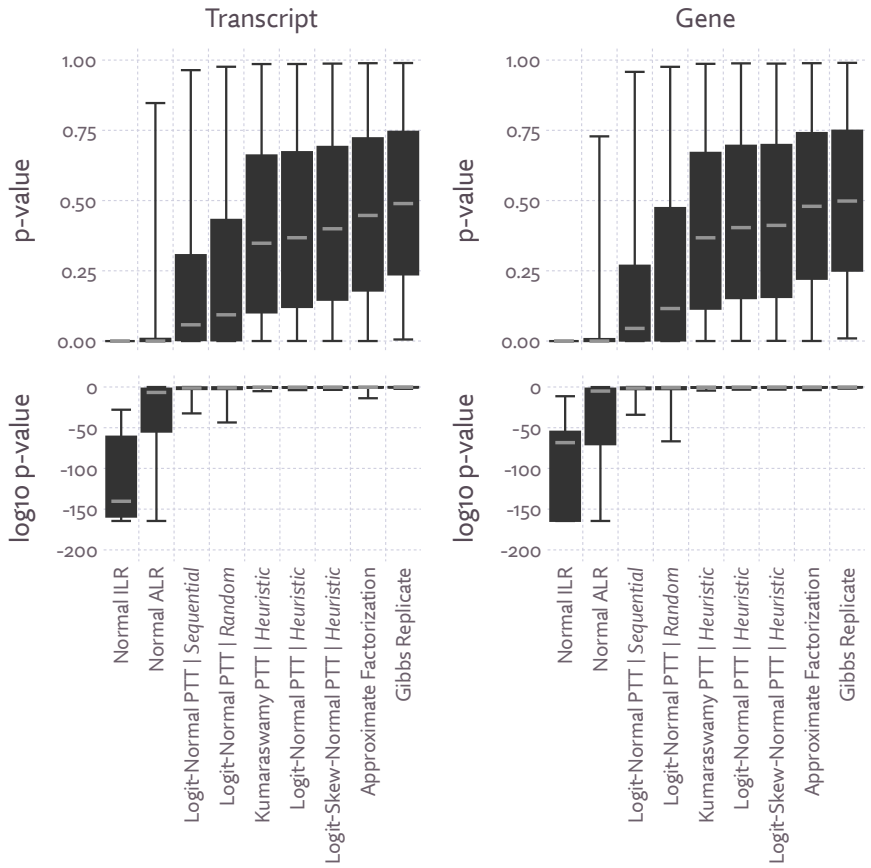

Figure 2.11: Distributions of p-values evaluating the agreement between 1000 samples from the Gibbs sampler and 1000 samples drawn directly from various approximations of the likelihood function for a mouse brain sample. P-values were computed using the Wilcoxon signed-rank test. On the left, p-values are computed comparing the marginal likelihood at the transcript level, on the right, at the gene level. Boxplots are drawn with upper and lower whiskers corresponding to the 99th and 1st percentile, respectively.

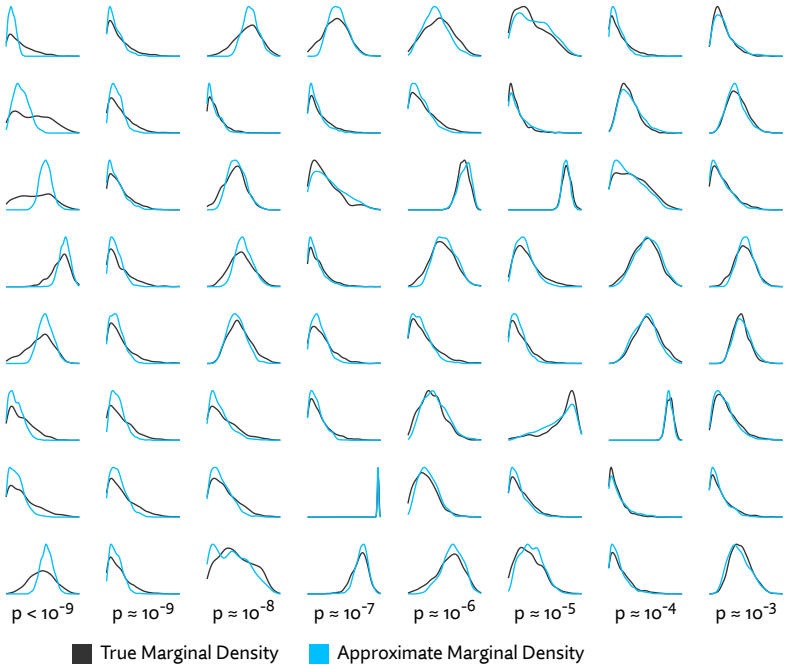

Figure 2.12: Kernel density plots of exact and approximate marginal densities for examples with small p-values. Columns correspond to p-values of decreasing powers of 10, while rows are randomly selected examples with approximately that p-value, giving a sense of how p-value corresponds to goodness of fit. It can be seen that a very small p-value does not necessarily correspond to a catastrophic failure of the approximation. The plots shown were generated from a mouse liver sample from Li et al. (2017) with approximate densities using logit-skew-normal Pólya tree transform distribution, using the hierarchical clustering heuristic to choose the tree topology.

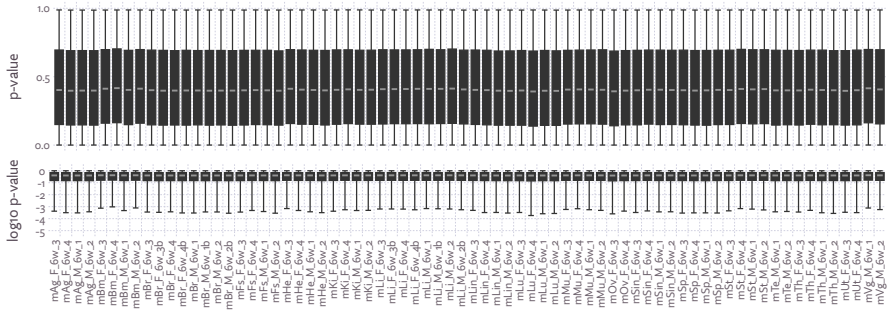

Figure 2.13: The likelihood function for every sample from (Li et al., 2017) was approximated. 1000 samples were drawn with probability proportional to the approximated likelihood, and 1000 samples were drawn using a Gibbs sampler. A Wilcoxon signed-rank test was performed for each annotated transcript, and the distribution plotted in a boxplot, drawn with upper and lower whiskers corresponding to the 99th and 1st percentile, respectively. An ideal fit would have a uniform distribution of p-values.

Species	Accession	Read Len.	Num. Reads	Num. Trans.
S. Cerevisiae	SRR453566	101	9×10^6	7126
D. Melanogaster	SRR030231	76	1.4×10^7	34749
C. Elegans	SRR065719	76	7.2×10^7	58941
M. Musculus	SRR023546	101	1.5×10^7	131195
H. Sapiens	SRR896663	101	1.1×10^7	200310

Table 2.3: Auxiliary datasets used to evaluate the consistency of likelihood approximation performance. Numbers of transcripts listed are those annotated in version 90 of Ensembl for each respective species. Numbers of reads are those that are properly paired and aligned. All datasets are paired-end reads. All accession numbers are for the Sequence Read Archive (Leinonen et al., 2011).

not somehow tuned to these particular datasets, we also evaluated logit-skew-normal Pólya tree transform distribution with heuristic tree topologies on a variety of of other RNA-Seq datasets, spanning a number of species with varying transcriptional complexity. We see that likelihood approximation has remarkably consistent performance across these samples (Table 2.3). Curiously though, the approximation appears to fit moderately better with a larger number of transcripts. The median p-value for both the human and mouse samples is approximately 0.4, and for the yeast sample, 0.28. Likely this is due to a smaller proportion of the transcripts being expressed in species with extensive alternative splicing.

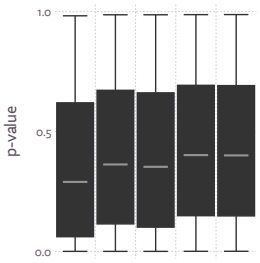

Figure 2.14: Distributions of p-values measuring the goodness of fit of likelihood approximated using logit-skew-normal hierarchical stick breaking, evaluated on auxiliary datasets (Table 2.3). P-values are calculated using Wilcoxon signed-rank tests between 1000 samples from a Gibbs sampler and the same number of samples drawn from the approximation. Boxplot whiskers correspond to 1st and 99th percentiles.

2.3.4 Approximate likelihood introduces little error into fold-change lower bound estimates

Evaluating marginal fit using null hypothesis tests is informative, but somewhat removed from the goals of transcriptomic analysis: detecting and quantifying changes in gene or transcript expression. To understand how results differ when using an approximation in place of sampling from the true likelihood using the Gibbs sampler, we compared the Gibbs sampler to the candidate approximations when computing what we are calling *minimum log2 fold-change*.

Minimum log2 fold-change here is defined as the 90% posterior lower bound on fold change. More precisely, if $x_i^{(a)}, x_i^{(b)}$ are the unobserved transcript expression values for transcript i from sample a and b, respectively, then the minimum log2 fold change is a number δ_i where

$$P\left(\left|\log_2 \frac{x_i^{(a)}}{x_j^{(b)}}\right| > \delta_i\right) = 0.9$$

Put another way, there is a 90% probability the log fold change is above the given level. This is a convenient way of measuring the support for differential expression. In comparison, most null hypothesis tests for differential expression tend to use an a priori implausible null hypothesis of no change whatsoever. In doing so they fail to consider effect size directly. A Bayesian alternative is to set a fold-change cutoff and compute the probability the true effect exceeds this cutoff. Doing so considers effect size, but only the binary of exceeding or falling below the predetermined cutoff.

We selected a male sample for each of 16 tissues and computed the minimum log2 fold-change for every transcript and every pair of samples. This subsetting was done simply to remain computationally manageable while covering a large variety of comparisons. We compared a total of 120 pairs

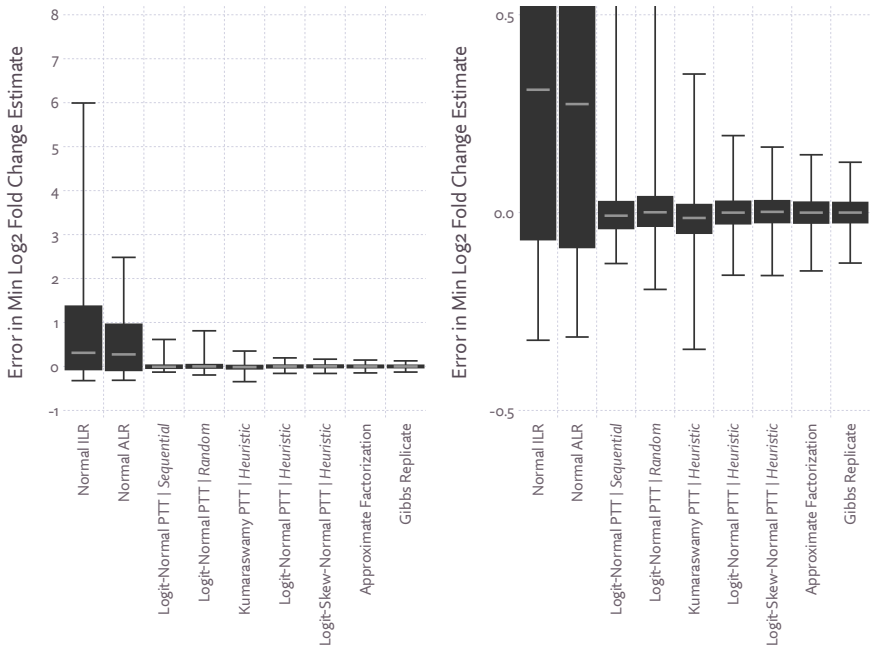

Figure 2.15: Error in estimates of minimum log2 fold-change introduced by various approximations. The same plot in shown on the left and right, with the right plot zoomed in. A positive error indicates an overestimation of the true effect size, potentially introducing false positives, a negative error an underestimation, potentially introducing false negatives. "Minimum log2 effect size" is the effect size at which there is a 90% posterior probability that the true effect size is higher. Boxplots are drawn with upper and lower whiskers corresponding to the 99th and 1st percentile, respectively.

in total, and for each the minimum log2 fold-change was computed for every transcript. The results of this test are show in Figure 2.15.

As we saw with the goodness of fit results, using a logit-skew-normal distribution with a Pólya tree transformation constructed using our heuristic produces an extremely accurate approximation, only exceeded slightly by approximate factorization, which uses significantly more space. In this benchmark the sinh-arcsinh transformation (what we are calling "skew-normal") had little benefit. Depending on the application, the extra computational cost of this transformation may not be worth the added accuracy.

We took this idea further and explicitly made differential expression calls to see how much they would be affected by approximation. In Table 2.4 false positive and false negative rates are listed for calls made with a minimum fold change of 1.5. On this test the best approximations — specifically, ours, and approximate factorization — introduce an approximately 3% false positive rate. The Pólya tree transformation approximation is only slightly worse than simply running the Gibbs sampler a second time, and actually has a lower false positive rate than approximate factorization.

Approximation	fpr	fnr
Normal ILR	0.6946	0.0072
Normal ALR	0.6519	0.0011
Logit-Normal|PTT|Sequential	0.1135	0.0025
Logit-Normal|PTT|Random	0.1222	0.0062
Kumaraswamy|PTT|Heuristic	0.0379	0.0072
Logit-Normal|PTT|Heuristic	0.0673	0.0156
Logit-Skew-Normal|PTT|Heuristic	0.0311	0.0078
Approximate Factorization	0.0327	0.0069
Gibbs Replicate	0.0284	0.0059

Table 2.4: Differential transcript expression false positive and false negative rates for various approximation methods. "False" here is with respect to Gibbs sampling which is treated as ground truth. Transcripts are considered differentially expressed if they have a at least a 90% probability of having at least 1.5 absolute fold change.

To further prioritize even lower false-positive rates, alternative objective functions could be used when fitting the approximation. Some recent alternatives to KL divergence have been explored by Dieng et al. (2016) and Li and Turner (2016). These approaches define a class of divergences that can be parameterized to control how inexact approximations are fit, and set to prefer overdispersion to underdispersion, which map be preferable if our desire is to make more conservative calls. Extending this method to use an alternative objective function is a promising future direction, but is not a trivial change.

2.3.5 Estimating and sampling from approximated likelihood can be faster than bootstrap sampling

The procedure for sampling expression vectors from an approximated likelihood function (or more precisely, from the posterior formed from a uniform prior and the approximate likelihood) is to simply generate a random vector from a Normal$(0, I)$ distribution, and apply the transform, which is $O(n)$ where n is the number of annotated transcripts. Because samples are so cheap to generate, for a sufficiently large numbers of samples, it outperforms not only MCMC approaches, but also the extremely fast bootstrap approach implemented in kallisto (Bray et al., 2016).

To locate the break even point, we recorded the overall time needed to generate increasingly large numbers of samples with both Kallisto and the software we use to perform likelihood approximation, which we call *Polee*. In both approaches these times include the necessary initialization time. Kallisto uses its own pseudoalignment algorithm, while Polee takes as in-

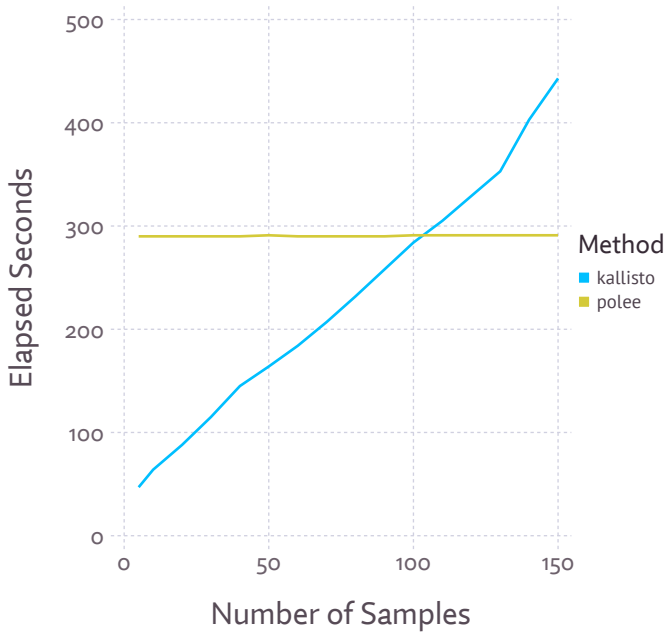

Figure 2.16: Overall run time to generate samples using kallisto (bootstrap) or Polee (approximate the likelihood and sample directly from the approximation). Polee requires additional run-time up front to approximate the likelihood function, but once computed, generating samples is exceedingly efficient, with the time only increasing by about 0.69 seconds between 1 sample and 150.

put existing alignments. To compare on equal ground, we exported alignments generated by Kallisto and used them as input into Polee. Added to the Polee timings is the time Kallisto took to generate these alignments, the time it took to approximate the likelihood function, and to generated the requested number of samples. Both methods were run on 8 cpu cores.

From the results shown in Figure 2.16, we see that past about 100 samples (where each sample is a vector of transcript expression estimates), the amortized cost of sampling becomes less for Polee than Kallisto. Approximate likelihood functions can be evaluated directly, so there is not necessarily a need to sample, but sampling is useful in some applications. For example, Polee can be used as a drop-in replacement for Kallisto when using sleuth (Pimentel et al., 2017) to call differential expression. Speed will be similar or faster, and this sampling technique is not subject to the limitations of bootstrap sampling, which can yield imprecise results for transcripts with a very small number of reads.

2.4 Notes on implementation

As mentioned, we implemented the approximate likelihood method described here in a tool called Polee (a portmanteau "Pólya tree") and made the source code publicly available under an open source license (`https://github.`

com/dcjones/polee). The software is implemented in the Julia programming language (Bezanson et al., 2012), Python, and C++.

There are essentially two parts to the software. The first part, implemented in Julia, produces approximations of the likelihood function given aligned RNA-Seq reads. The second consists of models and supporting code implemented largely in Python using TensorFlow (Abadi et al., 2016), with some custom TensorFlow operations implemented in C++.

2.4.1 Approximating likelihood

The code necessary for optimizing the approximation is not complicated. We perform stochastic gradient descent using the Adam update rules (Kingma and Ba, 2014), finding this to converge relatively quickly and not be particularly sensitive to choice in tuning parameters. We use exponential decay rates of $\beta_1 = 0.9$, $\beta_2 = 0.7$, for the moment estimates, and compute the exponentially decaying learning rate as $\alpha_t = \exp(-0.02t)$, where t is the step number. By default the optimization runs for 500 steps, using 6 Monte Carlo samples for each step. Optimization tends to take around 1-4 minutes depending on the sequencing depth and the complexity of the gene annotations. Most of the execution of this algorithm (e.g., computing transformations and the RNA-Seq likelihood) is trivially parallelized. The exception is the Pólya tree transformation itself. Computing the transformation and its gradients necessitates tree traversals, which are partially amenable to multi-threaded computation, but not trivially so.

Unlike Kallisto or Salmon, Polee is not alignment-free, although it can be used with approximate alignments produced by Kallisto or RapMap (Srivastava et al., 2016). Most of the complexity in the implementation of Polee stems from determining read compatibility and computing $p_j(r_i)$ for each read i and transcript j. In the worst case, this can be an intractable $O(mn)$ task, but in practice $p_j(r_i) = 0$ for the vast majority of i, j pairs. When represented as a matrix, it tends to be a quite sparse.

To make this task tractable we implemented a data structure to rapidly find overlapping intervals between two collections. Since $p_j(r_i) > 0$ only if the read (or read pair) interval intersects the transcript interval, this can rule out most i, j pairs in practice. The data structure consists of a B+-tree augmented so that internal node stores the largest interval end point in each its subtrees, an idea discussed by Cormen et al. (2009).

2.4.2 Bias correction

Bias correction methods implemented in Polee follow a methodology similar to Jones et al. (2012). The essential approach is to train a probabilistic classifier to discriminate between observed fragments and random fragments from the same transcripts, then using the probability to produces to compute bias.

1. Randomly sample a subset of reads.
2. Assign these reads to transcripts by maximum likelihood, to produce a set of implied fragments F_f
3. Reposition fragments in F_f uniformly at random within their transcripts to produce a set of hypothetical background fragments F_b.
4. Train a probabilistic classifier on $F_f \cup F_b$ to compute $P(f \in F_f | c(f))$, where $c(f)$ is the confounding features (e.g., GC content) of the fragment.
5. Compute bias for a fragment f as

$$b(f) = \frac{P(c(f)|f \in F_f)}{P(c(f)|f \in F_b)}$$

The calculation of bias here is equivalent to Equation 1.7. It should also be recognizable as a Bayes factor. Here $|F_f| = |F_b|$, so the obvious prior is $P(f \in F_f) = 0.5$, and in such case, the posterior probability is simple to compute from the Bayes factor, or vice versa,

$$P(f \in F_f | c(f)) = \frac{b(f)}{1 + b(f)}$$

$$b(f) = \frac{P(f \in F_f | c(f))}{1 - P(f \in F_f | c(f))}$$

If there are no technical effects biasing read coverage, this classifier will (setting aside any issues of overfitting or noise) match the prior of 0.5 and the bias will be computed as 1 for every fragment.

We train separate models for sequence bias at fragment ends, fragment GC content bias, and positional bias in sequence, at each step reweighting each training example according the classification error of the combined model up to that point in a manner similar to AdaBoost (Bishop, 2006b). Each model is alone is relatively simple.

- **Fragment sequence end** Sequence probability is modeled with a variable-order Markov chain model. The order at each position is fit by a greedy hill climbing approach. At each iteration, the order of the chain is incremented at the position that most improves the cross entropy. Iteration stops when no improvement is possible. Separate models are build for the 3' and 5' ends of the fragment, which correspond to first- and second-strand cDNA synthesis, as we observe slightly different bias patterns for the two sides.
- **Fragment GC content** A simple histogram is used where examples are split into bins so that the number of examples in each bin is approximately equal. The number of bins is optimized over.

- **Positional bias** Fragments are binned by relative position and transcript length, a histogram is computed, and bins are linearly interpolated (and extrapolated). The number of transcript length bins and relative position bins are optimized over.

The optimization carried out in each sub-model is performed by evaluating cross entropy on a portion of the training data that is set aside. The remainder of the training data is used for setting parameters to their maximum likelihood value, which is trivial for histograms and Markov chain models.

2.4.3 Inference

Defining, and inferring parameter values in complex probabilistic models is not trivial. More sophisticated MCMC and VI methods are not simple to implement. Once working, making changes to the model during development can be cumbersome and error prone. In short, even a successful ab initio implementation can easily become a morass of fragile and perplexing code.

A variety a tools have been developed to address this difficulty, falling under category of *probabilistic programming*, often built around a specific inference paradigms. These include BUGS (Lunn et al., 2009) and JAGS (Plummer and Others, 2003), based on the Gibbs sampler, as well as a newer generation of tools based around variational inference, including Stan (Carpenter et al., 2016), PyMC3 (Salvatier, Wiecki, and Fonnesbeck, 2016), and Edward (Tran et al., 2016). Each of the probabilistic programming tools in this new generation use some variation of automatic differentiation to allow for clean simpler implementations of complex models by avoiding the need to explicitly define gradient functions. Each also leverages automatic differentiation to implement variations of Hamiltonian Monte Carlo sampling.

Likelihood approximation can be incorporated in models built with any of these tools. We have chosen implement models in Polee using the TensorFlow Probability module, which was based on Edward, for a number of pragmatic reasons. TensorFlow is a Python library, which easily interfaces with Julia, the primary programming language Polee is written in. Secondly, computation and automatic differentiation in TensorFlow can be executed heterogeneously on CPUs and GPUs. As we will see, running model inference on a GPU can offer a dramatic speed up. Furthermore, we found implementing approximated likelihood in TensorFlow, a topic discussed in the next section, to be straightforward and manageable.

Computing the Pólya tree transformation with TensorFlow

Neither Likelihood approximation nor the Pólya tree transformation are standard techniques in probabilistic modeling and inference, so they require some care to adapt to a general purpose tool like Edward.

Implementing the Pólya tree transformation in TensorFlow is difficult. As the name suggests, computation in TensorFlow is organized around operations on tensors (multidimensional arrays, from the programming perspective). Computing the PTT involves traversing a binary tree (from the root to the leaves for the $\mathbb{R}^{n-1} \mapsto \Delta^{n-1}$ transformation, or from the leaves to the root for the inverse). This can be implemented as a series of sparse matrix multiplications (TensorFlow does have support for sparse matrices), each multiplication computing values for lower depth nodes, given their parents. This is slow and uses a great deal of memory computing and storing intermediate results.

A more efficient implementation is necessarily lower-level. To partially overcome the bottleneck of computing the PTT, we implemented the transformation and computation of its gradients in C++ and included it as a custom TensorFlow operation. Rather than rely on TensorFlow's automatic parallelization, the PTT kernel manually multithreads across samples.

Computing the PTT through tree traversal is easy to implement as a CPU compute kernel, but doing so on a GPU, where computation is largely array-based, is much more difficult. For the time being, the PTT kernel remains a bottleneck in inference. Particularly when using a GPU, data must be copied to and from GPU memory to compute the transformation, making the current implementation suboptimal. One possible approach is to use the same tree topology for every sample, which would let us compute the PTT across samples using array operations. This is an idea we revisit in Section 4.1.

2.5 Conclusion

In this chapter we have introduced an approximation to the standard RNA-Seq likelihood function that requires just $3(n-1)$ parameters, and we show it to be surprisingly accurate. The approximation works regardless of the underlying model of sequencing, as captured by transcript probability functions and effective lengths, and is entirely compatible with other approximations such as pseudoalignment. Fitting the approximation is parallelizable and takes a few minutes at most.

Likelihood approximation, however, is not a drop-in replacement for existing analyses, many of which have been devised explicitly to avoid having to do inference on a full model in which sequenced reads are the observations. Though a powerful technique, it necessitates building new analyses from scratch. Fortunately, armed with an efficient substitute for the likelihood, and with modern probabilistic programming techniques, this is not as daunting a task as it may seem.

In the chapter that follows we will show the approximation lets us define probabilistic models in a straightforward manner, with inference that scales to large numbers of samples without the need to resort to stochastic out-of-core techniques like stochastic variational inference (Hoffman et al., 2013).

3

Applications of approximate likelihood

Based on the goodness of fit tests in Section 2.3, the Pólya tree transformation provides an accurate and extremely compact approximation of the true likelihood function. What remains is to demonstrate that approximated likelihood functions are useful for analysis of RNA-Seq. In this section we will build models to evaluate the suitability of approximate likelihood to the task of detecting differential expression, pairwise correlation of transcript expression, and classification.

Throughout this section we will focus on data from the GTEx (GTEx Consortium, 2013) project. Lacking the computational resources to analyze the entire dataset, we focused on just the brain tissues. This consists of 13 tissues, and a total of 1443 samples, and a total of approximately 39 billion reads. Representing the exact likelihood function for this dataset would required approximately 1.5TB of memory. This is a large dataset, but we would ultimately like to scale up this methodology to even larger datasets, a topic we take up in Part 4.

3.1 Tissue classification using random forests

We investigated whether approximate likelihood can improve tissue classification in the GTEx data. We used a random forest model with 200 trees. To accommodate estimations of uncertainty using bootstrap samples and samples from the approximate likelihood, we simply expanded the training data by generating 20 samples for both methods, in place of single point estimates.

We evaluated the model on random training sets of increasing size, and evaluated the performance on 40 samples outside the training set. For the binary classification tasks we averaged performance across 10 random repetitions for the binary tasks, and 5 repetitions for the multiclass task. In Figure 3.1 we see that both bootstrap and approximate likelihood significantly improve classification accuracy over point estimates.

Unsurprisingly, accounting of estimation uncertainty trains a model that better generalizes. In both binary classification tasks, there is very little different between either point estimate method or between either method of accounting for uncertainty, with the exception of posterior mean being slightly better in the more difficult Amygdala versus Hippocampus case. In the full multiclass problem using 17 brain tissues, approximate likelihood does pull ahead of bootstrap samples, and posterior mean above maximum likelihood, but only to a small degree.

From this test we can see, at the very least, that accounting for estimation uncertainty does quite significantly improve tissue type classifiers,

and that approximate likelihood offers some improvements on more complex tasks. We choose random forests here to put methods on equal footings, but approximate likelihood can be used with a much larger class of learning algorithms, since it can be evaluated and differentiated, rather than just sampled from.

3.2 Pairwise correlation between transcripts

Large amounts of available sequencing data have led to an increasing emphasis on deciphering the functional relationships between genes. Co-expression networks are often a preliminary step towards inferring regulatory networks (Markowetz and Spang, 2007). Constructing co-expression networks necessitates estimating pairwise correlation or covariance between genes or transcripts across samples. There are many pairs, but relatively few that are highly correlated or highly anticorrelated, so results can easily be contaminated by false positives, a particular risk with low expression genes, and pairs of similar isoforms.

Co-expression in the the GTEx data was examined by Saha et al. (2017). To control false-positives, aggressive ad hoc filtering was done on the feature set. In addition to including only isoforms with relatively high expression ("isoforms with at least 10 samples with ≥1 TPM and ≥6 reads"), additional filters were applied for isoform variability, mappability, and many features were simply removed to maintain computational tractability. This left only 6000 genes and 9000 isoforms (for comparison, Ensembl annotates nearly 200 thousand transcripts). After filtering, a precision matrix was estimated using a graphical lasso model.

This filtering procedure reduces false-positives, but at the cost of potentially introducing false-negatives. A model affording a more principled accounting of estimation uncertainty would obviate the need for much of this ad hoc filtering.

To explore this idea, we used a more simplistic analysis of co-expression, computing pairwise Spearman correlation matrices across all annotated transcripts, with no filtering whatsoever. To see how the choice of estimate effected the results, we did this with maximum likelihood, posterior mean, bootstrap, and approximate likelihood. To avoid division by zero, and to otherwise slightly moderate the effects of zeros, we added a pseudocount of 0.1 tpm all estimates in the maximum likelihood and bootstrap samples. The uncertainty information provided by bootstrap and approximate likelihood was incorporated by computing the average Spearman correlation across 20 samples.

As with differential expression, there are no plausible gold standard estimates to compare to, so we resorted to using consistency as a proxy for accuracy. We selected one tissue from the GTEx data, cortex, consisting of 118 samples, and computed the correlation matrices. Treating this as ground truth for each respective estimate, we recomputed the matrices using random subsamples of 12 of the 118 samples, and measured the dif-

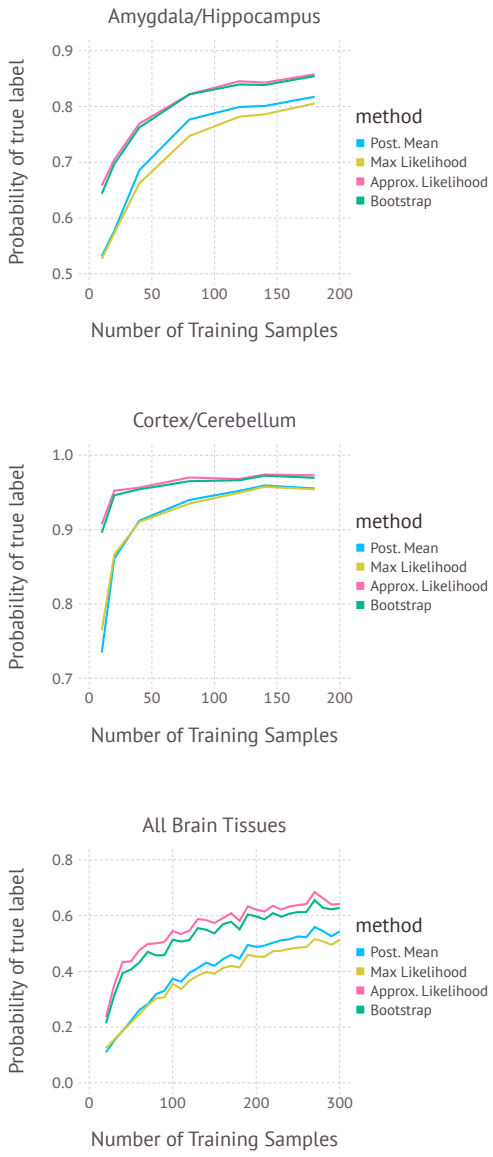

Figure 3.1: Average probability assigned to the true label was computing for training sets of various sizes. The top two plots show two binary classification tasks, the bottom shows the accuracy among 17 brain tissues.

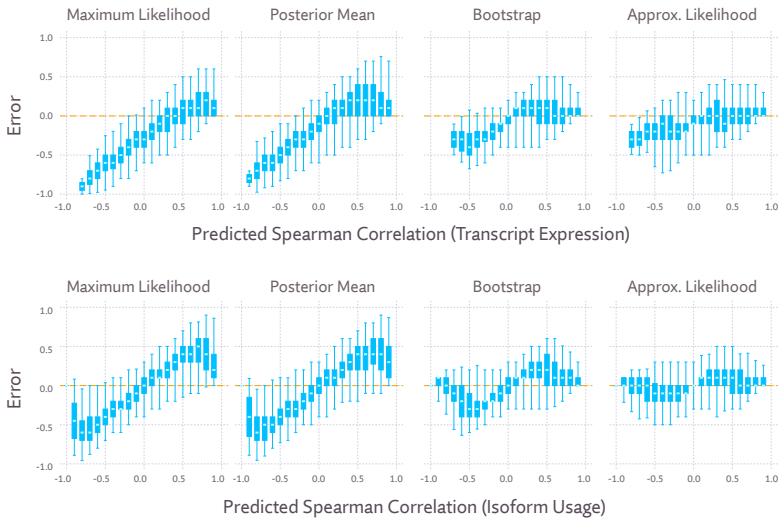

Figure 3.2: Consistency of Spearman correlation estimated using maximum likelihood and posterior mean point estimates, as well as averages across bootstrap samples, and samples from the approximate likelihood. In the upper plots, transcript expression was used, and in the lower, isoform usage (transcript expression divided by the overall expression of its gene). Error here measures the difference between estimates made using all 118 cortex samples, and estimates using a random subset of 12 samples (i.e., "true" correlation minus predicted correlation). These plots show the aggregate error across ten random subsets. Boxplots are drawn with upper and lower whiskers corresponding to the 99th and 1st percentile, respectively.

ference between each element in the matrix. This was repeated 10 times for different random subsamples. The aggregate results are plotted in Figure 3.2.

Looking first at transcript expression (Figure 3.2, upper plots), we see that point estimates tend to produce moderately unreliable estimates of positive correlation, and extremely unreliable estimates of negative correlation. In large part, this is remedied by using either bootstrap estimates or approximate likelihood, with the latter offering a slight improvement. When considering isoform usage (Figure 3.2, lower plots), point estimates are highly unreliable when measuring either positive or negative correlation. Bootstrap estimates improve this, but approximate likelihood estimates are clearly the most reliable here.

3.3 Models of transcript differential expression

Differential expression is determined throughout the paper using a probabilistic linear regression model on log-expression, but with the additional layer of approximate likelihood.

In defining the model we will use the following notation:

m	number of samples in the experiment
n	number of transcripts
k	number of factors/columns in the design matrix
X	m by n matrix of log-expression values
E	m by n noise matrix
A	m by k design matrix
W	k by n matrix of regression coefficients
μ	length n log-expression bias vector
σ	length n transcript standard deviation vector
$r^{(j)}$	some representation of the RNA-Seq reads for the jth sample
$\mathbf{1}$	ones vector (length m in all uses below)

The basic regression model is defined by,

$$\begin{aligned} E_{ij} &\sim \text{Normal}(0, \sigma_j) \\ X &= \mathbf{1}\mu^T + AW + E \\ r^{(j)}|X &\sim \text{ApproximateLikelihood}(\text{softmax}(X_j^T)) \end{aligned}$$

where X_j^T is the jth row of X, transposed.

The softmax functions transforms the log-expression to relative expression. There are some subtleties of this move noted below. The bias vector μ has a Normal prior with a large variance.

Apart from the inclusion of the extra layer $r|X$, treating the reads as observed and expression values as unobserved, this is a fairly standard log-

linear model of gene expression, similar to the one described by Smyth (2004), for example.

3.3.1 Prior on regression coefficients

Operating under the prior expectation that most transcripts are not differentially expressed, W is given a sparsity inducing prior. Elements of W_{ij} have the prior given by,

$$\begin{aligned} W_{ij}|\lambda_{ij} &\sim \text{Normal}(0, \lambda_{ij}) \\ \lambda_{ij}|\eta_{ij} &\sim \text{HalfCauchy}(0, \eta_{ij}) \\ \eta_{ij} &\sim \text{HalfCauchy}(0, 1) \end{aligned}$$

This is a version of what Bhadra et al. (2017) describe as the "Horseshoe+" prior.

3.3.2 Modeling variance

Transcript expression standard deviation σ is shrunk towards standard deviation estimates of transcripts with similar expression.

To do this we choose $h = 15$ "knots" at expression values, equally spaced between the minimum and maximum, and compute weights $u_{j\ell}$ between every transcript j and knot ℓ, using the squared difference between the knot's expression value and the transcript expression averaged across samples. These weights are pre-computed using initial approximate posterior mean expression estimates, obtained by applying the approximation's transformation to a zero vector. Each knot has associated parameters α_ℓ, β_ℓ.

$$\begin{aligned} \alpha_\ell &\sim \text{HalfCauchy}(0, 1) \\ \beta_\ell &\sim \text{HalfCauchy}(0, 1) \end{aligned}$$

Per-transcript standard deviation is Inverse-Gamma distributed with parameters computed as weighted sums of these.

$$\sigma_j \sim \text{InverseGamma}\left(\sum_{\ell=1}^{h} u_{j\ell}\alpha_\ell, \sum_{\ell=1}^{h} u_{j\ell}\beta_\ell\right)$$

3.3.3 Scale penalization

Because RNA-Seq measures relative expression, detecting changes is absolute expression involves additional assumptions. Various normalization

schemes have been proposed to overcome this (Bullard et al., 2010). Other methods avoid the problem by looking for compositional changes (McGee et al., 2019). Similar to the normalization schemes, our model attempts to find changes in absolute expression by assuming most transcripts or genes are maintaining relatively constant expression (or constant lack of expression).

Because of the the softmax transformation, log-expression values are not on a fixed scale, which is to say for a given sample i, $\sum_{j=1}^{n} \exp(x_{ij})$ is not constrained to sum to any particular number. How relative scales between samples are inferred determines how changes in transcript composition are interpreted as changes in expression.

Because of the sparsity inducing prior, higher probability is achieved when fewer transcripts are differentially expressed. If this assumption is reasonable, relative scaling will be arrived at automatically so as to minimize differences between samples. Of course, there are circumstances when this assumption fails, particularly in the presence of large changes in the overall quantity of RNA between groups of cells.

In practice we found it helpful for inference to additionally penalize these exponent sums from straying too far. To find reasonable values, we use initial posterior mean estimates to estimate relative scales that minimize differential expression of highly expressed transcripts, then include an additional Normal prior in the model on these exponent sums, centered on the scale estimates. This is simply one approach to putting an informative prior on relative scale.

3.3.4 Calling differential expression using minimum effect size

When calling differential expression with this model, effect size is always considered explicitly. Because the model is continuous, the posterior probability of no change in expression is zero. We instead look for transcripts or genes with a sufficiently large posterior probability of the effect size being above some threshold. Unlike most null-hypothesis test models which typically (though, not necessarily) adopt a null hypothesis of zero change, there is an extra threshold that must be chosen: the minimum effect size. This is a slight complication, but has the advantage of being up front about what is considered a potentially interesting effect.

Similarly to the procedure in Section 2.3.4, we call differential expression using *minimum log2 fold-change*, which we define for a given transcript as the number δ where

$$P(\text{absolute log2 fold change} \geq \delta) = c$$

Typically we choose $c = 0.9$. This number can be thought of then as determining differential expression by looking at a conservative estimate of effect size, which differs from common approaches that are blind to effect size, and call differential expression in contrast to a null hypothesis that posits no change whatsoever.

3.3.5 Modeling gene expression and isoform usage

An alternative to detecting changes in transcript expression, is to instead consider changes in gene expression and isoform usage. If we define a gene as a set of transcripts, and suppose that each transcript is assigned to exactly one gene, then these genes represent a partition of the set of transcripts. This partition of the transcripts also defines a bijection between transcript expression and a gene expression combined with isoform usage.

Let $g_1, \dots, g_k$ be k genes each represented by a set of transcript indexes. In the terminology of Aitchison (1986a), it is a $k-1$ partition of Δ^{n-1}, defined by an amalgamation $y^{\mathcal{G}}$ giving the relative expression of each gene

$$y^{\mathcal{G}} = \left(\sum_{j \in g_i} y_j \; ; \; i = 1 \dots, k \right)$$

along with subcompositions $y_1^{\mathcal{J}}, \dots, y_k^{\mathcal{J}}$ giving the isoform usage of every gene

$$y_i^{\mathcal{J}} = \left(\frac{y_j}{y_i^{\mathcal{G}}} \; ; \; j \in g_i \right)$$

Multiplying each isoform usage vector by the gene expression recovers the transcript expression. We will denote this de-partitioning operation using $\odot$, so

$$y = y^{\mathcal{J}} \odot y^{\mathcal{G}}$$

Partitioning the simplex like this offers an alternative to the direct regression over transcript expression described in Section 3.3. We can instead perform joint regression over gene expression and isoform usage. We introduce separate matrices $X^{\mathcal{G}}$ for the gene log-expression, and $X^{\mathcal{J}}$ for the unscaled log isoform usage. These are modeled as X is in Section 3.3, while the likelihood is

$$\begin{aligned} r^{(j)} | X \sim \text{ApproximateLikelihood}(\\ \text{softmax}((X_j^{\mathcal{G}})^T) \odot \text{softmax}^{\mathcal{G}}((X_j^{\mathcal{J}})^T)) \end{aligned}$$

Importantly, the softmax operation applied to $(X_j^{\mathcal{J}})^T$, denoted $\text{softmax}^{\mathcal{G}}$ is done according to the gene partition, rather than across the entire vector.

3.4 Approximate likelihood models outperform other models in identifying differentially expressed transcripts

We expanded an analysis performed in Pimentel et al. (2017), which demonstrated superior performance when using sleuth to call differential expression, especially at the transcript level, using simulated data. We include the likelihood approximation method described here in two ways. First we used the Bayesian regression model described in the previous section,

which is labeled "polee" in the results. Second, we generated samples from the approximated likelihood to mimic the bootstrap sampling output of kallisto, and used this as input to sleuth. This approach is labeled polee/sleuth.

The three simulations, labeled "gfr", "isoform", and "gcd" correspond to three sets of assumptions. The gfr simulation matches simulated effect sizes to those detected by Cufflinks (Trapnell et al., 2012) in a reference dataset. The "gcd" simulation adopts the assumption that gene expression is perturbed while holding isoform mixtures fixed, and the "isoform" simulation assumes the expression values of transcripts are perturbed independently of each other. Results from these simulations are shown in Figures 3.3 and 3.4

On the column to the right, we see that differential transcript expression tests show both polee and polee/sleuth with significantly improved performance over other methods (i.e., greatly increased recall at the same fdr levels). These results suggest that samples drawn in proportion to the likelihood are more informative than bootstrap samples for this task, and that most informative of all is actually including the likelihood function, or its approximation, in the model. When detecting gene-level differential expression, polee very slightly trails sleuth, which exceeds the performance of all the other methods. Oddly, using likelihood approximation samples with gene-level sleuth analysis (labeled "polee/sleuth" in Figure 3.3) does not yield similar performance. This may be due to sleuth's filtering heuristics being poorly calibrated for samples from the posterior, rather that bootstrap samples.

3.5 Differential expression calls made with approximate likelihood are more internally consistent

Evaluating the accuracy of differential expression calls is fraught by a lack of any agreed upon ground truth. Simulations are one way around that issue, but assume the model of expression and sequencing used to generate the simulated reads is a good approximation of reality. Here we explore another option: calling differential expression with a large number of samples, then testing the ability to recover the same calls with a small number of samples. This avoids putting our faith in the verisimilitude of a simulation, but to be a reasonable proxy for accuracy it instead assumes the model converges to the correct result with enough replicates. Models can of course be both perfectly internally consistent and totally wrong, but taken together with Section 3.4 makes a case for the accuracy.

Using brain tissue data from GTEx (GTEx Consortium, 2013), we compared the same regression model using four different approaches to transcript quantification: maximum likelihood estimates, maximum likelihood with bootstrap variance estimates, posterior mean estimates generated from the likelihood approximations, and the full approximated likelihood. In addition to running regression with all 13 brain tissues, we also evaluated pairwise differential expression between a transcriptionally similar pair of tis-

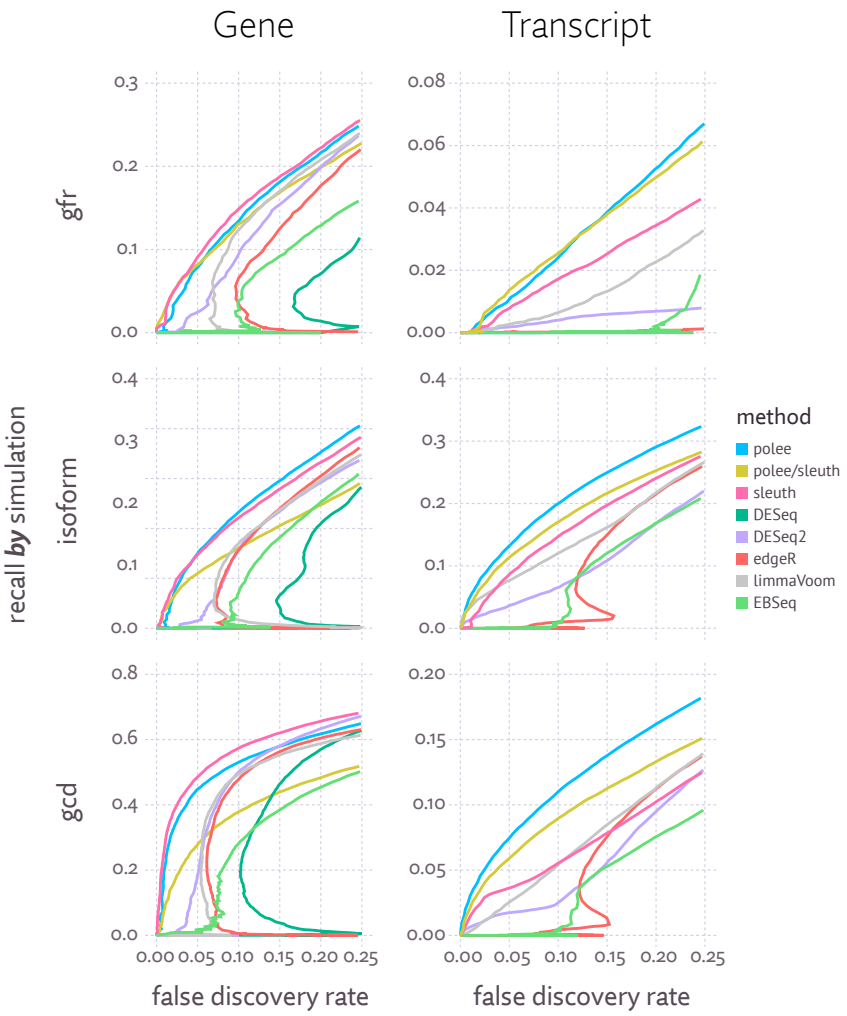

Figure 3.3: Plots of false discovery rate (i.e., the proportion of differential expression call which are incorrect) versus recall (i.e., proportion of differentially expressed features identified) when calling differential gene or isoform expression on three simulations (gfr, isoform, and gcd, explained in the text), with a variety of methods. Each line represents the aggregate FDR-recall curve across 20 replicates of the simulation, each consisting of six samples split across two conditions.

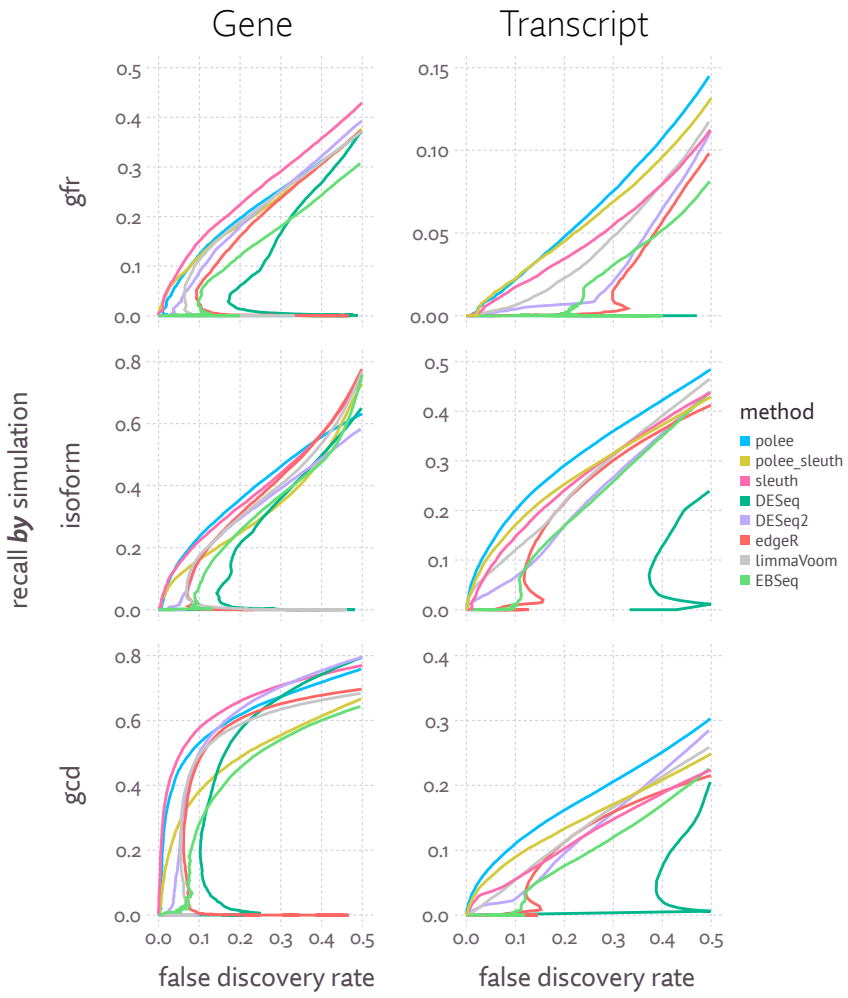

Figure 3.4: A zoomed out version of Figure 3.3 showing false discovery rate as a function of recall on simulation benchmarks.

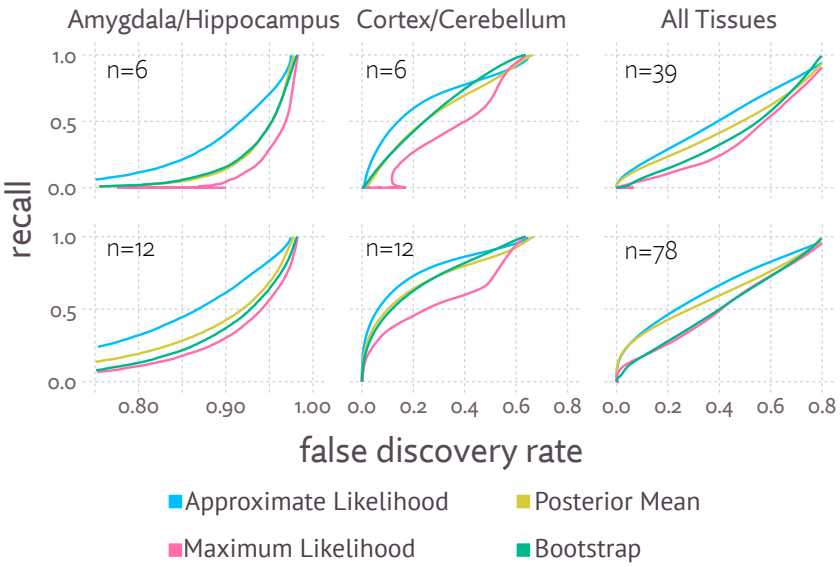

Figure 3.5: FDR/recall curves for subsets of the GTEx data, using the variants of the same regression model, with expression modeled as either maximum likelihood or posterior mean point estimates, bootstrap estimates of variance, or approximate likelihood. Note the x-axis has been adjusted for each column to show the details of the curve.

sues (hippocampus and amygdala), and a transcriptionally divergent pair (cortex and cerebellum).

Each run was compared to the same regression model run with a larger number of replicates (96 for the pairwise tests, and 1443 with all tissues). These tests were run with 10 different random subsets to draw the aggregate FDR/recall curves in Figure 3.5.

The results show broadly that the full approximate likelihood model outperforms point estimates and bootstrap estimates. Posterior mean estimates offer an improvement over maximum likelihood estimates, and appear to begin to catch up to the approximate likelihood approach when a large enough number of samples is used. An important caveat is that producing posterior mean point estimates either involves MCMC, or a variational inference, so in generating the estimates, there is little or no performance advantage to using posterior mean estimates instead of approximate likelihood. In the pairwise tests, bootstrap estimates perform similarly to using posterior mean point estimates, while underperforming when using all samples.

The amygdala versus hippocampus test shows very poor performance for all the involved methods, as the differences that do exist between these tissues are small or inconsistent, so they are only reliably detectable with a large number of samples. Nonetheless, approximate likelihood does make better use of the limited data, showing a clear improvement.

4

Scaling up approximate likelihood analysis

The approximate likelihood method described and deployed in the preceding sections can lead to more accurate results by making inference on otherwise intractably large full probabilistic models possible. It has limitations though. Though we were able to run a regression model with 1443 samples and 39 billion reads, the memory requirements were such that we were unable to run a regression of this size on GPUs. Running it on CPUs required approximately 200GB of memory and two days to finish. As full isoform-resolution single cell RNA-Seq is becoming increasing efficient (Hagemann-Jensen et al., 2020), scaling the method up to handle tens or hundreds of thousands of cells should be a priority.

In the following sections we will propose two ideas to overcome these limitations, which should let approximate likelihood models to the scale of current single-cell RNA-Seq protocols.

In Section 2.4.3 we noted that computing a different Pólya tree transformation, each with a different topology, for every sample is a major bottleneck in analysis. In this chapter, we show that an alternative heuristic can be used to find a topology based on sequence similarity to allow each sample to use the same topology, which should allow for much faster computation.

Secondly, we explore how to make inference far more efficient by exploiting the result in Theorem 2.2.11. In short, when a Dirichlet prior is used for the the transcript expression, a conjugate relationship can be exploited to efficiently sample from the posterior, without resorting to MCMC or variational inference.

4.1 Fixed topologies with sequence based heuristics

Lacking any tractable method of finding the the optimal tree structure for a Pólya tree transformation, we have relied a hierarchical clustering heuristic, using the Jaccard index (i.e., size of the intersection divided by size of the union) between sets of compatible reads as a similarity measurement. Each In the models we ran, each RNA-Seq sample has its own tree topology and transformation.

This has some implications for the efficiency of the overall likelihood approximation approach. While the time needed fit the tree topology tends to account for only around 15% of the overall time, half of the space needed to store and represent the approximated likelihood is spent on encoding the tree topology. For example, a mouse sample with $n = 138,929$ transcripts requires about 4.4MB to represent the approximated likelihood, 2.2MB of which is needed for the tree topology.

This becomes a more pressing issue when it comes to inference. In a model encompassing many RNA-Seq samples, each factor of the full approximated likelihood involves a different transformation. So when evaluating the approximate likelihood, a separate tree must be traversed (from the root) for each sample, then when evaluating gradients, we must again traverse (from the leaves) a separate trees for each sample. This is a major bottleneck when running inference on GPUs. If we were able to use a single tree topology, not only would we save time approximating likelihood functions and save space representing the approximation, but evaluating the approximated likelihood and its gradients could be implemented far more efficiently using SIMD (single instruction, multiple data) or MIMD (multiple instruction, multiple data) techniques, which would dramatically speed up inference.

With this in mind, we investigated replacing the per-sample read set Jaccard index heuristic, with a per-genome or per-transcriptome k-mer set Jaccard index heuristic. Instead of estimating a new topology for each RNA-Seq sample, sequence-based k-mer set heuristic would result in the same tree topology for every sample that uses a common set of transcripts.

4.1.1 Tractable hierarchical clustering of transcripts by k-mer sets

Clustering by transcript k-mer sets is straightforward in principle. Sets of k-mers are constructed for every transcript, the Jaccard index between each pair of transcripts is then computed, then hierarchical clustering is performed. Starting with each transcript in its own single-node subtree, each step of the hierarchical clustering algorithm joins a pair of subtrees with the largest Jaccard index, computes the union of their k-mer sets, then evaluates the Jaccard index of this union k-mer set against all other subtrees.

Unfortunately, a naive implementation is $O(n^3)$, where n is the number of transcripts. This analysis even charitably assumes a constant number of k-mers per transcript. There are 126,374,122 distinct 32-mers present in Ensembl's human gene annotations. Each transcript, or subtree constructed during clustering, could have a significant fraction of these k-mers, so this constant, even if treated as such, can be quite large. To make this tractable, we must more efficiently compute the Jaccard index between pairs, and reduce the number of pairs that are compared.

A common way of efficiently estimating Jaccard index is the probabilistic hashing technique MinHash (Broder, 1997). This approach works by hashing elements of each set and keeping track of the minimum value hash value. It was shown that the probability of these minimum hash values being equal between two sets is equal to the Jaccard index. Using multiple hash functions and computing the proportion of collisions then gives an arbitrarily accurate estimate of the Jaccard index.

A number of variations of this basic technique exist. Notably, Li, Owen, and Zhang (2012) introduced *one permutation hashing* which achieves

the same goal while only requiring a single hash function. They achieve this by partitioning the range of possible hash values (typically 64-bit integers, so $\{0, \ldots, 2^{64} - 1\}$) into some number of bins. Every element of a set is hashed and the minimum value that falls within each bin is recorded. The probability of two sets colliding in a single bin is again equal to the Jaccard index, so the proportion of bins colliding is an estimate of the Jaccard index that grows increasingly accurate with more bins. To aid intuition: in the extreme case of 2^{64} bins, the presence or absence of every possible k-mer hash value is recorded, yielding the exact Jaccard index, if the hash function is a random permutation and thus injective.

Not only does one permutation hashing accelerate Jaccard index calculations, but it can be used to reduce the number of pairs being compared. If no bins collide, the Jaccard index estimate is zero, in which case we need not compare that pair. To avoid making $O(n^2)$ comparisons, each bin in the data structure can be hashed or sorted to find all colliding pairs, thus avoiding ever comparing pairs of transcripts that have approximately zero k-mers in common.

The final operation we need to implement our hierarchical clustering technique is an efficient way of computing the union of sets of k-mers. In one permutation hashing this is done simply by taking minimum in each bin between two set approximations.

In practice, we use 32-mers (encoded in 64-bit integers), and 200 bins.

4.1.2 The k-mer heuristic produces an equally accurate approximation

We repeated some benchmarks using per-transcriptome k-mer heuristic in place of the existing per-sample read set heuristic. We see in both the p-value goodness of fit test (Figure 4.1) and the sleuth simulation benchmarks (Figure 4.2) results that are nearly identical between the two methods. The Jaccard index between read sets and between k-mer sets are both ways of getting at transcript sequence similarity. The fact that they are so close is an encouraging sign that the approximation is near its limit.

4.2 Efficient sampling of transcript expression

In a regression model like the one described in Section 3.3, most of the time and memory needed by inference is in estimating the posterior for the transcript expression values. In this section we show how, under certain circumstances we can sample directly from this posterior without resorting to MCMC or variational inference.

The biggest advantage of being able to efficiently generate samples from the transcript expression posterior is that it will make out of core inference far more efficient. Variational inference and MCMC both require retaining some state for each variable. In the case of MCMC, this is just the last

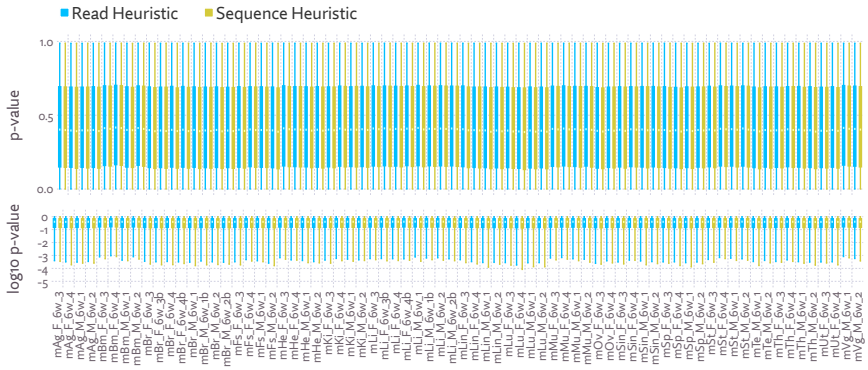

Figure 4.1: On goodness-of-fit benchmarks, the sequence heuristic perform nearly indistinguishably from the read heuristic. Here 1000 samples from the Gibbs sampler and 1000 samples draw from the likelihood approximations are compared using a signed-rank null-hypothesis test. If the likelihood approximation is a perfect fit, we should expect to see a uniform distribution of p-values. A poor fit will produce an abundance of low p-values. Boxplots are drawn with upper and lower whiskers corresponding to the 99th and 1st percentile, respectively.

sample, with variational inference, we need to store parameters for the approximating distribution.

Training large deep neural networks and huge datasets works by sampling minibatches of the training data and performing stochastic gradient descent. In principle, we can do the same. Random subsets of the samples can be cycled in and out of memory, updating the regression parameters. The extra burden of having to read and write this state for each sample and each epoch of the training algorithm would significantly slow this down. Exploiting the the relationship between the Dirichlet distribution and the Pólya tree transformation lets us sample directly, avoiding this hindrance.

4.2.1 Exploiting Dirichlet quasi-conjugacy

Conjugate prior relationships are often exploiting in Bayesian inference to speed inference. If the likelihood and prior are carefully chosen, the posterior is of convenient distribution family. For example, a common tactic in count-based models is exploiting Gamma-Poisson conjugacy (e.g., in scVI, a popular scRNA-Seq tool (Lopez et al., 2018a)). While there does not seem to be a distribution family that is a conjugate prior with the RNA-Seq likelihood function, our likelihood approximation scheme can yield a conjugate-like relationship which can be exploited in a similar way.

Since our likelihood approximation is only approximately proportional, not approximately equal, we are not necessarily interested in an exact conjugate prior relationship, but only a prior from a family that will yield a pos-

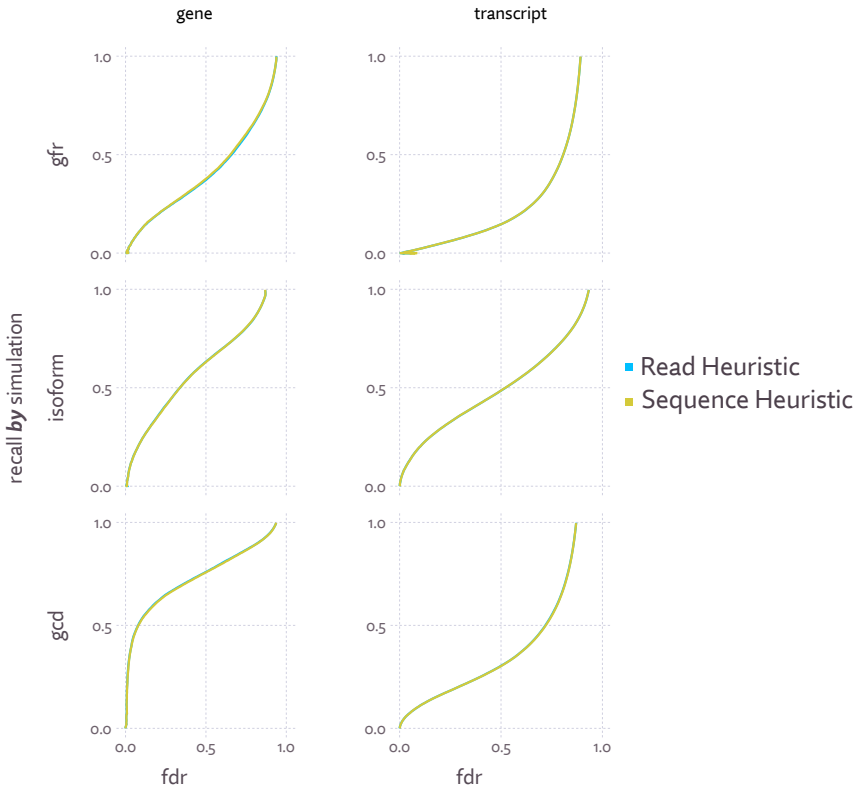

Figure 4.2: In calling differential expression at the gene and transcript level, the sequence heuristic is again nearly indistinguishable from the read heuristic. False discover rate is plotted against recall when calling differential expression on simulated RNA-Seq experiments. Three simulation configurations were used (labeled "gfr", "isoform", and "gcd"), and each was run 20 times. The curves were drawn by averaging these results.

terior that is proportional to a distribution the same family. We call this *quasi-conjugacy*.

Definition 4.2.1 (Quasi-conjugacy). *Two probability density functions $f(\cdot;\theta)$, $g(\cdot;\phi)$ with parameter vectors θ, ϕ, respectively, are* quasi-conjugate *iff for all θ, ϕ, there exists a ϕ', where for all x*

$$f(x;\theta) \cdot g(x;\phi) \propto g(x;\phi')$$

Quasi-conjugacy resembles the more familiar notion of conjugacy (e.g., Beta-Binomial conjugacy), but is in terms of proportionality rather than equality. Since for most inference tasks proportionality is all the matters, with a quasi-conjugacy relationship we can often sample directly from the posterior, bypassing the need for MCMC or variational inference.

In what follows, we will show that if we choose Beta distributions as the basis for the approximation, the Dirichlet distribution is quasi-conjugate, letting us very efficiently sample from the posterior. This will follow in three steps:

1. First we show that the Beta distribution family is quasi-conjugate with itself.
2. We then show that the distribution family resulting from transforming Beta distributions using a Pólya tree transformation, which we will call Beta-PTT, is also quasi-conjugate with itself.
3. Finally, Theorem 2.2.11 lets us write the Dirichlet distribution as just such a transformation of Beta random variables, thus showing that Dirichlet is Beta-PTT and thus conjugate with a Beta-PTT approximate likelihood.

Lemma 4.2.2 (Beta/Beta quasi-conjugacy). *Let $f(x;\alpha,\beta)$ be the Beta distribution probability density function. For any $\alpha_1, \beta_1, \alpha_2, \beta_2 \in \mathbb{R}_+$, where $\alpha_1 + \alpha_2 > 1$ and $\beta_1 + \beta_2 > 1$*

$$f(x;\alpha_1,\beta_1) \cdot f(x;\alpha_2,\beta_2) \propto f(x;\alpha_1+\alpha_2-1,\beta_1+\beta_2-1)$$

Proof. Let $\alpha_1, \beta_1, \alpha_2, \beta_2 \in \mathbb{R}_+$ with $\alpha_1 + \alpha_2 > 1$ and $\beta_1 + \beta_2 > 1$, then

$$\begin{aligned}
f(x;\alpha_1,\beta_1) \cdot f(x;\alpha_2,\beta_2) &= \left(\frac{x^{\alpha_1-1}(1-x)^{\beta_1-1}}{B(\alpha_1,\beta_1)}\right)\left(\frac{x^{\alpha_2-1}(1-x)^{\beta_2-1}}{B(\alpha_2,\beta_2)}\right) \\
&= \frac{x^{(\alpha_1+\alpha_2-1)-1}(1-x)^{(\beta_1+\beta_2-1)-1}}{B(\alpha_1,\beta_1)B(\alpha_2,\beta_2)} \\
&\propto \frac{x^{(\alpha_1+\alpha_2-1)-1}(1-x)^{(\beta_1+\beta_2-1)-1}}{B(\alpha_1+\alpha_2-1,\beta_1+\beta_2-1)} \\
&= f(x;\alpha_1+\alpha_2-1,\beta_1+\beta_2-1)
\end{aligned}$$

□

Note that this quasi-conjugacy relationship does come with the restrictions that the parameters have to remain positive, so the resulting Beta posterior is well defined. We will have to take this restriction into account in the following two theorems.

To simplify things for the theorems that follow, we will introduce the following notation giving vectors of the number of internal nodes in the left and right subtree of each nodes.

$$\mathcal{L} = (|\,\text{internal}(\text{left}(i))| : i = 1, \dots, n-1)$$

$$\mathcal{R} = (|\,\text{internal}(\text{right}(i))| : i = 1, \dots, n-1)$$

Lemma 4.2.3 (Beta-PTT/Beta-PTT quasi-conjugacy). *Let $q_T(\cdot; \alpha, \beta)$ denote the probability function resulting from transforming $n-1$ random variables with distributions Beta(α_i, β_i) for $i = 1, \dots, n-1$ using a Pólya tree transformation T. That is,*

$$q_T(\cdot; \alpha, \beta) = |\det J_{T^{-1}}| \prod_{i=1}^{n} f(T^{-1}(x)_i; \alpha_i, \beta_i)$$

where $f(\cdot)$ is the Beta density function.

Let $\alpha, \beta, \alpha', \beta' \in \mathbb{R}_+^{n-1}$, with $\alpha_i + \alpha'_i - \mathcal{L}_i > 1$ and $\beta_i + \beta'_i - \mathcal{R}_i > 1$ for all i. Let T be any Pólya tree transformation. Then, for any $x \in \Delta^{n-1}$,

$$q_T(x; \alpha, \beta) \cdot q_T(x; \alpha', \beta') \propto q_T(x, \alpha + \alpha' - \mathcal{L} - \mathbf{1}, \beta + \beta' - \mathcal{R} - \mathbf{1})$$

Proof. This fact follows nearly directly from exploiting Beta/Beta quasi-conjugacy, with the one complication that we have an extra Jacobian determinant to deal with. Fortunately the Jacobian determinant is of a form (Corollary 2.2.5) that can be absorbed into the Beta parameters.

$$
\begin{aligned}
& q_T(x;\alpha,\beta)\cdot q_T(x;\alpha',\beta') \\
&\quad = |\det J_{T^{-1}}|^2 \prod_{i=1}^{n-1} f(T^{-1}(x)_i;\alpha_i,\beta_i)\cdot f(T^{-1}(x)_i,\alpha_i',\beta_i') \\
&\quad \propto |\det J_{T^{-1}}|^2 \prod_{i=1}^{n-1} f(T^{-1}(x)_i;\alpha_i+\alpha_i'-1,\beta_i+\beta_i'-1) && \text{By Theorem 4.2} \\
&\quad \propto |\det J_{T^{-1}}|^2 \prod_{i=1}^{n-1} (T^{-1}(x)_i)^{\alpha_i+\alpha_i'-2}(1-T^{-1}(x)_i)^{\beta_i+\beta_i'-2} \\
&\quad = |\det J_{T^{-1}}| \left(\prod_{i=1}^{n-1} (T^{-1}(x)_i)^{\mathcal{L}_i}(1-T^{-1}(x)_i)^{\mathcal{R}_i}\right)^{-1} && \text{By Corollary 2.2} \\
&\qquad \left(\prod_{i=1}^{n-1} (T^{-1}(x)_i)^{\alpha_i+\alpha_i'-2}(1-T^{-1}(x)_i)^{\beta_i+\beta_i'-2}\right) \\
&\quad = |\det J_{T^{-1}}| \prod_{i=1}^{n-1} (T^{-1}(x)_i)^{\alpha_i+\alpha_i'-\mathcal{L}_i-2}(1-T^{-1}(x)_i)^{\beta_i+\beta_i'-\mathcal{R}_i-2} \\
&\quad \propto |\det J_{T^{-1}}| \prod_{i=1}^{n-1} f(T^{-1}(x)_i;\alpha_i+\alpha_i'-\mathcal{L}_i-1,\beta_i+\beta_i'-\mathcal{R}_i-1) \\
&\quad = q_T(x;\alpha+\alpha'-\mathcal{L}-\mathbf{1},\beta+\beta'-\mathcal{R}-\mathbf{1})
\end{aligned}
$$

□

Theorem 4.2.4 (Dirichlet/Beta-PTT quasi-conjugacy). *Let $f_D(\cdot;\gamma)$ denote the Dirichlet probability density function parameter vector γ.*

For any $\alpha,\beta\in\mathbb{R}_+^{n-1},\gamma\in\mathbb{R}_+^n$ with $\gamma_i\geq 1$ for each i,

$$f_D(x;\gamma)\cdot q_T(x;\alpha,\beta)\propto q_T(x;\alpha+a-\mathcal{L}-\mathbf{1},\beta+b-\mathcal{R}-\mathbf{1})$$

where

$$
\begin{aligned}
a_i &= \sum_{j\in \textit{leaves}(\textit{left}(i))} \gamma_{j-n+1} \\
b_i &= \sum_{j\in \textit{leaves}(\textit{right}(i))} \gamma_{j-n+1}
\end{aligned}
$$

Proof. For any Dirichlet parameter vector γ where $\gamma_i\geq 1$ for each i, from Theorem 2.2.11 we know that

$$f_D(x;\gamma)=q_T(x;a,b)$$

We can then invoke Lemma 4.2.3 to prove this theorem, but must first show that $\gamma_i\geq 1$ is enough to ensure its condition that $\alpha_i+a_i-\mathcal{L}_i>1$ and $\beta_i+b_i-\mathcal{R}_i>1$.

Since T is represented by a complete binary tree, each subtree is also a complete binary tree. The number of internal nodes is a complete binary tree is one less than the number of leaves, thus

$$|\,\text{leaves}(\text{left}(i))| = |\,\text{internal}(\text{left}(i))| + 1 = \mathcal{L}_i + 1$$

$$|\text{leaves}(\text{right}(i))| = |\text{internal}(\text{right}(i))| + 1 = \mathcal{R}_i + 1$$

Combining this with $\gamma_i \geq 1$ we have

$$a_i = \sum_{j \in \text{leaves}(\text{left}(i))} \gamma_{j-n+1} \geq \mathcal{L}_i + 1$$

$$b_i = \sum_{j \in \text{leaves}(\text{right}(i))} \gamma_{j-n+1} \geq \mathcal{R}_i + 1$$

This in turn implies

$$\alpha_i + a_i - \mathcal{L}_i \geq \alpha_i + 1 > 1$$

$$\beta_i + b_i - \mathcal{R}_i \geq \beta_i + 1 > 1$$

By Lemma 4.2.3 we have that

$$f_D(x;\gamma) \cdot q_T(x;\alpha,\beta) \propto q_T(x;\alpha + a - \mathcal{L} - \mathbf{1}, \beta + b - \mathcal{R} - \mathbf{1})$$

□

To restate the situation: if we build a Beta-PTT likelihood approximation, and a model using a Dirichlet prior, Corollary 4.2.4 lets us efficiently sample from the posterior by generating samples from the resulting posterior Beta-PTT distribution. Sampling from a Beta-PTT consists simply of sampling from the underlying Beta distributions and applying the transformation.

The only issue that remains is how to use the Beta distribution, which is not typically amenable to the reparameterization trick, as the base distribution for the approximations. Fortunately, there are options, which we will discuss in the following section.

4.2.2 Beta distributions and the Pólya tree transformation

In Section 2.2.6 we explored possible base distributions to transform with the Pólya tree transformation to construct a family of approximations. We restricted ourselves to those that are amenable to the reparameterization trick, which allows for more efficient optimization of the approximation. Exploiting the conjugate-like relationship with the Dirichlet distribution requires using an approximation based on the Beta distribution, which does not have a neat reparameterization. Fortunately, some options exist to overcome this.

One alternative route to minimizing the KL-divergence is through the *score function* (Ranganath, Gerrish, and Blei, 2014), which gives a noisy gradient for the evidence lower bound. Recall that maximizing the evidence lower bound is equivalent to minimizing the KL-divergence. The score function avoids having to reparameterized, but it is in practice a high variance estimate, which means many more samples must be generated to successfully approach the approximation's optimal parameterization.

Concurrently Jankowiak and Obermeyer (2018) and Figurnov, Mohamed, and Mnih (2018) explored another idea to work around the limitations of the reparameterization trick and enable backpropagation through distributions that cannot be easily reparameterized. They note that any univariate random variable X, distributed according to a distribution with parameters ϕ and density function q_ϕ can be reparameterized as its inverse CDF F_ϕ^{-1} applied to a uniform random variable $z \sim U(0, 1)$. That is

$$x = F_\phi^{-1}(z)$$

Often, as is the case with the Beta distribution, F_ϕ^{-1} does not have an analytical formula, so is not easy to compute or differentiate. What both papers note is that the gradients with respect to ϕ can be instead be computed as

$$\nabla_\phi x = -\frac{\nabla_\phi F_\phi(x)}{q_\phi(x)}$$

Unfortunately, the Beta CDF function is not easily differentiated. Jankowiak and Obermeyer (2018) propose numerical approximations, while Figurnov, Mohamed, and Mnih (2018) express Beta distributed variables as a function of two Gamma distributed variables and use numerical approximations for the Gamma distribution. Experiments in both papers show good performance compared with alternative methods.

5

Conclusion

In the previous chapters, we presented the Pólya tree transformation as an attempt to rethink how RNA-Seq analysis is done. By fully generalizing the space stick breaking transformations, we have a versatile tool for constructing distributions over compositions. This lets us very accurately approximate the full transcript-level RNA-Seq likelihood function using as few as 2 parameters per transcript. We then show that, despite this being a well studied problem, this method significantly improves upon the state of the art when detecting transcript differential expression, and shows promising results when examining classification and coregulation. Above all, these results show the importance of accounting for uncertainty when attempting to quantify features that produce ambiguous signal, such as isoforms and homologous genes.

As RNA-Seq technology expands to sequence larger and larger numbers of cells, full probabilistic models that account for estimation uncertainty become increasingly out of reach. Two-step analyses that do attempt account for uncertainty do so by generating samples — whether through bootstrap or MCMC — which effectively inflate the size of data, making downstream analysis many times more expensive. Alternatively, variational methods that have been proposed in the past often operate by method of moments fitting to samples, or through minimizing the KL divergence to inflexible distribution families. What we propose is a very accurate approximation, with as little as two parameters per transcript, offering a viable path towards very large scale, uncertainty aware, isoform-level RNA-Seq analysis.

In the final chapter we laid out that path. Heuristics based on sequence similarity can recapitulate the results of the read alignment heuristic, meaning a one transformation fits all approach is perfectly viable. This further reduces the space required by 50% and has the potential to dramatically speed up evaluating the likelihood. Secondly, models built to exploit the quasi-conjugacy property between the Dirichlet distribution and Beta distributions under the Pólya tree transformation can sample directly from the transcript expression posterior, speeding up inference.

Beyond having a promising outlook in transcriptomics, the Pólya tree transformation is more broadly a new take on compositional data analysis that may have applications in a variety of fields. As demonstrated, the transformation when applied to Beta distributions generalizes Dirichlet distribution. Any place where Dirichlet distributions are used, but the data has natural subcompositions may benefit from using this larger family of distributions. This includes other sequencing experiments, which are often also compositional in nature, but also in phylogenetic analyses where

the data naturally forms a tree. We look forward to uncovering more of these applications in the future.

A

Some notes on the effective length transformation

The rate at which reads are sampled from a transcript is related to its length (a longer transcript has more possible fragments to be sequenced), and potentially other factors that bias the library preparation and sequencing process. The standard approach is then to assume the probability of observing a read from transcript j is proportional to its expression x_i and weight $\mathbb{L}_j \in \mathbb{R}_{>0}$ referred to as an *effective length*.

For an arbitrary read i, the probability it is from transcript j, prior to observing the read is then

$$P(z_{ij} = 1|x) = x'_j = \frac{\mathbb{L}_j x_j}{\sum_{k=1}^{n} \mathbb{L}_k x_k} = \frac{\mathbb{L}_j x_j}{\mathbb{L}x^\top}$$

This effective length scaling is an unfortunate kink in an otherwise very simple mixture model application. An analytically convenient way to deal with effective length is treat it as a bijective transformation of the transcript expression. Consider the transformation $x' = T_{\mathbb{L}}(x)$, where x'_j is defined as it is above. This is similar to the *perturbation* operation in Aitchison geometry (Aitchison, 1986b), but there it is assumed both vectors are on the simplex, whereas in the effective length transformation, the elements of $\mathbb{L}$ need only by positive.

Theorem A.0.1. *The effective length transformation $T_{\mathbb{L}} : \Delta^{n-1} \mapsto \Delta^{n-1}$ is a bijection.*

Proof. For any $u, v \in \Delta^{n-1}$ and $\mathbb{L} \in \mathbb{R}^n_+$, suppose that $T(u) = T(v)$. Then for all j

$$\frac{\mathbb{L}_j u_j}{\mathbb{L}u^\top} = \frac{\mathbb{L}_j v_j}{\mathbb{L}v^\top} \tag{A.1}$$

$$\frac{u_j}{\mathbb{L}u^\top} = \frac{v_j}{\mathbb{L}v^\top} \tag{A.2}$$

Thus

$$\begin{aligned}
\sum_{j=1}^{n} \frac{u_j}{\mathbb{L}u^\top} &= \sum_{j=1}^{n} \frac{v_j}{\mathbb{L}v^\top} \\
\frac{1}{\mathbb{L}u^\top} \sum_{j=1}^{n} u_j &= \frac{1}{\mathbb{L}v^\top} \sum_{j=1}^{n} v_j \\
\frac{1}{\mathbb{L}u^\top} &= \frac{1}{\mathbb{L}v^\top} \\
\mathbb{L}u^\top &= \mathbb{L}v^\top
\end{aligned}$$

The penultimate step follows from u, v lying on a simplex and thus summing to 1. From this we see that denominators in Equation A.2 cancel and we have $u_j = v_j$ for all j, or simply $T(u) = T(v) \implies u = v$, proving that $T_{\mathbb{L}}$ is *injective*.

Next consider any $v \in \Delta^{n-1}$. Define a vector $u = T_{1/\mathbb{L}}(v)$, that is for each element j

$$u_j = \frac{v_j/\mathbb{L}_j}{\sum_{k=1}^{n} v_k/\mathbb{L}_k}$$

Note that $u \in \Delta^{n-1}$ and, then consider $T(u)$. For each element j

$$\begin{aligned} T(u)_j &= \frac{\mathbb{L}_j u_j}{\sum_{k=1}^{n} \mathbb{L}_k u_k} \\ &= \frac{\mathbb{L}_j \frac{v_j/\mathbb{L}_j}{\sum_{k=1}^{n} v_k/\mathbb{L}_k}}{\sum_{k=1}^{n} \mathbb{L}_k \frac{v_k/\mathbb{L}_k}{\sum_{k'=1}^{n} v_{k'}/\mathbb{L}_{k'}}} \\ &= \frac{v_j}{\sum_{k=1}^{n} v_k} \\ &= v_j \end{aligned}$$

Thus $T(u) = v$, so for any $v \in \Delta^{n-1}$ there exists a $u \in \Delta^{n-1}$ where $T(u) = v$, or $T_{\mathbb{L}}$ is *surjective*.

$T_{\mathbb{L}}$ is an injection and surjection, and thus a bijection. And from the proof of surjectivity we see that $T_{1/\mathbb{L}}$ is the inverse of $T_{\mathbb{L}}$. □

This proves useful when adapting a model of transcript expression $P(x|\theta)P(\theta)$ to compute the probability of expression weighted by effective length. Specifically,

$$P(T_{\mathbb{L}}(x)|\theta)|\det(J_{T_{\mathbb{L}}})| = P(x|\theta)$$

where $J_{T_{\mathbb{L}}}$ is the Jacobian matrix corresponding to $T_{\mathbb{L}(x)}$.

Theorem A.0.2. *The determinant of this Jacobian matrix for the effective transformation $T_{\mathbb{L}}$ is*

$$\det(J_{T_{\mathbb{L}}}) = \prod_{j=1}^{n} \frac{\mathbb{L}_j}{\sum_{k=1}^{n} \mathbb{L}_k x_k}$$

Proof. Making use of the fact that $x_n = 1 - \sum_{k=1}^{n-1} x_k$, diagonal entries of the Jacobian are

$$\left(J_{T_{\mathbb{L}}}\right)_{ii} = \frac{\partial T_{\mathbb{L}}(x)_i}{\partial x_i} = \frac{\mathbb{L}_i \left(\sum_{k=1}^{n} \mathbb{L}_k x_k\right) - \mathbb{L}_i(\mathbb{L}_i - \mathbb{L}_n)x_i}{\left(\sum_{k=1}^{n} \mathbb{L}_k x_k\right)^2}$$

And off-diagonal entries are

$$\left(J_{T_{\mathbb{L}}}\right)_{ij} = \frac{\partial T_{\mathbb{L}}(x)_i}{\partial x_j} = \frac{-\mathbb{L}_i(\mathbb{L}_j - \mathbb{L}_n)x_i}{\left(\sum_{k=1}^{n} \mathbb{L}_k x_k\right)^2}$$

If we define three length n vectors u, v, w where

$$u_j = \frac{-\mathbb{L}_j x_j}{\sum_{k=1}^{n} \mathbb{L}_k x_k}$$
$$v_j = \frac{\mathbb{L}_j - \mathbb{L}_n}{\sum_{k=1}^{n} \mathbb{L}_k x_k}$$
$$w_j = \frac{\mathbb{L}_j}{\sum_{k=1}^{n} \mathbb{L}_k x_k}$$

then the Jacobian matrix can be factored as

$$J_{T_{\mathbb{L}}} = Iw + uv^{\top}$$

Then, following from the matrix determinant lemma, we have

$$\begin{aligned}
\det(J_{T_{\mathbb{L}}}) &= \det(Iw + uv^{\top}) \\
&= (1 + v^{\top}(Iw)^{-1}u)\det(Iw) \\
&= \left(1 - \sum_{j=1}^{n-1} \frac{(\mathbb{L}_j - \mathbb{L}_n)x_j}{\sum_{k=1}^{n} \mathbb{L}_k x_k}\right) \prod_{j=1}^{n-1} \frac{\mathbb{L}_j}{\sum_{k=1}^{n} \mathbb{L}_k x_k} \\
&= \left(1 - \frac{\sum_{j=1}^{n-1} \mathbb{L}_j x_j}{\sum_{k=1}^{n} \mathbb{L}_k x_k} + \frac{\mathbb{L}_n \sum_{j=1}^{n-1} x_j}{\sum_{k=1}^{n} \mathbb{L}_k x_k}\right) \prod_{j=1}^{n-1} \frac{\mathbb{L}_j}{\sum_{k=1}^{n} \mathbb{L}_k x_k} \\
&= \left(\frac{\mathbb{L}_n x_n}{\sum_{k=1}^{n} \mathbb{L}_k x_k} + \frac{\mathbb{L}_n(1 - x_n)}{\sum_{k=1}^{n} \mathbb{L}_k x_k}\right) \prod_{j=1}^{n-1} \frac{\mathbb{L}_j}{\sum_{k=1}^{n} \mathbb{L}_k x_k} \\
&= \frac{\mathbb{L}_n}{\sum_{k=1}^{n} \mathbb{L}_k x_k} \prod_{j=1}^{n-1} \frac{\mathbb{L}_j}{\sum_{k=1}^{n} \mathbb{L}_k x_k} \\
&= \prod_{j=1}^{n} \frac{\mathbb{L}_j}{\sum_{k=1}^{n} \mathbb{L}_k x_k}
\end{aligned}$$

□

Bibliography

[1] Martín Abadi et al. “TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems.” In: *arXiv preprint arXiv:1603.04467* (Mar. 2016).

[2] John Aitchison. *The Statistical Analysis of Compositional Data*. Chapman and Hall London, 1986.

[3] John Aitchison. “The simplex as a sample space.” In: *The Statistical Analysis of Compositional Data*. Chapman and Hall London, 1986, pp. 24–47.

[4] John Aitchison. “On criteria for measures of compositional difference.” In: *Math. Geol.* 24.4 (May 1992), pp. 365–379.

[5] Simon Anders and Wolfgang Huber. “Differential expression analysis for sequence count data.” en. In: *Genome Biol.* 11.10 (Oct. 2010), R106.

[6] Simon Anders, Alejandro Reyes, and Wolfgang Huber. “Detecting differential usage of exons from RNA-seq data.” en. In: *Genome Res.* 22.10 (Oct. 2012), pp. 2008–2017.

[7] J Atchison and S M Shen. “Logistic-normal distributions:Some properties and uses.” In: *Biometrika* 67.2 (Jan. 1980), pp. 261–272.

[8] A Azzalini. “A Class of Distributions Which Includes the Normal Ones.” In: *Scand. Stat. Theory Appl.* 12.2 (1985), pp. 171–178.

[9] Elsa Bernard, Laurent Jacob, Julien Mairal, and Jean-Philippe Vert. “Efficient RNA isoform identification and quantification from RNA-Seq data with network flows.” en. In: *Bioinformatics* 30.17 (Sept. 2014), pp. 2447–2455.

[10] Michael Betancourt. “Cruising the simplex: Hamiltonian Monte Carlo and the Dirichlet distribution.” In: *AIP Conf. Proc.* 1443.1 (May 2012), pp. 157–164.

[11] Jeff Bezanson, Stefan Karpinski, Viral B Shah, and Alan Edelman. “Julia: A Fast Dynamic Language for Technical Computing.” In: (Sept. 2012). arXiv: 1209.5145 [cs.PL].

[12] Anindya Bhadra, Jyotishka Datta, Nicholas G Polson, and Brandon Willard. “The Horseshoe+ Estimator of Ultra-Sparse Signals.” en. In: *Bayesian Anal.* 12.4 (Dec. 2017), pp. 1105–1131.

[13] Dean Billheimer, Peter Guttorp, and William F Fagan. “Statistical Interpretation of Species Composition.” In: *J. Am. Stat. Assoc.* 96.456 (Dec. 2001), pp. 1205–1214.

[14] Christopher M Bishop. “Approximate Inference.” In: *Pattern Recognition and Machine Learning*. springer, 2006.

[15] Christopher M Bishop. “Combining Models.” In: *Pattern Recognition and Machine Learning*. springer, 2006.

[16] P M Bracci, S B Bull, and M D Grynpas. “Analysis of compositional bone density data using log ratio transformations.” en. In: *Biometrics* 54.1 (Mar. 1998), pp. 337–349.

[17] Nicolas L Bray, Harold Pimentel, Páll Melsted, and Lior Pachter. “Near-optimal probabilistic RNA-seq quantification.” en. In: *Nat. Biotechnol.* 34.5 (May 2016), pp. 525–527.

[18] Andrei Z Broder. “On the resemblance and containment of documents.” In: *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171)*. 1997, pp. 21–29.

[19] James H Bullard, Elizabeth Purdom, Kasper D Hansen, and Sandrine Dudoit. “Evaluation of statistical methods for normalization and differential expression in mRNA-Seq experiments.” en. In: *BMC Bioinformatics* 11 (Feb. 2010), p. 94.

[20] Bob Carpenter, Andrew Gelman, Matt Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Michael A Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. “Stan: A probabilistic programming language.” In: *J. Stat. Softw.* 20 (2016), pp. 1–37.

[21] Giles Celeux and Jean Diebolt. “The SEM algorithm : a probabilistic teacher algorithm derived from the EM algorithm for the mixture problem.” In: *Computational Statistics Quarterly* 2 (1985), pp. 73–82.

[22] T Cormen, C Leiserson, R Rivest, and C Stein. “Augmented Data Structures.” In: *Introduction to Algorithms*. MIT Press (2009), 2009.

[23] C Cox. “Multinomial regression models based on continuation ratios.” en. In: *Stat. Med.* 7.3 (Mar. 1988), pp. 435–441.

[24] Adji B Dieng, Dustin Tran, Rajesh Ranganath, John Paisley, and David M Blei. “The χ-Divergence for Approximate Inference.” In: *arXiv preprint arXiv:1611. 00328* (2016).

[25] Alexander Dobin, Carrie A Davis, Felix Schlesinger, Jorg Drenkow, Chris Zaleski, Sonali Jha, Philippe Batut, Mark Chaisson, and Thomas R Gingeras. “STAR: ultrafast universal RNA-seq aligner.” en. In: *Bioinformatics* 29.1 (Jan. 2013), pp. 15–21.

[26] Juliane C Dohm, Claudio Lottaz, Tatiana Borodina, and Heinz Himmelbauer. “Substantial biases in ultra-short read data sets from high-throughput DNA sequencing.” en. In: *Nucleic Acids Res.* 36.16 (Sept. 2008), e105.

[27] C M Dutton, C Paynton, and S S Sommer. “General method for amplifying regions of very high G+C content.” en. In: *Nucleic Acids Res.* 21.12 (June 1993), pp. 2953–2954.

[28] J J Egozcue and V Pawlowsky-Glahn. "Groups of Parts and Their Balances in Compositional Data Analysis." en. In: *Math. Geol.* 37.7 (Oct. 2005), pp. 795–828.

[29] J J Egozcue, V Pawlowsky-Glahn, G Mateu-Figueras, and C Barceló-Vidal. "Isometric Logratio Transformations for Compositional Data Analysis." en. In: *Math. Geol.* 35.3 (Apr. 2003), pp. 279–300.

[30] Ensembl. *Gene annotation in Ensembl*. `http://ensembl.org/info/genome/genebuild/genome_annotation.html`. Accessed: 2018-5-18. Apr. 2018.

[31] Thomas S Ferguson. *A Bayesian Analysis of Some Nonparametric Problems*. 1973.

[32] Thomas S Ferguson. "Prior Distributions on Spaces of Probability Measures." In: *Ann. Stat.* 2.4 (1974), pp. 615–629.

[33] Stephen E Fienberg. *The Analysis of Cross-Classified Categorical Data*. en. Springer Science & Business Media, Aug. 2007.

[34] Mikhail Figurnov, Shakir Mohamed, and Andriy Mnih. "Implicit Reparameterization Gradients." In: *Advances in Neural Information Processing Systems 31*. Ed. by S Bengio, H Wallach, H Larochelle, K Grauman, N Cesa-Bianchi, and R Garnett. Curran Associates, Inc., 2018, pp. 441–452.

[35] GTEx Consortium. "The Genotype-Tissue Expression (GTEx) project." en. In: *Nat. Genet.* 45.6 (June 2013), pp. 580–585.

[36] Andrew Gelman. "The problems with p-values are not just with p-values." In: *The American Statistician* 10 (2016).

[37] Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. "Basics of Markov chain simulation." en. In: *Bayesian Data Analysis*. CRC Press, Nov. 2013.

[38] Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. "Standard probability distributions." en. In: *Bayesian Data Analysis, Third Edition*. CRC Press, Nov. 2013, pp. 575–584.

[39] Andrew Gelman and Donald B Rubin. "Inference from Iterative Simulation Using Multiple Sequences." In: *Stat. Sci.* 7.4 (1992), pp. 457–472.

[40] Peter Glaus, Antti Honkela, and Magnus Rattray. "Identifying differentially expressed transcripts from RNA-seq data with biological variation." en. In: *Bioinformatics* 28.13 (July 2012), pp. 1721–1728.

[41] J Goodman. "Classes for fast maximum entropy training." In: *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*. Vol. 1. May 2001, 561–564 vol.1.

[42] Manfred G Grabherr et al. “Full-length transcriptome assembly from RNA-Seq data without a reference genome.” en. In: *Nat. Biotechnol.* 29.7 (May 2011), pp. 644–652.

[43] S L Grokhovsky. “Specificity of DNA cleavage by ultrasound.” In: *Mol. Biol.* 40.2 (Mar. 2006), pp. 276–283.

[44] Mitchell Guttman et al. “Ab initio reconstruction of cell type–specific transcriptomes in mouse reveals the conserved multi-exonic structure of lincRNAs.” In: *Nat. Biotechnol.* 28 (May 2010), p. 503.

[45] Brian J Haas et al. “De novo transcript sequence reconstruction from RNA-seq using the Trinity platform for reference generation and analysis.” en. In: *Nat. Protoc.* 8.8 (Aug. 2013), pp. 1494–1512.

[46] Michael Hagemann-Jensen, Christoph Ziegenhain, Ping Chen, Daniel Ramsköld, Gert-Jan Hendriks, Anton J M Larsson, Omid R Faridani, and Rickard Sandberg. “Single-cell RNA counting at allele and isoform resolution using Smart-seq3.” In: *Nat. Biotechnol.* (May 2020).

[47] Paul R Halmos. “Random Alms.” In: *Ann. Math. Stat.* 15.2 (1944), pp. 182–189.

[48] Kasper D Hansen, Steven E Brenner, and Sandrine Dudoit. “Biases in Illumina transcriptome sequencing caused by random hexamer priming.” en. In: *Nucleic Acids Res.* 38.12 (July 2010), e131.

[49] Thomas J Hardcastle and Krystyna A Kelly. “baySeq: empirical Bayesian methods for identifying differential expression in sequence count data.” en. In: *BMC Bioinformatics* 11 (Aug. 2010), p. 422.

[50] Stephen W Hartley and James C Mullikin. “Detection and visualization of differential splicing in RNA-Seq data with JunctionSeq.” en. In: *Nucleic Acids Res.* 44.15 (Sept. 2016), e127.

[51] James Hensman, Panagiotis Papastamoulis, Peter Glaus, Antti Honkela, and Magnus Rattray. “Fast and accurate approximate inference of transcript expression from RNA-seq data.” In: *Bioinformatics* (Aug. 2015).

[52] David Hiller, Hui Jiang, Weihong Xu, and Wing Hung Wong. “Identifiability of isoform deconvolution from junction arrays and RNA-Seq.” en. In: *Bioinformatics* 25.23 (Dec. 2009), pp. 3056–3059.

[53] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. “Stochastic variational inference.” In: *J. Mach. Learn. Res.* 14.1 (2013), pp. 1303–1347.

[54] J R M Hosking and James R Wallis. *Regional Frequency Analysis: An Approach Based on L-Moments*. en. Cambridge University Press, Sept. 2005.

[55] Yuanhua Huang and Guido Sanguinetti. “BRIE: transcriptome-wide splicing quantification in single cells.” en. In: *Genome Biol.* 18.1 (June 2017), p. 123.

[56] Hemant Ishwaran and Lancelot F James. "Gibbs Sampling Methods for Stick-Breaking Priors." In: *J. Am. Stat. Assoc.* 96.453 (Mar. 2001), pp. 161–173.

[57] Martin Jankowiak and Fritz Obermeyer. "Pathwise Derivatives Beyond the Reparameterization Trick." In: (June 2018). arXiv: 1806.01851 [stat.ML].

[58] Hui Jiang and Wing Hung Wong. "Statistical inferences for isoform expression in RNA-Seq." en. In: *Bioinformatics* 25.8 (Apr. 2009), pp. 1026–1032.

[59] Daniel C Jones, Kavitha T Kuppusamy, Nathan J Palpant, Xinxia Peng, Charles E Murry, Hannele Ruohola-Baker, and Walter L Ruzzo. "Isolator: accurate and stable analysis of isoform-level expression in RNA-Seq experiments." en. Nov. 2016.

[60] Daniel C Jones, Walter L Ruzzo, Xinxia Peng, and Michael G Katze. "A new approach to bias correction in RNA-Seq." en. In: *Bioinformatics* 28.7 (Apr. 2012), pp. 921–928.

[61] M C Jones. "Kumaraswamy's distribution: A beta-type distribution with some tractability advantages." In: *Stat. Methodol.* 6.1 (2009), pp. 70–81.

[62] M C Jones and Arthur Pewsey. "Sinh-arcsinh distributions." In: *Biometrika* 96.4 (Dec. 2009), pp. 761–780.

[63] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. "An Introduction to Variational Methods for Graphical Models." In: *Mach. Learn.* 37.2 (Nov. 1999), pp. 183–233.

[64] Yarden Katz, Eric T Wang, Edoardo M Airoldi, and Christopher B Burge. "Analysis and design of RNA sequencing experiments for identifying isoform regulation." In: *Nat. Methods* 7 (Nov. 2010), p. 1009.

[65] Mohammad E Khan, Shakir Mohamed, Benjamin M Marlin, and Kevin P Murphy. "A stick-breaking likelihood for categorical data analysis with latent Gaussian models." In: *International conference on Artificial Intelligence and Statistics*. 2012, pp. 610–618.

[66] Daehwan Kim, Ben Langmead, and Steven L Salzberg. "HISAT: a fast spliced aligner with low memory requirements." en. In: *Nat. Methods* 12.4 (Mar. 2015), pp. 357–360.

[67] D P Kingma and J Ba. "Adam: A method for stochastic optimization." In: *arXiv preprint arXiv:1412.6980* (2014).

[68] Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. "Improved Variational Inference with Inverse Autoregressive Flow." In: *Advances in Neural Information Processing Systems 29*. Ed. by D D Lee, M Sugiyama, U V Luxburg, I Guyon, and R Garnett. Curran Associates, Inc., 2016, pp. 4743–4751.

[69] Diederik P Kingma and Max Welling. “Auto-encoding variational bayes.” In: 2014.

[70] Alp Kucukelbir, Dustin Tran, Rajesh Ranganath, Andrew Gelman, and David M Blei. “Automatic Differentiation Variational Inference.” In: *J. Mach. Learn. Res.* 18.1 (Jan. 2017), pp. 430–474.

[71] Shailesh Kumar, Angie Duy Vo, Fujun Qin, and Hui Li. “Comparative assessment of methods for the fusion transcripts detection from RNA-Seq data.” en. In: *Sci. Rep.* 6 (Feb. 2016), p. 21597.

[72] Gioele La Manno et al. “RNA velocity in single cells.” en. Oct. 2017.

[73] Serge Lang. “Bases, matrices, and linear maps.” In: *Linear Algebra*. Undergraduate Texts in Mathematics. New York, NY: Springer New York, 1987.

[74] Michael Lavine. “Some Aspects of Polya Tree Distributions for Statistical Modelling.” In: *Ann. Stat.* 20.3 (1992), pp. 1222–1235.

[75] Michael Lavine. “More Aspects of Polya Tree Distributions for Statistical Modelling.” In: *Ann. Stat.* 22.3 (1994), pp. 1161–1176.

[76] Rasko Leinonen, Hideaki Sugawara, Martin Shumway, and International Nucleotide Sequence Database Collaboration. “The sequence read archive.” en. In: *Nucleic Acids Res.* 39.Database issue (Jan. 2011), pp. D19–21.

[77] Bin Li, Tao Qing, Jinhang Zhu, Zhuo Wen, Ying Yu, Ryutaro Fukumura, Yuanting Zheng, Yoichi Gondo, and Leming Shi. “A Comprehensive Mouse Transcriptomic BodyMap across 17 Tissues by RNA-seq.” en. In: *Sci. Rep.* 7.1 (June 2017), p. 4200.

[78] Bo Li and Colin N Dewey. “RSEM: accurate transcript quantification from RNA-Seq data with or without a reference genome.” en. In: *BMC Bioinformatics* 12 (Aug. 2011), p. 323.

[79] Bo Li, Victor Ruotti, Ron M Stewart, James A Thomson, and Colin N Dewey. “RNA-Seq gene expression estimation with read mapping uncertainty.” en. In: *Bioinformatics* 26.4 (Feb. 2010), pp. 493–500.

[80] C Li and W H Wong. “Model-based analysis of oligonucleotide arrays: expression index computation and outlier detection.” en. In: *Proc. Natl. Acad. Sci. U. S. A.* 98.1 (Jan. 2001), pp. 31–36.

[81] Jun Li, Hui Jiang, and Wing Hung Wong. “Modeling non-uniformity in short-read rates in RNA-Seq data.” en. In: *Genome Biol.* 11.5 (May 2010), R50.

[82] Ping Li, Art Owen, and Cun-Hui Zhang. “One Permutation Hashing.” In: *Advances in Neural Information Processing Systems 25*. Ed. by F Pereira, C J C Burges, L Bottou, and K Q Weinberger. Curran Associates, Inc., 2012, pp. 3113–3121.

[83] Wei Vivian Li, Anqi Zhao, Shihua Zhang, and Jingyi Jessica Li. "MSIQ: Joint modeling of multiple RNA-seq samples for accurate isoform quantification." en. In: *Ann. Appl. Stat.* 12.1 (Mar. 2018), pp. 510–539.

[84] Yang I Li, David A Knowles, Jack Humphrey, Alvaro N Barbeira, Scott P Dickinson, Hae Kyung Im, and Jonathan K Pritchard. "Annotation-free quantification of RNA splicing using LeafCutter." en. In: *Nat. Genet.* 50.1 (Jan. 2018), pp. 151–158.

[85] Yingzhen Li and Richard E Turner. "Rényi Divergence Variational Inference." In: (Feb. 2016). arXiv: 1602.02311 [stat.ML].

[86] Yen-Yi Lin, Phuong Dao, Faraz Hach, Marzieh Bakhshi, Fan Mo, Anna Lapuk, Colin Collins, and S Cenk Sahinalp. "CLIIQ: Accurate Comparative Detection and Quantification of Expressed Isoforms in a Population." In: *Algorithms in Bioinformatics*. Springer Berlin Heidelberg, 2012, pp. 178–189.

[87] Romain Lopez, Jeffrey Regier, Michael B Cole, Michael I Jordan, and Nir Yosef. "Deep generative modeling for single-cell transcriptomics." en. In: *Nat. Methods* 15.12 (Dec. 2018), pp. 1053–1058.

[88] Romain Lopez, Jeffrey Regier, Michael B Cole, Michael Jordan, and Nir Yosef. "Bayesian Inference for a Generative Model of Transcriptome Profiles from Single-cell RNA Sequencing." en. Mar. 2018.

[89] Michael I Love, John B Hogenesch, and Rafael A Irizarry. "Modeling of RNA-seq fragment sequence bias reduces systematic errors in transcript abundance estimation." en. In: *Nat. Biotechnol.* 34.12 (Dec. 2016), pp. 1287–1291.

[90] David Lovell, Vera Pawlowsky-Glahn, Juan José Egozcue, Samuel Marguerat, and Jürg Bähler. "Proportionality: a valid alternative to correlation for relative data." en. In: *PLoS Comput. Biol.* 11.3 (Mar. 2015), e1004075.

[91] David Lunn, David Spiegelhalter, Andrew Thomas, and Nicky Best. "The BUGS project: Evolution, critique and future directions." en. In: *Stat. Med.* 28.25 (Nov. 2009), pp. 3049–3067.

[92] Igor Mandric, Yvette Temate-Tiagueu, Tatiana Shcheglova, Sahar Al Seesi, Alex Zelikovsky, and Ion I Mandoiu. "Fast bootstrapping-based estimation of confidence intervals of expression levels and differential expression from RNA-Seq data." en. In: *Bioinformatics* 33.20 (Oct. 2017), pp. 3302–3304.

[93] Florian Markowetz and Rainer Spang. "Inferring cellular networks-a review." en. In: *BMC Bioinformatics* 8 Suppl 6 (Sept. 2007), S5.

[94] R Daniel Mauldin, William D Sudderth, and Stanley C Williams. "Polya Trees and Random Distributions." en. In: *Ann. Stat.* 20.3 (Sept. 1992), pp. 1203–1221.

[95] Warren A McGee, Harold Pimentel, Lior Pachter, and Jane Y Wu. "Compositional Data Analysis is necessary for simulating and analyzing RNA-Seq data." en. In: *bioRxiv* (Mar. 2019), p. 564955.

[96] Stephen B Montgomery, Micha Sammeth, Maria Gutierrez-Arcelus, Radoslaw P Lach, Catherine Ingle, James Nisbett, Roderic Guigo, and Emmanouil T Dermitzakis. "Transcriptome genetics using second generation sequencing in a Caucasian population." en. In: *Nature* 464.7289 (Apr. 2010), pp. 773–777.

[97] Ali Mortazavi, Brian A Williams, Kenneth McCue, Lorian Schaeffer, and Barbara Wold. "Mapping and quantifying mammalian transcriptomes by RNA-Seq." en. In: *Nat. Methods* 5.7 (July 2008), pp. 621–628.

[98] Lior Pachter. "Models for transcript quantification from RNA-Seq." In: (Apr. 2011). arXiv: 1104.3889 [q-bio.GN].

[99] Susan M Paddock, Fabrizio Ruggeri, Michael Lavine, and Mike West. "Randomized Polya tree models for nonparametric Bayesian inference." In: *Statistica Sinica* 13.2 (2003), pp. 443–460.

[100] Panagiotis Papastamoulis, James Hensman, Peter Glaus, and Magnus Rattray. "Improved variational Bayes inference for transcript expression estimation." en. In: *Stat. Appl. Genet. Mol. Biol.* 13.2 (Apr. 2014), pp. 203–216.

[101] Rob Patro, Geet Duggal, Michael I Love, Rafael A Irizarry, and Carl Kingsford. "Salmon provides fast and bias-aware quantification of transcript expression." en. In: *Nat. Methods* 14.4 (Apr. 2017), pp. 417–419.

[102] Rob Patro, Stephen M Mount, and Carl Kingsford. "Sailfish enables alignment-free isoform quantification from RNA-seq reads using lightweight algorithms." en. In: *Nat. Biotechnol.* 32.5 (May 2014), pp. 462–464.

[103] V Pawlowsky-Glahn and J J Egozcue. "Geometric approach to statistical analysis on the simplex." en. In: *Stoch. Environ. Res. Risk Assess.* 15.5 (Oct. 2001), pp. 384–398.

[104] Vera Pawlowsky-Glahn and Antonella Buccianti. *Compositional Data Analysis: Theory and Applications*. en. John Wiley & Sons, Aug. 2011.

[105] Vera Pawlowsky-Glahn, Juan José Egozcue, and Raimon Tolosana-Delgado. *Modeling and Analysis of Compositional Data*. en. John Wiley & Sons, Feb. 2015.

[106] Karl Pearson. "Contributions to the Mathematical Theory of Evolution." In: *Philos. Trans. R. Soc. Lond. A* 185 (1894), pp. 71–110.

[107] Zhiyu Peng et al. "Comprehensive analysis of RNA-Seq data reveals extensive RNA editing in a human transcriptome." en. In: *Nat. Biotechnol.* 30.3 (Feb. 2012), pp. 253–260.

[108] Mihaela Pertea, Geo M Pertea, Corina M Antonescu, Tsung-Cheng Chang, Joshua T Mendell, and Steven L Salzberg. "StringTie enables improved reconstruction of a transcriptome from RNA-seq reads." en. In: *Nat. Biotechnol.* 33.3 (Mar. 2015), pp. 290–295.

[109] Harold Pimentel, Nicolas L Bray, Suzette Puente, Páll Melsted, and Lior Pachter. "Differential analysis of RNA-seq incorporating quantification uncertainty." en. In: *Nat. Methods* 14.7 (July 2017), pp. 687–690.

[110] Jim Pitman. "Poisson-Dirichlet and GEM Invariant Distributions for Split-and-Merge Transformations of an Interval Partition." In: *Comb. Probab. Comput.* 11.5 (Sept. 2002), pp. 501–514.

[111] Martyn Plummer and Others. "JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling." In: *Proceedings of the 3rd international workshop on distributed statistical computing*. Vol. 124. 2003.

[112] Rajesh Ranganath, Sean Gerrish, and David Blei. "Black Box Variational Inference." en. In: PMLR, Apr. 2014, pp. 814–822.

[113] Danilo Jimenez Rezende and Shakir Mohamed. "Variational Inference with Normalizing Flows." In: (May 2015). arXiv: 1505.05770 [stat.ML].

[114] Adam Roberts, Cole Trapnell, Julie Donaghey, John L Rinn, and Lior Pachter. "Improving RNA-Seq expression estimates by correcting for fragment bias." en. In: *Genome Biol.* 12.3 (Mar. 2011), R22.

[115] Gordon Robertson et al. "De novo assembly and analysis of RNA-seq data." en. In: *Nat. Methods* 7.11 (Nov. 2010), pp. 909–912.

[116] Mark D Robinson, Davis J McCarthy, and Gordon K Smyth. "edgeR: a Bioconductor package for differential expression analysis of digital gene expression data." en. In: *Bioinformatics* 26.1 (Jan. 2010), pp. 139–140.

[117] SEQC/MAQC-III Consortium. "A comprehensive assessment of RNA-seq accuracy, reproducibility and information content by the Sequencing Quality Control Consortium." en. In: *Nat. Biotechnol.* 32.9 (Sept. 2014), pp. 903–914.

[118] Ashis Saha, Yungil Kim, Ariel D H Gewirtz, Brian Jo, Chuan Gao, Ian C McDowell, GTEx Consortium, Barbara E Engelhardt, and Alexis Battle. "Co-expression networks reveal the tissue-specific regulation of transcription and splicing." en. In: *Genome Res.* 27.11 (Nov. 2017), pp. 1843–1858.

[119] Tim Salimans and David A Knowles. "Fixed-Form Variational Posterior Approximation through Stochastic Linear Regression." en. In: *Bayesian Anal.* 8.4 (Dec. 2013), pp. 837–882.

[120] John Salvatier, Thomas V Wiecki, and Christopher Fonnesbeck. "Probabilistic programming in Python using PyMC3." en. In: *PeerJ Comput. Sci.* 2 (Apr. 2016), e55.

[121] J L Scealy and A H Welsh. "Colours and Cocktails: Compositional Data Analysis 2013 Lancaster Lecture." In: *Aust. N. Z. J. Stat.* 56.2 (June 2014), pp. 145–169.

[122] M Schena, D Shalon, R W Davis, and P O Brown. "Quantitative monitoring of gene expression patterns with a complementary DNA microarray." en. In: *Science* 270.5235 (Oct. 1995), pp. 467–470.

[123] Jayaram Sethuraman. "A Constructive Definition of Dirichlet Priors." In: *Stat. Sin.* 4.2 (1994), pp. 639–650.

[124] Nicholas Shackel. "Bertrand's Paradox and the Principle of Indifference." In: *Philos. Sci.* 74.2 (2007), pp. 150–175.

[125] Justin D Silverman, Alex D Washburne, Sayan Mukherjee, and Lawrence A David. "A phylogenetic transform enhances analysis of compositional microbiota data." en. In: *Elife* 6 (Feb. 2017).

[126] Gordon K Smyth. "Linear models and empirical bayes methods for assessing differential expression in microarray experiments." en. In: *Stat. Appl. Genet. Mol. Biol.* 3.1 (Feb. 2004).

[127] Charlotte Soneson, Michael I Love, and Mark D Robinson. "Differential analyses for RNA-seq: transcript-level estimates improve gene-level inferences." en. In: *F1000Res.* 4 (Dec. 2015), p. 1521.

[128] Avi Srivastava, Hirak Sarkar, Nitish Gupta, and Rob Patro. "RapMap: a rapid, sensitive and accurate tool for mapping RNA-seq reads to transcriptomes." en. In: *Bioinformatics* 32.12 (June 2016), pp. i192–i200.

[129] Connie Stewart and Christopher Field. "Managing the Essential Zeros in Quantitative Fatty Acid Signature Analysis." en. In: *JABES* 16.1 (Mar. 2011), pp. 45–69.

[130] Yee Whye Teh. "Dirichlet Process." In: *Encyclopedia of Machine Learning*. Ed. by Sammut, Claude Webb, Geoffrey. Boston, MA: Springer US, 2010, pp. 280–287.

[131] Jakub M Tomczak and Max Welling. "Improving Variational Auto-Encoders using Householder Flow." In: (Nov. 2016). arXiv: 1611.09630 [cs.LG].

[132] Alexandru I Tomescu, Anna Kuosmanen, Romeo Rizzi, and Veli Mäkinen. "A novel min-cost flow method for estimating transcript expression with RNA-Seq." en. In: *BMC Bioinformatics* 14 Suppl 5 (Apr. 2013), S15.

[133] Dustin Tran, Alp Kucukelbir, Adji B Dieng, Maja Rudolph, Dawen Liang, and David M Blei. "Edward: A library for probabilistic modeling, inference, and criticism." In: (Oct. 2016). arXiv: 1610.09787 [stat.CO].

[134] Cole Trapnell, Adam Roberts, Loyal Goff, Geo Pertea, Daehwan Kim, David R Kelley, Harold Pimentel, Steven L Salzberg, John L Rinn, and Lior Pachter. "Differential gene and transcript expression analysis of RNA-seq experiments with TopHat and Cufflinks." en. In: *Nat. Protoc.* 7.3 (Mar. 2012), pp. 562–578.

[135] Cole Trapnell, Brian A Williams, Geo Pertea, Ali Mortazavi, Gordon Kwan, Marijke J van Baren, Steven L Salzberg, Barbara J Wold, and Lior Pachter. "Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation." en. In: *Nat. Biotechnol.* 28.5 (May 2010), pp. 511–515.

[136] Ernest Turro, William J Astle, and Simon Tavaré. "Flexible analysis of RNA-seq data using mixed effects models." en. In: *Bioinformatics* 30.2 (Jan. 2014), pp. 180–188.

[137] Ernest Turro, Shu-Yi Su, Ângela Gonçalves, Lachlan J M Coin, Sylvia Richardson, and Alex Lewin. "Haplotype and isoform specific expression estimation using multi-mapping RNA-seq reads." en. In: *Genome Biol.* 12.2 (Feb. 2011), R13.

[138] Zhong Wang, Mark Gerstein, and Michael Snyder. "RNA-Seq: a revolutionary tool for transcriptomics." en. In: *Nat. Rev. Genet.* 10.1 (Jan. 2009), pp. 57–63.

[139] D F Watson and G M Philip. "Measures of variability for geological data." In: *Math. Geol.* 21.2 (Feb. 1989), pp. 233–254.

[140] Anna Wetterbom, Adam Ameur, Lars Feuk, Ulf Gyllensten, and Lucia Cavelier. "Identification of novel exons and transcribed regions by chimpanzee transcriptome sequencing." en. In: *Genome Biol.* 11.7 (July 2010), R78.

[141] Mohsen Zakeri, Avi Srivastava, Fatemeh Almodaresi, and Rob Patro. "Improved data-driven likelihood factorizations for transcript abundance estimation." en. In: *Bioinformatics* 33.14 (July 2017), pp. i142–i151.

[142] Quan Zhang and Mingyuan Zhou. "Permuted and Augmented Stick-Breaking Bayesian Multinomial Regression." In: *J. Mach. Learn. Res.* 18.1 (2017), pp. 7479–7511.

[143] Yao-Zhong Zhang, Rui Yamaguchi, Seiya Imoto, and Satoru Miyano. "Sequence-specific bias correction for RNA-seq data using recurrent neural networks." en. In: *BMC Genomics* 18.Suppl 1 (Jan. 2017), p. 1044.

[144] Anqi Zhu, Avi Srivastava, Joseph G Ibrahim, Rob Patro, and Michael I Love. "Nonparametric expression analysis using inferential replicate counts." en. In: *Nucleic Acids Res.* 47.18 (Oct. 2019), e105.