Ambiguous fragment assignment for high-throughput sequencing experiments

Adam Roberts

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2013-177
http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-177.html

October 30, 2013

Acknowledgement

I would like to thank my advisor, Lior Pachter, and my thesis and qual committees: Yun S. Song, Michael Eisen, and Dan Klein. Most of my work was funded by a National Science Foundation Graduate Research Fellowship, and I was also supported by NIH grant R01 HG006129.

Ambiguous fragment assignment for high-throughput sequencing experiments

by

Adam Roberts

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Engineering – Computer Science

and the Designated Emphasis

in

Computational and Genomic Biology

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Lior Pachter, Chair
Professor Michael Eisen
Professor Yun S. Song

Fall 2013

Ambiguous fragment assignment for high-throughput sequencing experiments

Abstract

Ambiguous fragment assignment for high-throughput sequencing experiments

by

Adam Roberts

Doctor of Philosophy in Engineering – Computer Science

University of California, Berkeley

Professor Lior Pachter, Chair

As the cost of short-read, high-throughput DNA sequencing continues to fall rapidly, new uses for the technology have been developed aside from its original purpose in determining the genome of various species. Many of these new experiments use the sequencer as a digital counter for measuring biological activities [70] such as gene expression (RNA-Seq [44]) or protein binding (ChIP-Seq [56]).

A common problem faced in the analysis of these data is that of sequenced fragments that are "ambiguous", meaning they resemble multiple loci in a reference genome or other sequence. In early analyses, such ambiguous fragments were ignored or were assigned to loci using simple heuristics. However, statistical approaches using maximum likelihood estimation have been shown to greatly improve the accuracy of downstream analyses and have become widely adopted [71, 64, 34, 67, 53]. Optimization based on the expectation-maximization (EM) algorithm are often employed by these methods to find the optimal sets of alignments, with frequent enhancements to the model. Nevertheless, these improvements increase complexity, which, along with an exponential growth in the size of sequencing datasets, has led to new computational challenges.

Herein, we present our model for ambiguous fragment assignment for RNA-Seq, which includes the most comprehensive set of parameters of any model introduced to date, as well as various methods we have explored for scaling our optimization procedure. These methods include the use of an online EM algorithm and a distributed EM solution implemented on the Spark cluster computing system. Our advances have resulted in the first efficient solution to the problem of fragment assignment in sequencing.

Furthermore, we are the first to create a fully generalized model for ambiguous fragment assignment and present details on how our method can provide solutions for additional high-throughput sequencing assays including ChIP-Seq, Allele-Specific Expression (ASE), and the detection of RNA-DNA Differences (RDDs) in RNA-Seq.

To Ashley and Sagan.

Contents

Contents **ii**

1 Introduction **1**
1.1 Fragment Assignment . . . 1
1.2 RNA-Seq . . . 3
1.3 Prior work . . . 3

2 The `eXpress` Model **6**
2.1 Problem statement . . . 6
2.2 Parameters . . . 6
2.3 Derived values . . . 7
2.4 Generative model . . . 8
2.5 Likelihood function . . . 10
2.5.1 Approximation . . . 10
2.5.2 Positional uniformity . . . 11

3 Parameter Estimation with the EM Algorithm **12**
3.1 The EM algorithm . . . 12
3.2 Basic EM algorithm for fragment assignment . . . 12
3.3 Auxiliary parameter estimation . . . 14
3.4 EM formulation . . . 15
3.5 Batch estimation algorithm . . . 16

4 Fragmentation Bias **18**
4.1 Background . . . 18
4.2 Method . . . 20
4.2.1 Sequence probabilities . . . 20
4.2.2 Positional bias weights and probabilities . . . 22
4.2.3 Effective length . . . 23
4.2.4 Updated likelihood . . . 23
4.2.5 Estimation procedure . . . 24
4.3 Validation . . . 24

4.3.1 Comparison to alternative expression arrays . . . 25
4.3.2 Comparison with previous methods . . . 26
4.3.3 Technical replicates . . . 27
4.3.4 Library preparation methods . . . 28
4.3.5 Sequencing platforms . . . 28

5 Scaling the Optimization 31
5.1 Comparisons . . . 32
5.2 The naïve batch approach of `RSEM` . . . 33
5.3 Partitioning the data . . . 36
5.4 The heuristic partitioning approach of `Cufflinks` . . . 39
5.4.1 Model . . . 39
5.4.2 Optimization . . . 39
5.5 The online EM approach of `eXpress` . . . 41
5.5.1 Overview . . . 41
5.5.2 Mathematical details . . . 43
5.5.3 Allele-specific expression . . . 47
5.5.4 Performance comparison . . . 47
5.5.5 Discussion . . . 48
5.6 The distributed batch EM approach of `eXpress-D` . . . 48
5.6.1 Background . . . 48
5.6.2 Method . . . 50
5.6.3 Results . . . 54
5.6.4 Discussion . . . 56

6 Updating Estimates After Changes to Target Set 57
6.1 Introduction . . . 57
6.2 Methods . . . 60
6.2.1 Incremental adjustment of abundance estimates . . . 60
6.2.2 Mathematical details . . . 60
6.2.3 Improving performance by approximating the affected set . . . 62
6.3 Results . . . 63
6.3.1 Accuracy of partitioning approximation . . . 63
6.3.2 Application to RefSeq incremental update . . . 64
6.4 Discussion . . . 64

7 Detecting RNA-DNA Differences 67
7.1 Background . . . 67
7.2 Previous Methods . . . 68
7.2.1 Li *et al.* [35] . . . 68
7.2.2 Peng *et al.* [51] . . . 69
7.3 Method . . . 70

7.3.1 Generative model 70
7.3.2 Likelihood function 72
7.3.3 Optimization of likelihood 73
7.3.4 Method 74
7.4 Results 76
7.5 Discussion 77

8 Conclusion 78

Bibliography 79

Acknowledgments

I would like to thank Lior Pachter for being a truly amazing advisor and collaborator on this work. It was really a great experience working with you, and I value all that I learned from you. Thanks to my thesis and qual committees: Yun S. Song for teaching me mathematical rigor, Michael Eisen for challenging the weaknesses in my understanding in biology, and Dan Klein for providing a new perspective on the problems herein. Leonard McMillan and Wei Wang are responsible for introducing me to the field of bioinformatics and pushing me to get involved in research, and I am grateful for their support throughout my undergraduate career. Thanks to Cole Trapnell for laying the groundwork for a lot of my research and for helping me to become a better engineer and to the other members of the Pachter Lab with whom I've collaborated: Harold Pimentel, Lorian Schaeffer, Meromit Singer, Brielin Brown, Nicholas Bray, Shannon Hateley, and Sharon Aviran. Harvey Feng from the AmpLab was instrumental in the implementation of eXpress-D. My original officemates in the Song lab–Andrew Chan, Ma'ayan Bresler, Joshua Paul, and Paul Jenkins–were incredibly helpful in getting me over the graduate student learning curve and helping me survive my first year. Thanks to the Brian McClendon for managing the Computational Biology DE group. Also, thanks to Trey Anastasio, Jon Fishman, Mike Gordon, and Page McConnell for providing the soundtrack to my research. Most of my work was funded by a National Science Foundation Graduate Research Fellowship, and I was also supported by NIH grant R01 HG006129.

Finally, I would like to thank my incredible family for supporting me throughout this time. My parents for always encouraging me to pursue my interests, my brothers for joining me at concerts, my dog Sagan for forcing me to take a break when he needed a walk, and my amazing wife Ashley for teaching me to practice yoga, being my editor, and picking up the slack at home when I was too busy.

Chapter 1

Introduction

In recent years, the cost of sequencing DNA molecules has dropped dramatically, leading to to an explosion of new data and important biological discoveries. While the change is often attributed to the introduction of new short-read sequencers such as the Illumina Hi-Seq, ABI SOLiD, and Ion Torrent platforms, the hardware is only a part of the story. These new machines are no doubt breakthroughs in and of themselves, producing massive amounts of data quickly and cheaply. However, the data is not of the quality of the previous generation of Sanger sequencing devices, as the read lengths are an order of magnitude shorter and the output more error-prone. What has allowed the data to be of valuable to the scientific community nonetheless is the advanced algorithms and software that have been developed alongside it.

Morever, the decrease in price and increase and output has led researchers to develop new assays that essentially turn the sequencing device into a digital counter [70]. Issues associated with the shorter read length, higher error rates, and biases inherent in library preparation are being solved *in silico* in conjunction with statistical inferences for these new assays by a new class of bioinformatics software, often based on modern machine learning techniques.

One very important problem associated with modern sequencers and assays is that of “ambiguous fragment assignment”, which is the focus of this thesis.

1.1 Fragment Assignment

Modern sequencing experiments usually involve the shearing of DNA or cDNA into relatively short fragments for processing on a high-throughput sequencing device such as the Illumina HiSeq. In the analysis of the resulting data, one of the first steps is to align the reads representing these partially-sequenced fragments to a set of target sequences. This procedure identifies locations within the target sequences that each fragment may have originated from using a threshold on mismatches and insertions or deletions (indels), thus reducing the focus of downstream analysis to only highly probable loci. Numerous read mappers exist to solve this problem with various features and performance characteristics, the most popular of which are based on the Burrows-Wheeler transform [29, 33].

A common problem in downstream analysis of the resultant alignment data is that fragments often map ambiguously to multiple target sequences. For example, in the case of RNA-Seq a given fragment might align to multiple isoforms of a gene as well as to multiple genes within a gene family. This ambiguity makes it difficult to measure the abundance of transcripts, especially those with few unique regions. A similar problem occurs with ChIP-Seq data, where fragments align to many regions of the genome, complicating the problem of peak finding for determining binding sites [11]. Another example is metagenomics, where researchers wish to detect the presence and relative abundance of various closely related species of microorganisms in a pooled sample of DNA [42].

We have developed a maximum likelihood estimation approach for determining the *most likely* abundances of a set of target sequences to explain the output of a sequencing experiment. These targets can be transcripts in RNA-Seq, binding sites in ChIP-Seq, or entire genomes in metagenomic analyses. Our model–called the *eXpress model*–is the most sophisticated to date and combines various features of alignments including fragment length, sequence composition, and profiles of potential sequencer errors in order to compute the probability of the observed sequencing reads in an experiment. The parameters modeling these features are simultaneously estimated from the data and used to probabilistically assign ambiguous fragments as part of an expectation-maximization (EM) likelihood optimization procedure.

In the interest of scaling our method to the growing size of sequencing datasets, we have explored several algorithms for optimization including the online EM as well as a distributed version of the batch EM. In the former case we achieve linear-time scaling in time with constant memory use while maintaining very high accuracy with a software tool called `eXpress`. In the latter, we have shown that it is possible to process any amount of data in constant time by scaling the size of a cluster in a cloud computing environment (such as Amazon's EC2) using `eXpress-D`.

We have also introduced a method for efficiently updating the results of our analysis after a modification of the target set with a tool called `ReXpress`.

Moreover, we have generalized the `eXpress` model to accept uncertainty in target sequences. This extension adds robustness to `eXpress` while also allowing it to detect differences between a reference sequence and that from which the fragments are composed, such as in the case of RNA-DNA Differences (RDDs) or Single Nucleotide Polymorphisms (SNPs).

Earlier versions of this work have been presented in a piecemeal fashion, over the course of several papers [55, 53, 54]. This dissertation is meant to simplify and unify those previous works, while also providing an overview of the state of the art for solving the ambiguous fragment assignment problem.

1.2 RNA-Seq

While the methods herein are applicable to many high-throughput sequencing experiments, we shall focus most of our descriptions on RNA-Seq, as isoform abundance estimation is one of the first areas where the community focused on solving the ambiguous fragment assignment problem. Applications to other technologies will be discussed more in Chapter 8.

RNA-Seq technology offers the possibility of accurately measuring transcript abundances in a sample of RNA by sequencing of double stranded cDNA [40]. Unfortunately, current technological limitations of sequencers require that the cDNA molecules represent only partial fragments of the RNA being probed. The cDNA fragments are obtained by a series of steps, often including reverse transcription primed by random hexamers (RH), or by oligo(dT). Most protocols also include a fragmentation step, typically RNA hydrolysis or nebulization, or alternatively cDNA fragmentation by DNase I treatment or sonication. Many sequencing technologies also require constrained cDNA lengths, so a final gel cutting step for size selection may be included. Figure 1.1 shows how some of these procedures are combined in a typical experiment.

1.3 Prior work

Work in the area of ambiguous fragment assignment [19] has, until recently, been focused in the area of isoform and gene reconstruction and abundance measurements, initially using expressed sequence tags (ESTs) [71] and later RNA-Seq [44, 23, 67, 32] as input ata.

[71] introduced a simple (and slightly inaccurate–see Section 3.2 and [48]) multinomial likelihood model and EM optimization procedure for assigning ambiguous ESTs to aid in reconstructing isoforms using splice graphs. Their model assumes isoforms that are known to generate many fragments are more likely to generate compatible ambiguous fragments than others. However, by not taking the isoform length into account, they miss the fact that an isoform may produce fewer fragments than another simply because it's shorter and lose the ability to measure abundance.

In the first paper on RNA-Seq, [44] attempted to solve the problem of assigning fragments that align ambiguously to the genome, also known as multi-reads. Since [44] is not interested in finding isoform-level abundances, there is no attempt to disambiguate within the isoforms of a gene. Instead, the authors introduce a method–later named the "rescue method" by [32]–for assigning multi-reads based on the length-normalized unique read counts of genes measured in units of reads per kilobase per million mapped (RPKM). This method is nearly identical to a single step of the EM algorithm of [71], with a focus on genes instead of isoforms. Aside from inaccuracies in abundance estimation introduced by using gene-level RPKMs without regard for isoform-level abundances [65], the rescue method lacks a probabilistic basis which is likely why a more complete optimization was not used.

Shortly after the introduction of RNA-Seq, [32] and [67] recognized the need for isoform-level disambiguation and rediscovered role of generative models and the EM algorithm in

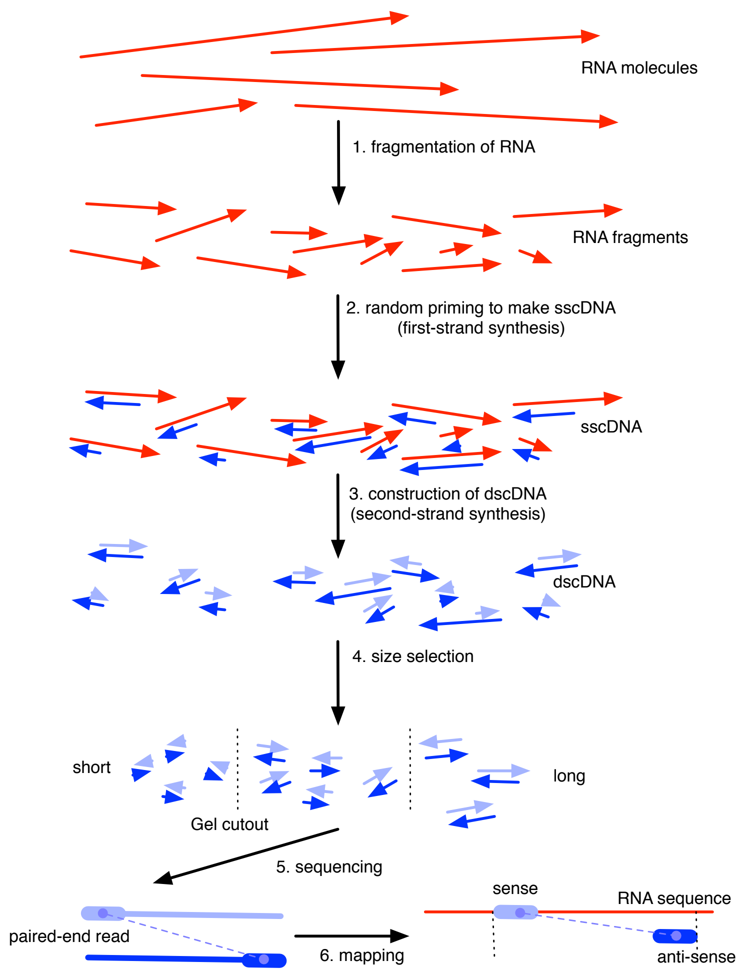


Figure 1.1: *Overview of a typical RNA-Seq experiment.* RNA is initially fragmented (1) followed by first-strand synthesis priming (2), which selects the 3' fragment end (in transcript orientation), to make single stranded cDNA. Double stranded cDNA created during second-strand synthesis (3), which selects the 5' fragment end, is then size selected (4) resulting in fragments suitable for sequencing (5). Sequenced reads are mapped to opposite strands of the genome (6), and in the case of known transcript or fragment strandedness, the read alignments reveal the $5'$ and $3'$ ends of the sequenced fragment. All arrows are directed $5'$ to $3'$ in transcript orientation.

solving this problem. Both groups introduced very similar generative models and optimization procedures, extending the work of both [71] and [44] and showing a much improved ability to disambiguate fragment assignments over the latter [32]. Along with the methods, software implementations were released called `RSEM` [32] and `Cufflinks` [67]. Other similar EM-based solutions have been developed such as `MISO` [24] and `IsoEM` [45], but since `RSEM` and `Cufflinks` have become the most widely-adopted and represent most of the breadth in the field, we will focus on these two methods in our comparisons.

Major differences between the methods include the addition of parameters for paired-end fragment lengths in `Cufflinks` and sequencing errors in `RSEM` as well as a heuristic in `Cufflinks` for partitioning the data based on genomic loci (see Chapter 5 for more details).

It is also worth noting that this problem shares many similarities with latent Dirichlet allocation (LDA) [6].

Chapter 2

The eXpress Model

This chapter will proceed as follows: In Section 2.1 we will formulate our problem as one of statistical inference and optimization, with respect to a set of parameters modeling a modern high-throughput sequencing experiment. The details of these parameters and their origins are introduced in Section 2.2 followed by an explanation of the generative model (Section 2.4) and resulting likelihood function (Section 2.5). An approximation to the likelihood and an assumption of uniformity are described in Sections 2.5.1 and Section 2.5.2, respectively. The latte assumption will be relaxed when we discuss fragmentation bias in Chapter 4 following the basic EM optimization procedure in Chapter 3.

2.1 Problem statement

Given a set of target sequences $\mathcal{T}$ and a set of fragments $\mathcal{F}$ generated from $\mathcal{T}$ in a sequencing experiment, we wish to find the set of partial assignments,

$$\mathcal{P}_f = \left\{ P_f(t,p,l) : \sum_{t',p',l'} P_f(t',p',l') = 1 \right\} \text{for all} f \in \mathcal{F}, \tag{2.1}$$

that maximize the likelihood of a generative model for the experiment, where $P_f(t,p,l)$ is the probability that fragment f originated at position p of transcript t with a fragment length of l.

2.2 Parameters

Before we defining our generative model, we describe the parameters affecting the outcome of an experiment. For our purposes, we will describe these parameters in terms of an RNA-Seq experiment as shown in Figure 1.1, but they apply in an identical manner to many other cases. We will again use f to represent a fragment and its sequence and t, p, l to represent the target, position, and inferred length of a single alignment, respectively.

- Relative abundance (ρ): The relative abundances are the proportions of the initial, intact (c)DNA molecules matching each target sequence such that $\sum_t \rho_t = 1$. In the case of RNA-Seq, the targets are transcripts and ρ_t represents the number of copies of transcript t relative to the number of copies of all transcripts in the transcriptome $\mathcal{T}$.

- Fragment length (λ): The fragment length distribution measures the probability of observing various fragment lengths created by the shearing process. As illustrated in Figure 3.2A this distribution can be often be approximated with a Gaussian due to the size selection step (4 in Figure 1.1). In the case of single-end reads, the fragment length cannot be exactly determined since one of the fragment ends cannot be aligned. For paired-end reads, both ends can be aligned to the target references to determine an inferred length for each alignment. If multiple alignments exists, the exact length is still unknown but λ_l provides the probability of observing a fragment of inferred length l independent of all other parameters. We also define M_L to be the maximum allowed length such that $\lambda_{l'} = 0$ for $l' > M_L$.

- Position (π): The position of the start and end of a fragment within a target sequence is known to be non-uniform in sequencing experiments such as RNA-Seq [18, 55]. In our method, we attribute this non-uniformity primarily to the sequence specificity of portions of the experimental protocol. Certain steps in the library preparation, including priming for reverse transcription in RNA-Seq, tend to select certain positions over others based on the nucleotides present as well as their order. Chapter 4 is devoted to the details of our model for fragmentation bias which affects the positions of fragments. For the purposes of this chapter, we will assume the position is chosen uniformly at random from within the transcriptome, which implies that $\pi_{p|t,l} = (l(t) - l + 1)^{-1}$ where $l(t)$ is the length of target t in base pairs (see Section 2.5.2).

- Fragment sequence (ϕ): The observed sequence of a fragment is not completely determined by the sequence of the molecule from which it originated. Sequencing devices will occasionally emit base substitution errors or even insert nonexistent bases or delete real ones. The distribution of these errors is dependent on the platform used for sequencing as well as the chemistry used in library preparation, among other factors. Given the target and observed sequences of the fragment, we assume this parameter to independent of all other features of the fragment.

2.3 Derived values

Using the above set of basic parameters, we derive several other values that will be useful in simplifying our generative model and likelihood function.

- Effective length ($\tilde{l}$): We define $\tilde{l}(t)$ to be the effective length of a target t which, under the uniform position assumption, can be computed as

$$\tilde{l}(t) = \sum_{l=1}^{M_L} \lambda_l (l(t) - l + 1), \tag{2.2}$$

where $l(t)$ is the actual length (in base pairs) of target t.

This quantity is important since a fragment can have endpoints at only a limited number of locations in a target dependent on the fragment's length. Averaging over the fragment length distribution allows for a better measure of the targets "length" in the view of the stochastic process that generates fragments. We will revisit this quantity and provide more intuition in Chapter 4 when we leave behind the assumption of uniformity.

- Target sampling probability (τ): The sampling probability of target t, τ_t, is the probability of selecting target t to generate a fragment independent of all other parameters. It can be derived as

  $$\tau_t = \frac{\rho_t \tilde{l}(t)}{\sum_u \rho_u \tilde{l}(u)}. \tag{2.3}$$

  Furthermore, as proven in [67], given τ and $\tilde{l}$, ρ can be recovered as

  $$\rho_t = \frac{\tau_t / \tilde{l}(t)}{\sum_u \tau_u / \tilde{l}(u)}. \tag{2.4}$$

We also derive the target sampling probability conditional on fragment length as

$$\tau_{t|l} = \frac{\rho_t \cdot (l(t) - l + 1)}{\sum_{u \in \mathcal{T}} \rho_u \cdot (l(u) - l + 1)}. \tag{2.5}$$

With the relationship between $\tilde{l}$, ρ, and τ formally stated, a more intuitive explanation of the effective length is now possible. The effective length is proportional to the target sampling probability, given equal abundance. In other words, if one were to select a fragment of any length (using λ) from a sample where each target had exactly a single copy, then $\tau_t \propto \tilde{l}(t)$.

2.4 Generative model

Figure 2.1 presents a graphical model defining the generative model used by our method. According to this model, a fragment length l is first sampled from λ. A target is then selected given l and the target sampling probability τ (which itself relies on ρ, λ, ϕ–see Section 2.3 for more details). Next, a position within t is selected given l and the position distribution π. Finally, the fragment is generated from the target after introducing sequencer errors based on ϕ.

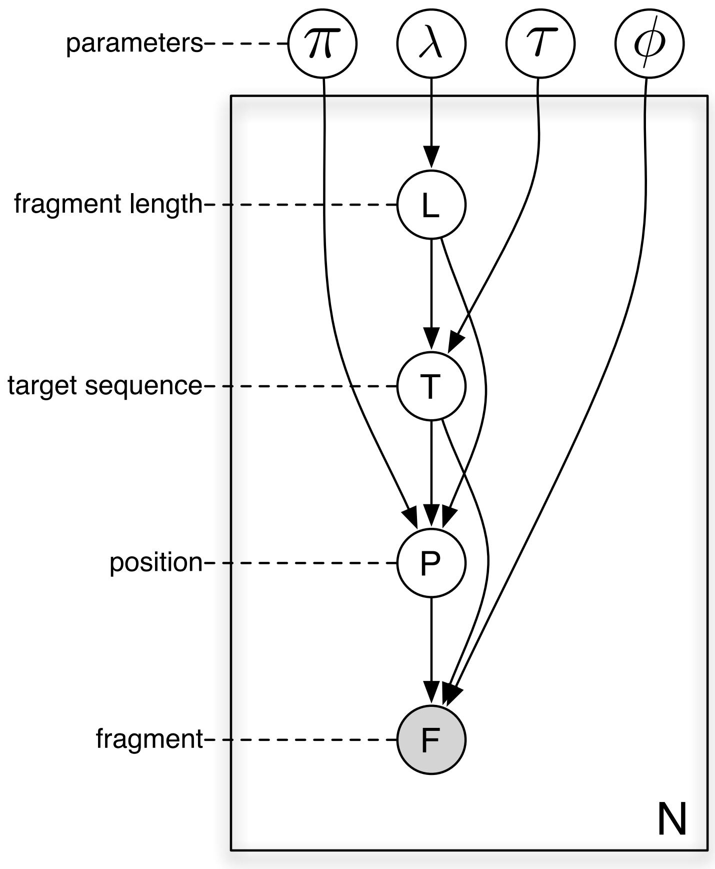


Figure 2.1: *The* ***eXpress*** *generative model.* A graphical model describing the generative process for obtaining fragments that underlies the likelihood function used in our method. The model represents the relationship between the positional bias (π), fragment length distribution (λ), target sampling probabilities (τ), and fragment sequence (error) probabilities (ϕ) that produce the observed fragment sequences.

Some discussion of this model is required, as it does not follow be directly from Figure 1.1. For example, while size selection occurs at the end of the RNA-Seq library preparation, it is the first step in our generative model. This is a simplification following from an independence assumption that enables our estimate for λ to directly match the observed fragment lengths. Without it, the observed length distribution would need to mix λ with the other parameters in the model. Slightly different orderings of the generative model are used by both `Cufflinks` [67] and `RSEM` [32], but the changes appear to have little affect on the outcome (Figure 5.1).

2.5 Likelihood function

The generative model described above provides the following likelihood function:

$$L(\lambda, \rho, \pi, \phi|\mathcal{F}) = \prod_{f\in\mathcal{F}} \sum_{l=1}^{M_L} \sum_{t\in\mathcal{T}} \sum_{p=1}^{l(t)-l+1} \lambda_l \cdot \tau_{t|l} \cdot \pi_{p|t,l} \cdot \phi_{f|t,p,l}, \tag{2.6}$$

where $l(t)$ is the length of target t and M_L is an upper bound on fragment length, based on the technology being used.

2.5.1 Approximation

Note that in order to compute this likelihood, for each observed fragment sequence we must sum over approximately every possible fragment length at every position in the set of target sequences and compute the probability that the fragment originated from that location. Since the probability of observing a large numbers of errors quickly approaches zero, we can approximate the above likelihood with the aid of a short read aligner that efficiently identifies locations where the fragment sequences match the target nearly perfectly. For each fragment, we can then sum over this set of alignments $\hat{\mathcal{A}}_f$, ignoring those positions which do not significantly contribute to the likelihood. The approximate likelihood is then

$$L(\lambda, \tau, \pi, \phi|\mathcal{F}) \approx \prod_{f\in\mathcal{F}} \sum_{(t,p,l)\in\hat{\mathcal{A}}_f} \lambda_l \cdot \tau_{t|l} \cdot \pi_{p|t,l} \cdot \phi_{f|t,p,l}, \tag{2.7}$$

which is much easier to compute.

2.5.2 Positional uniformity

We can rewrite this function assuming positional uniformity as

$$L(\lambda, \rho, \pi, \phi|\mathcal{F}) = \prod_{f\in\mathcal{F}} \sum_{l=1}^{M_L} \sum_{t\in\mathcal{T}} \sum_{p=1}^{l(t)-l+1} \lambda_l \cdot \tau_{t|l} \cdot \pi_{p|t,l} \cdot \phi_{f|t,p,l} \tag{2.8}$$

$$= \prod_{f\in\mathcal{F}} \sum_{l=1}^{M_L} \sum_{t\in\mathcal{T}} \sum_{p=1}^{l(t)-l+1} \lambda_l \cdot \frac{\rho_t \cdot (l(t) - l + 1)}{\sum_{u\in\mathcal{T}} \rho_u \cdot (l(u) - l + 1)} \cdot \frac{1}{l(t) - l + 1} \cdot \phi_{f|t,p,l} \tag{2.9}$$

$$= \prod_{f\in\mathcal{F}} \sum_{l=1}^{M_L} \sum_{t\in\mathcal{T}} \sum_{p=1}^{l(t)-l+1} \lambda_l \cdot \frac{\rho_t}{Q(l)} \cdot \phi_{f|t,p,l} \tag{2.10}$$

$$\approx \prod_{f\in\mathcal{F}} \sum_{(t,p,l)\in\hat{\mathcal{A}}_f} \lambda_l \cdot \frac{\rho_t}{Q(l)} \cdot \phi_{f|t,p,l}, \tag{2.11}$$

where $Q(l) = \sum_{u\in\mathcal{T}} \rho_u \cdot (l(u) - l + 1)$.

When deriving the EM formulation in Chapter 3, we will expand the likelihood function to specify how the parameters can be estimated from observed data.

Chapter 3

Parameter Estimation with the EM Algorithm

3.1 The EM algorithm

The expectation-maximization (EM) algorithm [13] is an iterative method for finding estimates of the latent parameter values that maximize the likelihood of a statistical model. The method consists of an expectation (E) step for finding the expected value of the likelihood function under the current parameter estimates and a maximization (M) step for finding new parameter estimates that maximize the function. These steps are alternated, and the estimates after each iteration are guaranteed to increase the likelihood. However, the maximum likelihood is only guaranteed to be found when the likelihood function is convex.

3.2 Basic EM algorithm for fragment assignment

As mentioned in the Introduction, the first EM-based approach to ambiguous fragment assignment was introduced for assigning EST fragments to transcripts by [71]. Their model is very simple in that it only includes a single parameter, what we refer to as τ in Chapter 2. The likelihood function is then equivalent to

$$L(\tau|F) = \prod_{f \in \mathcal{F}} \sum_{t \in \mathcal{T}} X_{f,t}\tau_t, \tag{3.1}$$

where X is a compatibility matrix with a value of 1 for $X_{f,t}$ if and only if f aligns to t with a minimal number of errors.

Since errors are not modeled, it is assumed that the alignment with the fewest mismatches is the best. Aside from ignoring sequencing errors, this model is fundamentally inaccurate in that it assumes $\rho_t = \tau_t$ for all t, which is only true when all transcripts are of equal length. These simplifications help to produce a log-linear model, implying a convex function. This allows for a very straightforward implementation of the EM algorithm that guarantees

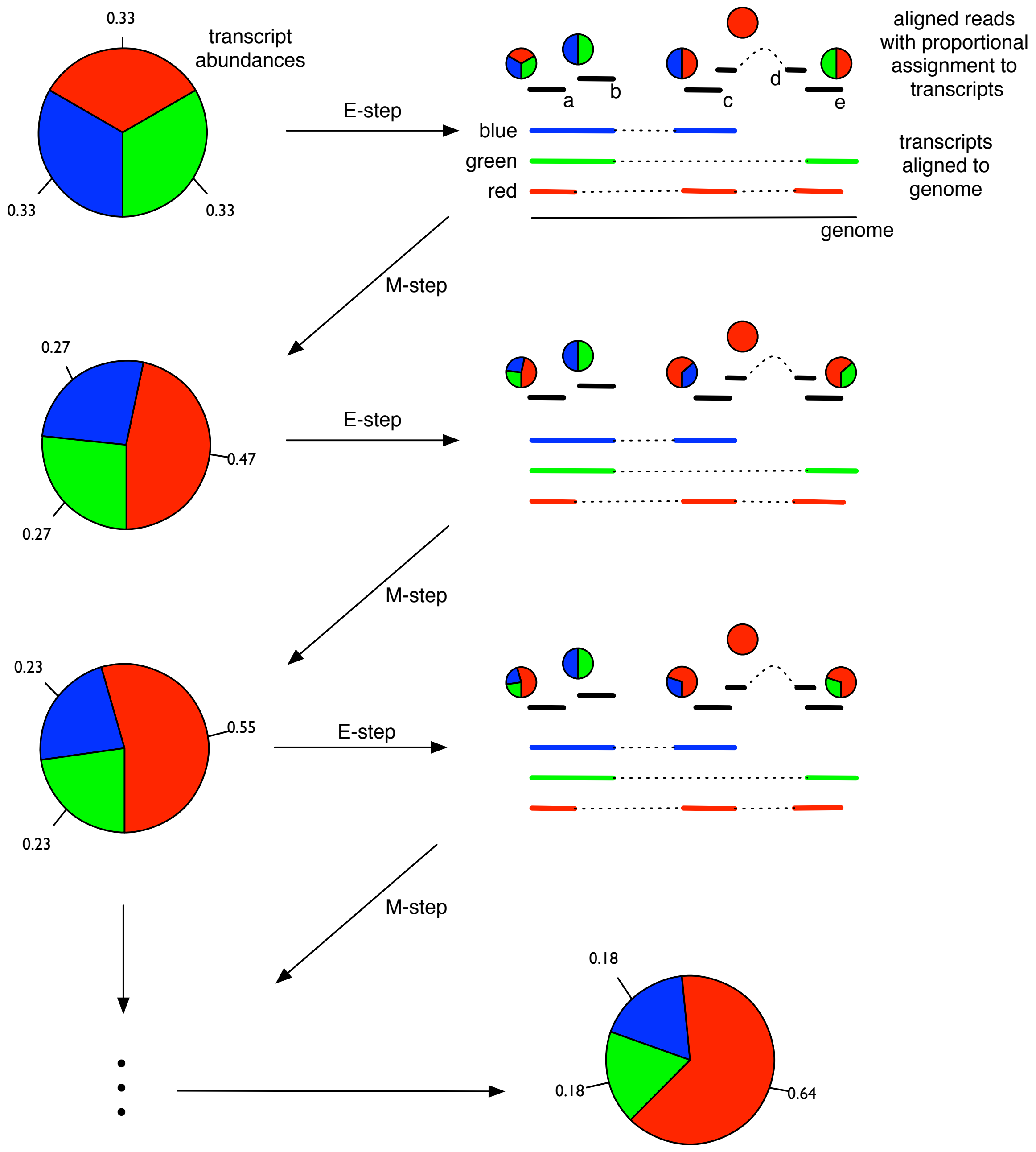


Figure 3.1: *The basic EM algorithm for RNA-Seq.* An illustration of the EM algorithm for RNA-Seq assuming no errors, uniform coverage, and equal transcript lengths. This specific example converges after 18 iterations.

convergence to the maximum likelihood solution, as illustrated in Figure 3.1, where we have made both assumptions.

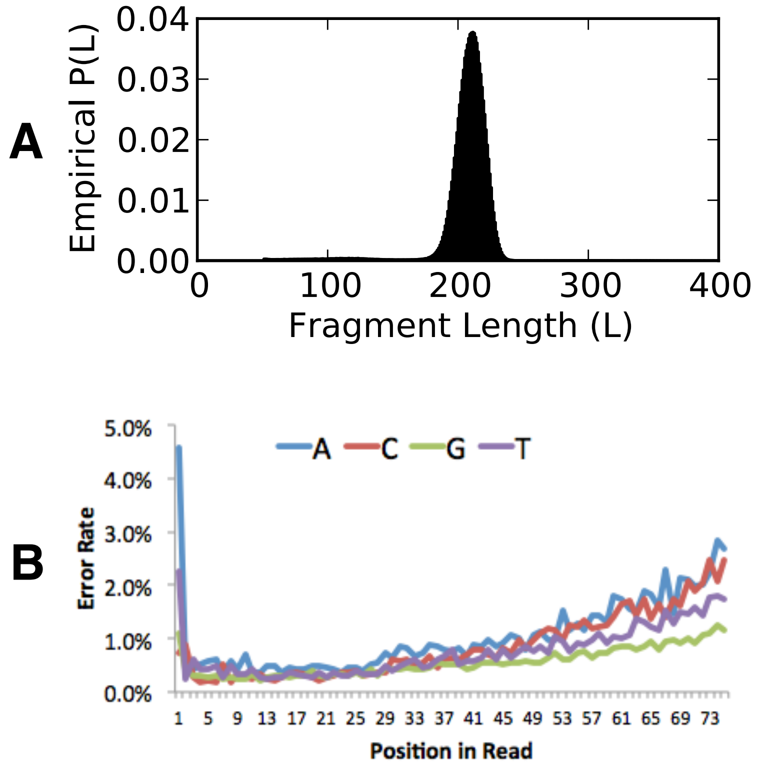


Figure 3.2: *Examples of typical auxiliary parameter distributions.* A) An empirical estimate of the fragment length distribution from a paired-end RNA-Seq experiment measured with unique alignments. The distribution is nearly Gaussian due to the size selection step of the RNA-Seq library preparation. B) An empirical estimate of the probability of a substitution error for each nucleotide at each position in a 75bp read sequenced on the Illumina platform and measured using uniquely aligning reads. Note that the error rate tends to increase towards the end of the read.

The algorithm begins by assuming a uniform (or random) abundance distribution for the transcripts. In the E-step, the fragments are partially assigned to compatible transcripts in amounts proportional to the transcript abundances. The partial assignments to each transcript are then summed and normalized to produce new transcript abundance estimates in the M-step. The E-step and M-step are repeated until the abundance estimates converge.

3.3 Auxiliary parameter estimation

Later formulations of the EM algorithm for fragment assignment [67, 31] reduced the model assumptions by including parameters for features such as fragment length, target (transcript) length, positional bias, and sequencing errors, as described in Section 2.2. These auxiliary parameters tend to vary little between runs on the same platform and library, but often vary widely between platforms and preparation methods (see Chapter 4 and [30]). Unfortunately, with the addition of these parameters to the model, the likelihood function is not guaranteed to be convex and the EM algorithm may not converge to the maximum likelihood solution.

One way around this issue is to use a different estimation procedure to estimate the independent auxiliary distributions (i.e., fragment lengths and sequencing errors). With these auxiliary parameter estimates fixed, we can use the EM algorithm as before to optimize the primary parameter of interest: the target abundances.

Treating the fragment lengths as a multinomial distribution and setting $M_L = 800$, we can estimate its 800 parameters simply by computing the empirical distribution of those fragments that align uniquely to our targets. An example of such a distribution for RNA-Seq can be seen in Figure 3.2A.

Similarly, we can treat the sequence errors as four independent models: one for base substitutions in the first read, one for base substitutions in the second read, one for insertions, and one for deletions. We model base substitutions in each position of the reads as multinomial distributions for each observed base conditioned on the target bases at the aligned position as well as one position upstream. Indels are modeled with two separate multinomial distributions of gap size (up to some maximum allowed size, $M_i = 10$). The parameters in these distributions (a total of 9,620 for 75-bp paired-end reads) can also be estimated empirically using unique alignments. Although there are many parameters, each alignment provides error information for every position in the reads, so there are effectively only 84 parameters. An example of the single base substitution probabilities is shown in Figure 3.2B.

3.4 EM formulation

Because of the large number of abundance parameters ($\sim 76,000$ in the human transcriptome for RNA-Seq) and the bias towards single-isoform genes in the above auxiliary parameter estimation procedures, we cannot rely on a simple empirical estimation for these parameters of interest. However, with the auxiliary parameters fixed, we can return to a formulation of the EM algorithm similar to the basic case of [71].

We avoid much of the mathematical detail of the formulation here and simply focus on deriving the fragment assignment probabilities used in the E-step. This is the only modification required to the algorithm described in Section 3.2 and Figure 3.1. We direct the reader to the supplementary material in [32] for a more detailed derivation of the EM estimation procedure for RNA-Seq.

We begin by rewriting the likelihood function in terms of known auxiliary parameter estimates with the uniformity assumption and short-read aligner alignment approximation.

Thus,

$$L(\rho, |\hat{\lambda}, \hat{\phi}, \hat{\pi}, \mathcal{F}) \approx \prod_{f\in\mathcal{F}} \sum_{(t,p,l)\in\hat{\mathcal{A}}_f} \hat{\lambda}_l \cdot \frac{\rho_t}{Q(l)} \cdot \hat{\phi}_{f|t,p,l}. \tag{3.2}$$

The probability of each alignment is then

$$\begin{aligned}
\hat{P}_f(t,p,l) &\approx \frac{\hat{\lambda}_l \cdot \frac{\rho_t}{Q(l)\cdot\hat{\phi}_{f|t,p,l}}}{\sum_{(t',p',l')\in\hat{\mathcal{A}}_f} \hat{\lambda}'_l \cdot \frac{\rho'_t}{Q(l')} \cdot \hat{\phi}_{f|t',p',l'}} && (3.3)\\
&\approx \frac{\hat{\lambda}_l \cdot \rho_t \cdot \hat{\phi}_{f|t,p,l}}{\sum_{(t',p',l')\in\hat{\mathcal{A}}_f} \hat{\lambda}_{l'} \cdot \rho_{t'} \cdot \hat{\phi}_{f|t',p',l'}} && (3.4)\\
&\propto \hat{\lambda}_l \cdot \frac{\tau_t}{\tilde{l}(t)} \cdot \hat{\phi}_{f|t,p,l}, && (3.5)
\end{aligned}$$

where we assume that $Q(l_1) \approx Q(l_2)$ for $l_1, l_1 \in (0, M_l]$. This is a good assumption since the difference between the length of a target and the length of a fragment alignment is usually much larger than the difference between the length of two fragment alignments, as evidenced by a typical λ distribution (see Figure 3.2A). Furthermore, even though this approximation slightly biases the assignment towards shorter fragment lengths, the difference in the value of λ will have a much larger effect on the outcome.

With this assumption and given our estimated auxiliary parameters, we initialize τ for positional uniformity as

$$\tau_t^0 = \frac{l(t)}{\sum_{u\in\mathcal{T}} l(t)}, \tag{3.6}$$

and write the update formula for τ at iteration i as

$$\hat{\tau}_t^i \leftarrow \sum_{f\in\mathcal{F}} \sum_{(u,p,l)\in\hat{\mathcal{A}}_f} \mathbb{1}(t=u) \cdot \frac{\hat{\lambda}_l \cdot \frac{\hat{\tau}_t^{i-1}}{\tilde{l}(t)} \cdot \hat{\phi}_{f|t,p,l}}{\sum_{(t',p',l')\in\hat{\mathcal{A}}_f} \hat{\lambda}_{l'} \cdot \frac{\hat{\tau}_{t'}^{i-1}}{\tilde{l}(t')} \cdot \hat{\phi}_{f|t',p',l'}}. \tag{3.7}$$

Once $\hat{\tau}_t^i = \hat{\tau}_t^{i-1}$ for all t, we can compute the final values of $\hat{\mathcal{P}}$ and $\hat{\rho}$.

3.5 Batch estimation algorithm

The computation inherent in the above update formula is clearly not efficient. Instead, the batch estimation algorithm follows the EM structure below:

1. *Short read alignment:* Align raw reads to target sequences using a short-read aligner, allowing for a small number of errors.

2. *Auxiliary parameter estimation:* Compute the empirical fragment length and error distributions using uniquely aligned reads.

3. *Expectation step:* For each aligned fragment, compute the assignment probability given the current parameter estimates, accumulating the portions assigned to each target. At iteration i, fragments are assigned using

$$\hat{P}_f^i(t,p,l) \propto \hat{\lambda}_l \cdot \frac{\hat{\tau}_t^{i-1}}{\tilde{l}(t)} \cdot \hat{\phi}_{f|t,p,l}. \tag{3.8}$$

4. *Maximization step:* Normalize the counts of partial fragments assigned to each target to update the target sampling probability estimates ($\hat{\tau}^i$).

5. *Iterate:* Repeat the previous two steps until $\hat{\tau}$ converges.

In Chapter 5 we will describe several other optimization techniques based on the one above with varying levels of success in dealing with the exponential growth of input data while producing accurate estimates.

Chapter 4

Fragmentation Bias

Early models of sequencing experiments, including RNA-Seq, assumed uniform coverage of the target sequences for simplicity [32, 67], as we have up to this point. Under this assumption,

$$\pi_{p|t,l} = \frac{1}{l(t) - l + 1}, \tag{4.1}$$

and

$$\tilde{l}(t) = \sum_{l=1}^{M_l} \lambda_l (l(t) - l + 1). \tag{4.2}$$

Nevertheless, non-uniformity was discovered early on (see Figure 4.2), and partially explained in the case of RNA-Seq by biases in fragmentation due to the unequal binding affinities of primers used for reverse transcription in the synthesis of cDNA copies of the RNA fragments [18, 55].

In this chapter we describe how we model this sequence-specific bias to derive the positional probabilities used in our likelihood calculation. We also show that by adding this correction to the `Cufflinks` model, which is nearly identical what is described in Chapter 2 without an error model (see Section 5.4), we are able to improve its ability to estimate transcript abundances. Furthermore, we show that this improvement is greater than that achieved by two other methods, `Genominator` and `mseq`, which do not use an EM-based approach.

While this section is specifically focused on modeling the biases in RNA-Seq, similar biases are apparent in other sequencing experiments [18] and this portion of our model is therefore applicable in other cases.

4.1 Background

The randomness inherent in many of the preparation steps for RNA-Seq leads to fragments whose starting points (relative to the transcripts from which they were sequenced) appear to be chosen *approximately* uniformly at random. This observation has been the basis of assumptions underlying a number of RNA-Seq analysis approaches that, in computer science terms, invert the "reduction" of transcriptome estimation to DNA sequencing [23, 32, 46, 50, 67]. However, recent careful analysis has revealed sequence-specific [18, 62] biases in sequenced fragments. Sequence-specific bias is a global effect where the sequence surrounding the beginning or end of potential fragments affects their likelihood of being selected for sequencing. These biases can affect expression estimates [34], and it is therefore important to correct for them during RNA-Seq analysis.

Although many biases can be traced back to specifics of the preparation protocols (see Figure 4.1 and [18]), it is currently not possible to predict fragment distributions directly from a protocol. This is due to many factors, including uncertainty in the biochemistry of many steps and the unknown shape and effect of RNA secondary structure on certain procedures [34]. It is therefore desirable to estimate the extent and nature of bias indirectly by inferring it from the data (fragment alignments) in an experiment. However, such inference is non-trivial due to the fact that fragment abundances are proportional to transcript abundances, so that the expression levels of transcripts from which fragments originate must be taken into account when estimating bias, as Figure 4.1 demonstrates. At the same time, expression estimates made without correcting for bias may lead to the over- or under-representation of fragments. Therefore the problems of bias estimation and expression estimation are fundamentally linked, and must be solved together. Likelihood based approaches are well suited to resolving this difficulty, as the bias and abundance parameters can be estimated jointly by maximizing a likelihood function for the data.

We describe a likelihood based approach for simultaneous estimation of bias parameters and expression levels using the likelihood framework of Chapter 2. This complements work of [18, 34] where corrections are developed based on another likelihood model, but do not take abundance information into account when estimating bias parameters. We demonstrate that our method improves expression estimates in comparison with independently obtained qRT-PCR on a benchmark dataset. Using the same data, we also show that our method improves on the approaches of [18, 34].

RNA-Seq technology is changing rapidly, and this is evident in the development of numerous preparation protocols (for a recent review see [30]) and increasingly longer read lengths from sequencing machines [25]. When assessing the impact of bias correction, we have therefore included both early RNA-Seq data of the type that many laboratories might be producing with older machines, as well as newer data that reflects recent protocol choices and demonstrates the improvements in sequencing technologies. This has required us to make our methods robust to both single- and paired-end reads, strand specific and non-specific protocols, and a variety of priming and fragmentation methods. One of our main findings is that bias correction improves the correlation of expression estimates obtained from sequence data generated using different sample preparations and different sequencing technologies.

4.2 Method

Fragment counts in an RNA-Seq experiment are determined by two different phenomena: fragments originating from highly expressed transcripts will appear more often in the data than those originating from lower-expressed transcripts, and library preparations include biases that may preferentially select some potential fragments over others. By *fragment bias* we mean only the over- or under-representation of fragments due to sequence-specific bias as discussed in Section 4.1. Because expression levels also affect fragment abundances, it is necessary to jointly estimate transcript abundances and bias parameters in order to properly learn the bias directly from RNA-Seq data.

This issue is illustrated by example in Figure 4.1 where the need for joint estimation of bias parameters and expression values is evidenced by comparison of the raw counts of bases at the starts/ends of fragments (panel A) and the adjusted counts normalized by the abundances of transcripts (panel B). The latter calculation is affected by the bias parameters, so that joint estimation is required. We expand upon the likelihood framework described in [67] in order to perform such parameter estimation, resulting in “learned” bias weights (panel D Figure 4.1) that are used to adjust expected fragment counts in the computation of abundances using our likelihood model. Figure 4.2 shows an example of how well these bias estimates capture the over- and under-representations of reads at different positions of a transcript, based on its sequence.

We now discuss how we compute the value of $\pi_{p|t,l}$, the probability of starting a fragment at position p given that the fragment is in transcript t and has length l.

4.2.1 Sequence probabilities

We first define a window of size 21 surrounding each end of a fragment, which includes the extreme base of the fragment as well as 10bp upstream and 10bp downstream of the end. Our analysis shows that most of the sequence bias exists in this region (see Figure 4.1. Furthermore, we note that the biases may differ on the 3’ and 5’ end of the transcript, so we consider them independently. To model the bias in these two windows, we must compute the probability of a given sequence s, which will denote as $\psi^{5'}_{\text{obs}}(s)$ and $\psi^{3'}_{\text{obs}}(s)$ for the 5’ and 3’ fragment ends, respectively. Given the set of partial assignments of fragments in $\mathcal{F}$, $\mathcal{A}_{\mathcal{F}}$, we can estimate $\psi^{5'}_{\text{obs}}(s)$ as the sum of the probabilities of fragment assignments where the sequence in the window around the 5’ end of the fragment is identical to s. More formally,

$$\hat{\psi}^{5'}_{\text{obs}}(s) \propto \sum_{f\in\mathcal{F}} \sum_{t,p,l} P_f(t,p,l) \cdot \mathbb{1}(s = t_{\text{seq}}[p-10, p+10]), \tag{4.3}$$

where $t_{\text{seq}}[i,j]$ represents the subsequence of target t from position i to j, inclusive.

The ψ_{obs} parameters give us the probabilities of observing certain sequences at the ends of a fragment in an experiment. Nevertheless, as stated previously, these probabilities are a function of both the sequence bias as well as the transcript abundances. To aid in removing the effect of abundance, we also require parameters for expected probabilities given known

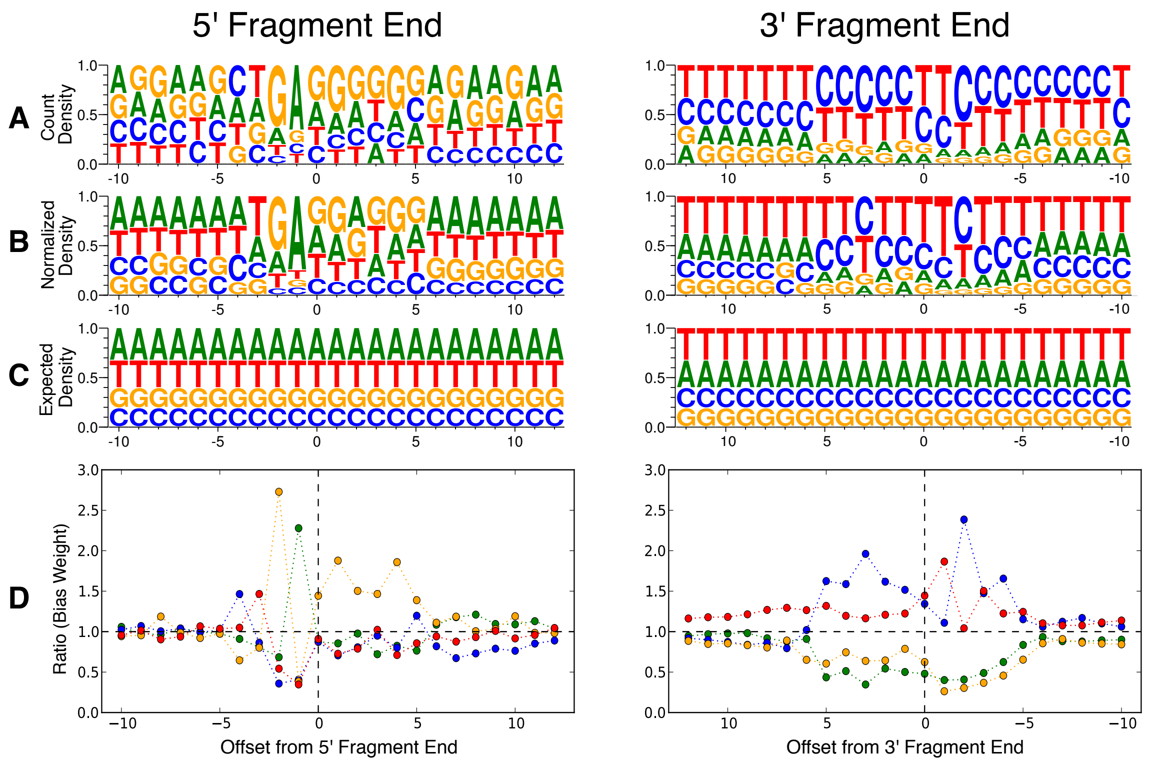


Figure 4.1: *Nucleotide distribution surrounding fragment ends and calculation of bias weights.* (A) Sequence logos showing the distribution of nucleotides in a 21bp window surrounding the ends of fragments from an experiment primed with "not not so random" (NNSR) hexamers [30]. The 3′ end sequences are complemented (but not reversed) to show the sequence of the primer during first-strand synthesis (see Figure 1.1). The offset is calculated so that 0 is the "first" base of the end sequence and only non-negative values are internal to the fragment. Counts were taken only from transcripts mapping to single-isoform genes. (B) Sequence logo showing normalized nucleotide frequencies after reweighting by initial (not bias corrected) FPKM in order to account for differences in abundance. (C) The background distribution for the yeast transcriptome, assuming uniform expression of all single-isoform genes. The difference in 5′ and 3′ distributions are due to the ends being primed from opposite strands. Comparing (C) to (A) and (B) shows that while the bias is confounded with expression in (A), the abundance normalization reveals the true bias to extend from 5bp upstream to 5bp downstream of the fragment end. Taking the ratio of the normalized nucleotide frequencies (B) to the background (C) for the NNSR dataset gives bias weights (D), which further reveal that the bias is partially due to selection for upstream sequences similar to the strand tags, namely TCCGATCTCT in first-strand synthesis (which selects the 5′ end) and TCCGATCTGA in second-strand synthesis (which selects the 3′ end). Although the weights here are based on independent frequencies, we found correlations among sites in the window and take these into account in our full model to produce more informative weights.

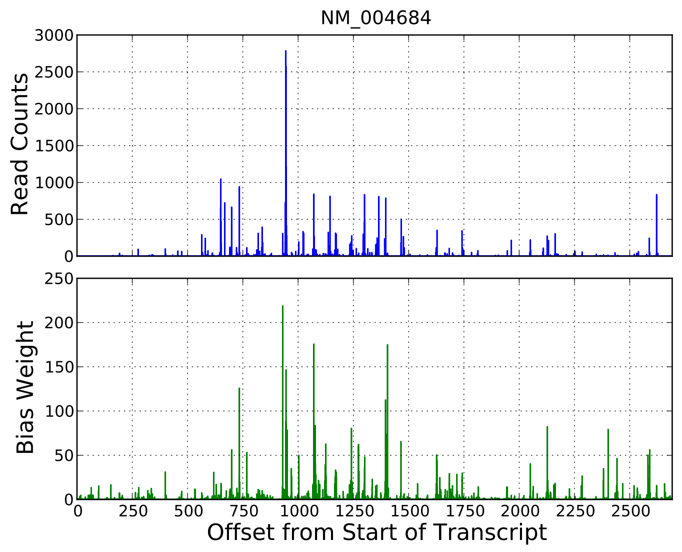


Figure 4.2: *Non-uniformity of fragmentation in RNA-Seq.* An example showing the effect of bias correction on the read counts for human transcript `NM_004684`. The top panel shows raw read counts (number of $3'$ ends of fragments at each location), and the bottom panel shows the product of the bias weights (see Section 4.2.2) computed at the same locations. We correctly identify bias at different positions and can therefore correct for the non-uniformity. Note that the bias parameters were learned from the entire dataset excluding reads mapped to this transcript in order to cross-validate our results. The RNA-Seq for the experiment was performed with the NSR protocol [2], which is why $3'$ counts were used instead of $5'$.

abundances with uniform coverage. Given a set of known relative transcript abundances, ρ, as well as the cumulative fragment length distribution, λ_{cdf}, we can compute the expected probabilities as

$$\psi^{5'}_{\text{exp}}(s) \propto \sum_{t \in \mathcal{T}} \sum_{p=1}^{l(t)} \rho_t \cdot \lambda_{\text{cdf}}(l(t) - p + 1) \cdot \mathbb{1}(s = t_{\text{seq}}[p-10, p+10]). \tag{4.4}$$

Without going into full detail, one should note that modeling the full 21bp window sequences would require 2×4^{21} parameters. We instead use a third-order Markov chain for each end, reducing the number of parameters to $2 \times (4 + 4^2 + 4^3 + 18 \times 4^4) = 9384$.

4.2.2 Positional bias weights and probabilities

Given the observed and expected sequence probabilities and assuming conditional independence between the 5' and 3' ends, we compute a bias weight at each position in the target sequences for every possible fragment length as

$$w_{t,p,l} = \frac{\psi^{5'}_{\text{obs}}(t_{\text{seq}}[p-10, p+10]) \cdot \psi^{3'}_{\text{obs}}(t_{\text{seq}}[p+l-11, p+l+9])}{\psi^{5'}_{\text{exp}}(t_{\text{seq}}[p-10, p+10]) \cdot \psi^{3'}_{\text{exp}}(t_{\text{seq}}[p+l-11, p+l+9])}. \quad (4.5)$$

Once these weights are known, the positional probability given a target and fragment length is easily derived as

$$\pi_{p|t,l} = \frac{w_{t,p,l}}{\sum_{p'=1}^{l(t)-p+1} w_{t,p',l}}. \quad (4.6)$$

Figure 4.1 shows an example of how these weights are calculated when using a 0-order Markov chain.

Note that if there is no bias (i.e., $\forall s, \psi_{\text{obs}} = \psi^{3'}_{\text{exp}}$), then $w_{t,p,l} = 1$ for all t, p, l and $\pi_{p|t,l} = \frac{1}{l(t)-p+1}$, which is the value that was used in earlier models that assumed uniformity [67, 34] and previously in Section 2.5.2. However, as Figure 4.1D illustrates, the empirical weights often vary greatly from that assumption.

4.2.3 Effective length

Aside from modifying the positional probabilities, sequencing bias also requires that we change how the target sampling probabilities, τ are computed, since a target containing higher-weighted sequences is more likely to have fragments selected for sequencing. In Section 2.3, we showed the relationship between the target abundance (ρ_t) and sampling probabilities (τ_t) via the effective length ($\tilde{l}(t)$). We update the effective length as follows to include the effects of sequence-specific bias:

$$\tilde{l}(t) = \frac{\sum_{l=1}^{M_l} \sum_{p=1}^{l(t)-l+1} \lambda_l \cdot w_{t,p,l}}{\sum_{u \in \mathcal{T}} \sum_{l=1}^{M_l} \sum_{p=1}^{l(u)-l+1} \lambda_l \cdot w_{u,p,l}}. \quad (4.7)$$

Furthermore, the target sampling probability conditioned on fragment length becomes

$$\tau_{t|l} = \frac{\rho_t \sum_{p=1}^{l(t)-l+1} w_{t,p,l}}{\sum_{u \in \mathcal{T}} \rho_u \sum_{p=1}^{l(u)-l+1} w_{u,p,l}}. \quad (4.8)$$

4.2.4 Updated likelihood

We now update our likelihood including sequence-specific fragmentation bias to get

$$L(\lambda, \rho, \psi, \phi|\mathcal{F}) \approx \prod_{f \in \mathcal{F}} \sum_{(t,p,l) \in \hat{\mathcal{A}}_f} \lambda_l \frac{\rho_t \sum_{p'=1}^{l(t)-l+1} w_{t,p',l}}{\sum_{u \in \mathcal{T}} \rho_u \sum_{p'=1}^{l(u)-l+1} w_{u,p',l}} \cdot \frac{w_{t,p,l}}{\sum_{p'=1}^{l(t)-l+1} w_{t,p',l}} \cdot \phi_{f|t,p,l} \quad (4.9)$$

$$= \prod_{f \in \mathcal{F}} \sum_{(t,p,l) \in \hat{\mathcal{A}}_f} \lambda_l \cdot \frac{\rho_t \cdot w_{t,p,l}}{Q(l)} \cdot \phi_{f|t,p,l}, \quad (4.10)$$

$$(4.11)$$

where $Q(l) = \sum_{u \in \mathcal{T}} \rho_u \sum_{p'=1}^{l(u)-l+1} w_{u,p',l}$.

4.2.5 Estimation procedure

Unlike the other two auxiliary parameters, the bias weights are dependent on transcript abundances and therefore cannot be estimated independently. Our approach is to alternate the optimization of the ρ and ψ parameters. Since the optimization procedure for ρ (the EM algorithm) is relatively time-consuming, we restrict ourselves to few (three or less) alternating rounds. The full optimization procedure is as follows:

1. *Short read alignment:* Align raw reads to target sequences using a short-read aligner, allowing for a small number of errors.

2. *Auxiliary parameter estimation:* Compute the empirical fragment length and error distributions using uniquely aligned reads.

3. *EM optimization of ρ:* Run the EM algorithm to convergence as described in Chapter 3 but using the new assignment function:

$$\hat{P}_f(t,p,l) \propto \lambda_l \cdot \frac{\hat{\tau}_t^{i-1}}{\tilde{l}^j(t)} \cdot \hat{w}_{t,p,l}^j \cdot \hat{\phi}_{f|t,p,l}, \tag{4.12}$$

   where j is the alternating round and all positional weights are 1 when $j = 0$. Note we have excluded $Q(l)$ for the same reasons as in Chapter 3.

4. *Compute bias weights:* Given the estimate $\hat{\rho}^j$, compute new estimates for ψ and update the positional weights and probabilities.

5. *Iterate:* Repeat the previous two steps several times.

4.3 Validation

We emphasize that our our goal is not to validate RNA-Seq *per se*, but rather to show that bias correction improves expression estimation. Therefore, in interpreting the correlations throughout this section, we focus on improvements in correlation with bias correction and not on the absolute value. In this regard, we report most of our results as *fraction discrepancy explained*, which we calculated by dividing the change in R^2 after bias correction by the difference of the initial R^2 from 1 (a perfect correlation). Furthermore, we mention that we observed that correlation results are sensitive to the extent of filtering of low abundance fragments and we therefore attempted to eliminate filtering in the experiments we performed.

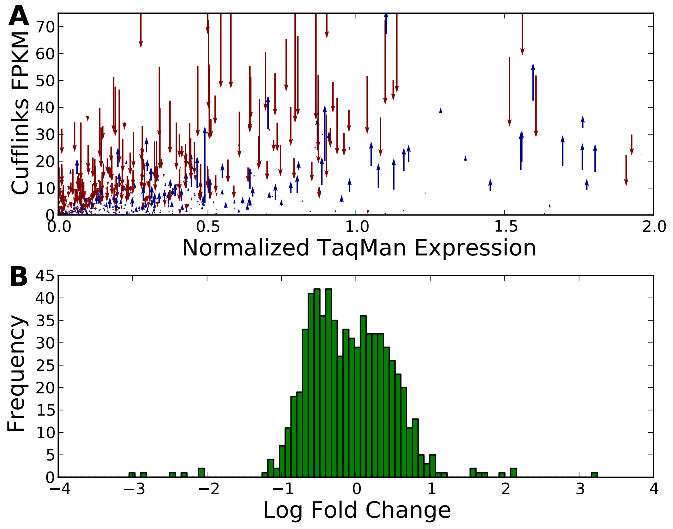


Figure 4.3: *Comparisons with MAQC qRT-PCR.* (A) Expression estimates before bias correction (tail of arrows) and after correction (points of arrows) for the SRA012427 dataset compared to qRT-PCR values for the same transcripts. Red arrows show decrease in expression after correction and blue an increase. Note that we have zoomed in on lower-expression transcripts (the majority) for clarity. (B) Distribution of log-fold change in expression after bias correction.

4.3.1 Comparison to alternative expression arrays

A major problem with validating RNA-Seq expression estimates is that there is no clear "gold standard" for expression estimation. Comparison of RNA-Seq to microarrays has suggested that the former technology is more accurate than the latter [7]. Quantitative reverse transcription PCR (qRT-PCR) has served as a benchmark in numerous studies but it is not a perfect expression measurement assay [15], and it is therefore *a priori* unclear which technology currently produces the most accurate expression estimates. Nevertheless, at present we believe it to be the best measure of expression aside from, perhaps, RNA-Seq itself. Due to the previously demonstrated superiority of RNA-Seq over microarrays, we performed all our benchmarking with respect to qRT-PCR.

We begin by comparing the expression estimates on the Microarray Quality Control (MAQC) Human Brain Reference (HBR) dataset, which includes 907 transcripts with uniquely mapping TaqMan qRT-PCR probes [60], with RNA-Seq data from the same sample sequenced by Illumina (SRA012427) [4] (Figure 4.3). When examining the correlation of the `Cufflinks` output with the qRT-PCR expression data, we observe an increase of R^2 from 0.753 before correction to 0.807 after correction.

To understand the basis for change in correlation, we further investigated, for each transcript, whether its expression estimate increases or decreases after bias correction, and by how much. The arrows in Figure 4.3 show the direction and extent of expression change with correction, and the overall fold-change distribution. Many fragments show large changes in expression with a median absolute fold change of 1.5 (Figure 4.3B).

To establish the significance of the improvement in correlation, we perform a permutation test, changing the expression estimates of transcripts *randomly* according to the fold change distribution in Figure 4.3B. We obtain a p-value of 0.0007, meaning that the improvement in R^2 our correction accomplishes is highly significant. Together, these results show that bias correction may dramatically affect expression estimates via both increases and decreases of expression values, and that these changes provide an overall improvement in abundance estimates.

4.3.2 Comparison with previous methods

In [18], a method for bias correction is proposed that is based on correcting read counts for transcripts according to the bias learned for patterns at the start of reads (normalized using sequences in the interior of reads). This approach uses less information than our method, as it is restricted to learning bias within the read sequence, and cannot capture bias surrounding the start site. Furthermore, count-based methods do not fully exploit the information available in paired-end reads which allow for the determination of fragment length. Fragment length can help in assigning ambiguously mapped fragments to transcripts and our method takes advantage of this. On the other hand, since read counts have been promoted as an acceptable way to measure abundance [1], we compared the method to ours using the MAQC qRT-PCR data from the previous section. The method of [18], implemented in the software package `Genominator`, produced a correlation of $R^2 = 0.711$ initially and $R^2 = 0.715$ after bias correction.

We also compared our approach to the `mseq` method in [34]. We again used the MAQC HBR qRT-PCR data and this time prepared the sequences and learned parameters for models following the suggested guidelines in [34], i.e. we trained the parameters of a MART model for bias by learning from the 100 most expressed transcripts in the experiment, and then tested on the set of 907 transcripts with uniquely mapping TaqMan probes. In this case, we observed an uncorrected $R^2 = 0.730$ and corrected $R^2 = 0.755$. Note that the even though the expression was again calculated using counts, the initial correlation of `mseq` is better than that of `Genominator` due to the fact that the implementation in [34] required us to remap the reads directly to the transcript sequences, which is presumably more accurate than relying on spliced mapping.

We suspect that the overall inferior results of both the `Genominator` and `mseq` in comparison to `Cufflinks` are due in part to the fact that the bias parameters cannot be learned from raw read counts, but must be normalized by the expression values of the transcripts from which the reads originate (Figure 4.1). For example, in [34], bias parameters are learned from what are estimated to be the most highly expressed transcripts based on RPKM, but these are likely to also be the most positively biased transcripts, and are therefore not representative in terms of their sequence content. We also believe that, as we argued in [67], it

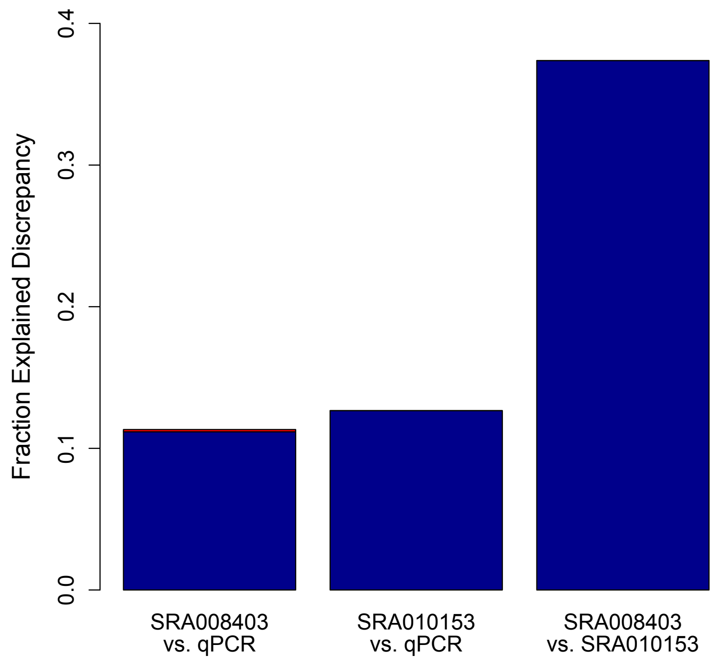


Figure 4.4: *Comparisons of technical replicates.* Results of correlation tests showing improvement after bias correction for technical replicates. Fraction Explained Discrepancy was calculated by dividing the change in R^2 after bias correction by the difference of the initial R^2 from 1 (a perfect correlation). Note that when two RNA-Seq datasets are compared, the correction was applied to both. The pairwise correlations of the four SRA010153 replicates versus qRT-PCR and SRA008403, respectively, were averaged for the figure. Even though the same RH priming protocol was used by both labs, the bias differs slightly between the preps, which is why our correction method was able to improve the correlation.

is important to account for fragment lengths in estimating expression, and read count based expression measures do not use such information. Another issue affecting `Genominator` is that instead of computing the expected read count as is done in `Cufflinks` and `mseq`, the observed read counts are adjusted. This means that in positions lacking read alignments, there is no correction of bias. We believe this may partially explain the improved performance of `mseq` in comparison to `Genominator`.

4.3.3 Technical replicates

A recurring worry with RNA-Seq has been that repeated experiments, possibly based on different libraries or performed in different laboratories, may be variable due to experimental "noise". We investigated these effects starting with an exploration of the correlation between technical replicates before and after bias correction. We define technical replicates to be the sequencing of two different libraries that have been prepared using the same protocol from a single sample. This differs slightly from some previous uses; in particular, technical replication has also referred to two sequencing experiments from the same library. Such replicates have already been shown to exhibit very little variability [1, 9].

We postulate that the differences between expression estimates from two different libraries should be reduced after bias correction and tested this hypothesis in a series of analyses whose results are shown in Figure 4.4. First, we examined libraries prepared in two different experiments from the same MAQC Universal Human Reference (UHR) sample. In the first experiment [68], which we will refer to by its accession SRA008403, the sample was sequenced from one library preparation. In the second experiment [9], which we will refer to as SRA010153, the sample was sequenced in four separate library preparations. Although the same protocol was used in all five replicates, the learned bias weights differ somewhat between the data produced by the two labs.

Figure 4.4 shows how correlations of the replicates with qRT-PCR and each other were affected by bias correction. Although the method does improve the pairwise correlations between different library preparations within SRA010153, the initial correlation is already so high (average $R^2 > 0.96$) that we only show the average pairwise correlations against qRT-PCR and the SRA008403 dataset. The greater correlation among the SRA010153 replicates as compared to the correlation between them and SRA008403 further indicates that bias is more similar when the protocol is carried out by the same lab, presumably by the same person. Bias correction clearly recovers much of the differences in quantification between the replicates introduced by fragmentation bias. Furthermore, as in the initial validation example, the correction brings both sets closer in line with the qRT-PCR standard.

4.3.4 Library preparation methods

In Figure 4.5 we demonstrate our ability to correct bias specific to libraries prepared using different protocols. For this experiment, we tried our method on several libraries from a study comparing strand-specific protocols (SRA020818) using the same yeast sample [30], as well as a dataset generated using the "not so random" (NSR) priming protocol on the human MAQC HBR sample [2]. We compared all of these datasets with a standard Random Hexamer (RH) control for the given sample.

Because the NSR dataset was sequenced from the MAQC HBR sample, we were also able to compare it to the qRT-PCR standard. We found that our method explained 33.5% of the discrepancy between an initial estimation and qRT-PCR.

4.3.5 Sequencing platforms

Previous studies on bias in RNA-Seq have focused on experiments performed with Illumina sequencers. To investigate whether bias persists with other prep and sequencing

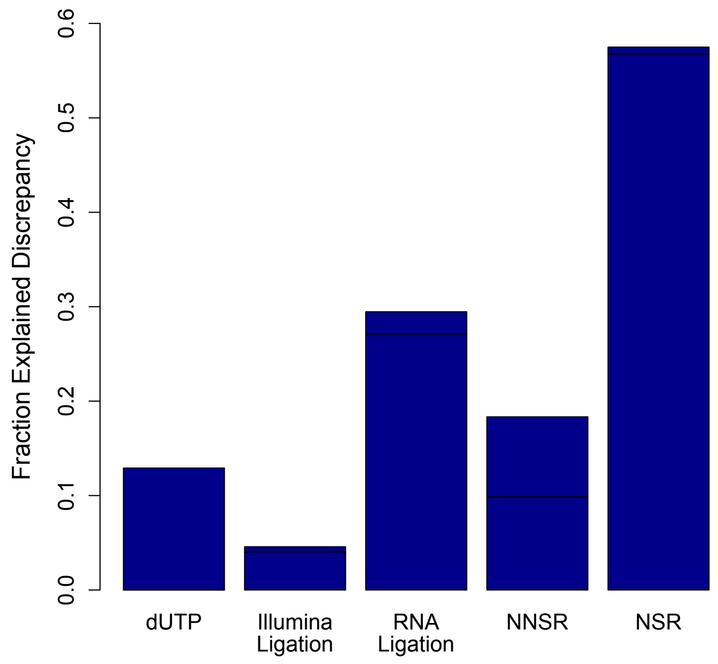


Figure 4.5: *Comparisons of library preparation protocols.* Results of correlation tests showing improvement after bias correction of datasets generated using different library prep methods, all of which are strand-specific. The first four protocols are described in [30] and the final in [2]. All datasets were compared against a control that was generated using the standard Illumina RH protocol. The first four datasets used the control from [30] with the same yeast sample. The last dataset (NSR) was compared against the HBR dataset from SRA010153 since it is also consists of single-end reads.

technologies, we examined bias in a SOLiD experiment that sequenced both MAQC samples using the standard whole transcriptome (WT) protocol. We saw clear signs of sequence-specific bias that differs from the other protocols we had examined.

We next compared the expression estimates for the SOLiD dataset with one from Illumina (accession SRA012427) before and after bias correction. In order to illustrate that our improvement in correlation does not come solely from correcting bias in the Illumina dataset, we tested whether there was some improvement from correcting one dataset at a time, as compared to simultaneous correction for both platforms. We found an increase of R^2 from 0.74 to 0.88 (Illumina correction) and 0.85 (SOLiD correction) compared to 0.94 for both. These results are summarized in Figure 4.6. While one cannot draw general conclusions based on a single experiment, we note that our approach to quantifying bias is useful for quantitatively comparing the bias among different sequencing platforms.

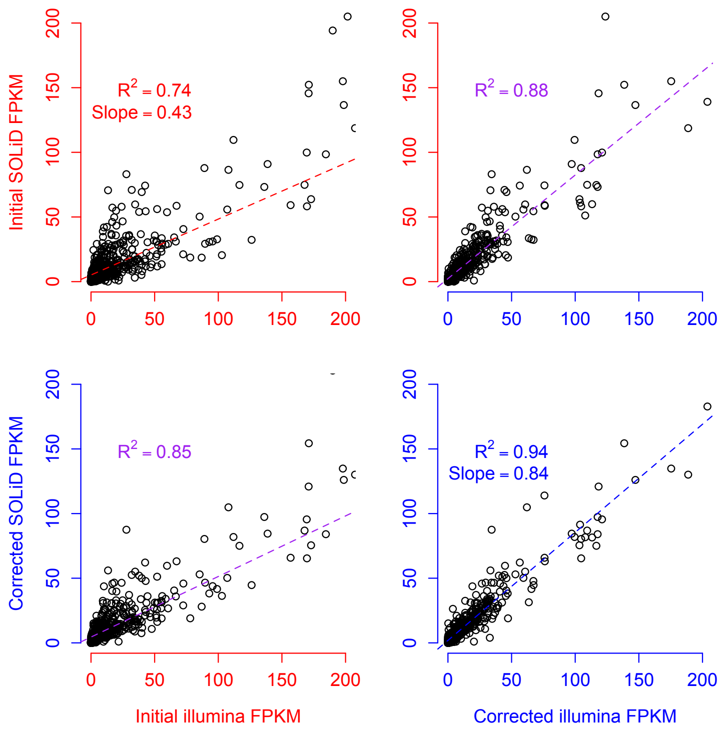


Figure 4.6: *Comparisons of sequencing platforms.* Results of correlation tests showing improvement after bias correction of datasets generated using different sequencing technologies. The Illumina dataset is SRA012427 (x-axes) and the SOLiD data is SOLiD4_HBR_PE_50x25 (y-axes). Both used the same MAQC HBR sample. Red axes and lines denote uncorrected FPKM values and blue corrected, while purple regression lines denote a comparison between corrected and uncorrected values. Both datasets are being corrected for different biases, which causes their expression estimates to become more correlated. Note that the plot is zoomed in on the lower abundance transcripts for clarity but captures over 98% of those in the experiment.

Chapter 5

Scaling the Optimization

The proliferation of high-throughput sequencing has resulted in high-volume data that is increasingly expensive to archive and unwieldy to process [63]. This data is stored as reads representing partially-sequenced DNA or cDNA fragments in either single- or paired-end form, along with quality information. The challenges brought by the rapidly increasing depths achieved in modern sequencing experiments include the feasibility of long-term archiving as well as the increase in memory requirements of informatics tools and analysis pipelines. For example, uncompressed alignments to the transcriptome for a typical human RNA-Seq experiment require approximately 1 GB of space for every million fragments sequenced. This has constrained algorithm design for high-throughput sequence analysis and is evident in one of the key computational bottlenecks in sequencing based experiments: the problem of fragment assignment [19], i.e. inference of the origin of ambiguously mapping sequence fragments. This problem was previously best addressed using the batch EM algorithm of Chapter 3 with restrictions on the extent of ambiguity allowed for multi-mapping reads [31, 67]. Limits are necessary because, unlike algorithms used for read mapping, the batch EM algorithm is not trivially parallelizable. Even with these restrictions, current methods scale poorly with sequencing depth.

Fragment assignment is a crucial step in many experiments based on high-throughput sequencing, including RNA-Seq analysis, ChIP-Seq [11], and metagenomic analysis [42]. In such applications, sequenced reads may map to many transcriptomic or genomic locations, and the resolution of ambiguity is frequently the focus of the biological question being investigated. Ad hoc heuristics in fragment assignment algorithms can produce biased results [64]. Furthermore, as sequencing depth increases, current assignment heuristics are being adversely affected by "the curse of deep sequencing. The flood of sequence not only overwhelms hardware resources but also confounds algorithms relying on heuristics that may not scale.

In this chapter we introduce how scaling was dealt with by previous methods including `RSEM` (Section 5.2) and `Cufflinks` (Section 5.4). We then introduce our method called `eXpress` (Section 5.5), which uses an online EM algorithm to solve the problem of abundance estimation and ambiguous fragment assignment in linear time with constant memory use.

Finally, we introduce `eXpress-D` (Section 5.6), which applies the *eXpress model* to a batch setting using cluster computing. We show that the `eXpress-D` provides the best accuracy in our simulations with cloud-based scaling, while `eXpress` has efficiency that is unmatched in solving the fragment assignment problem.

5.1 Comparisons

The software methods compared in this chapter vary in both their likelihood models and the optimization procedures they use. When describing these methods, we will attempt to properly attribute differences in accuracy to each of these factors.

In order to compare accuracy, four different benchmarks were used, all based on the ability of the tools to properly estimate the ρ parameters from a set of RNA-Seq reads. We introduce these benchmarks here so that they may be referenced throughout the chapter.

- Two sets (one assuming uniformity, one with sequence-specific positional bias) of a billion paired-end reads from an RNA-Seq experiment were simulated using the *eXpress model* (Chapter 2 and Figure 2.1) with parameters determined by running `eXpress` on RNA-Seq data from the ENCODE project human embryonic stem cells (cell line H1- hESC) consisting of 50,170,737 75bp paired-end reads (Accession SRX026669). `Bowtie` [29] was used for the mapping with the options -a to report all mappings, -X 800 to allow fragments up to length 800, and -v 3 to allow up to three mismatches in each read, providing proper alignments for 33,189,908 of the pairs. The 73,660 transcripts in the UCSC Genes hg19 annotation (http://genome.ucsc.edu) were used as the target set. The simulated reads together with the parameters used are available at http://bio.math.berkeley.edu/eXpress/simdata/.

- A million paired-end reads were simulated as above for the three-isoform gene UGT3A2, as annotated in human by RefSeq, in order to illustrate the convergence properties of the methods at various depths of sequencing. Figure 5.2, which presents these results, will be discussed in more detail in Section 5.5.

- To establish the effectiveness of the methods on experimental data, we re-examined previous comparisons of RNA-Seq to qPCR ([55, 31] and Section 4.3.1) from the “gold standard” MAQC dataset [60, 4]. The results are presented in Figure 5.3.

In order to compare resource usage, each algorithm was tested individually on the same 8-core Intel Xeon 2.27 GHz Mac Pro with 24 GB of RAM and 16 hyper threads. `Cufflinks` and `RSEM` were allowed 8 threads for processing, and both were run with the same options as when computing accuracy. As before, each algorithm was presented with the same multi-sized subsets of 1 billion simulated reads. For each input size, the total run time and peak memory use were measured and are displayed in Figure 5.4. `Cufflinks` and `RSEM` were halted once they crashed or began to receive memory errors. `eXpress-D` was measured separately due to being run on Amazon EC2 with different numbers of cores. Its scaling properties will

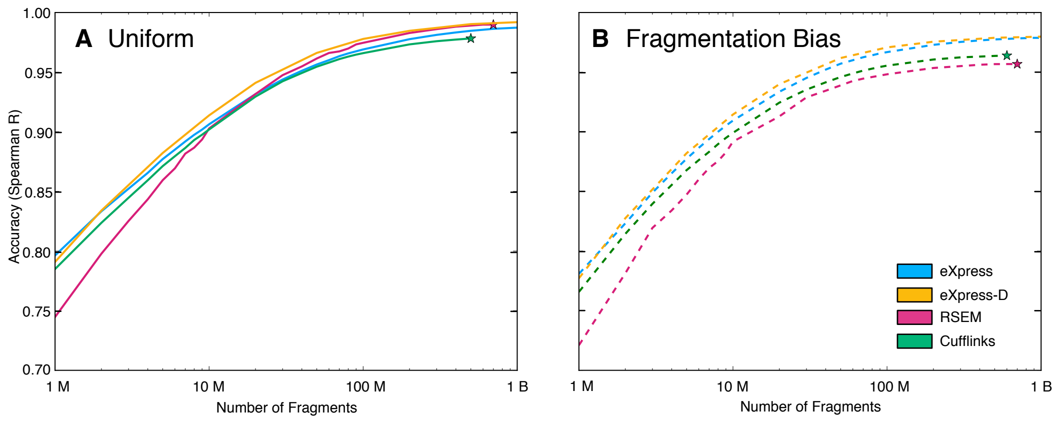


Figure 5.1: *Accuracy comparison.* Accuracy of `eXpress`, `eXpress-D`, `RSEM`, and `Cufflinks` at multiple sequencing depths in a simulation of a billion fragments (read pairs) generated with (B) and without (A) positional sequencing bias.

be explained in Section 5.6, but some details can also be seen in Figure 5.5 which represents how each algorithm scales as a function of input data using the approximated linear rate of increase.

5.2 The naïve batch approach of `RSEM`

`RSEM` takes the most naïve approach of the methods discuss here. The entire set of alignments are loaded into memory and the batch EM algorithm is executed, requiring hundreds of iterations. The only heuristic used to reduce complexity is that reads that align to more than 200 locations are ignored by the algorithm.

Figure 5.4 shows the results of this approach in red. RSEM's memory and time use grow exponentially with the size of the input alignments and quickly saturate our machine, causing it to crash when given 200 million or more fragments. However, because `RSEM` does not rely on heuristics to reduce its runtime, it produces very accurate results (Figure 5.1A) for data without sequence-specific bias. Its poor performance on biased data (Figure 5.1A and Figure 5.3) is due to these features being left out of the model and not the optimization procedure itself.

As the most directly implemented version of the EM algorithm, `RSEM` will act as a good baseline for what these methods should be able to achieve (on unbiased data).

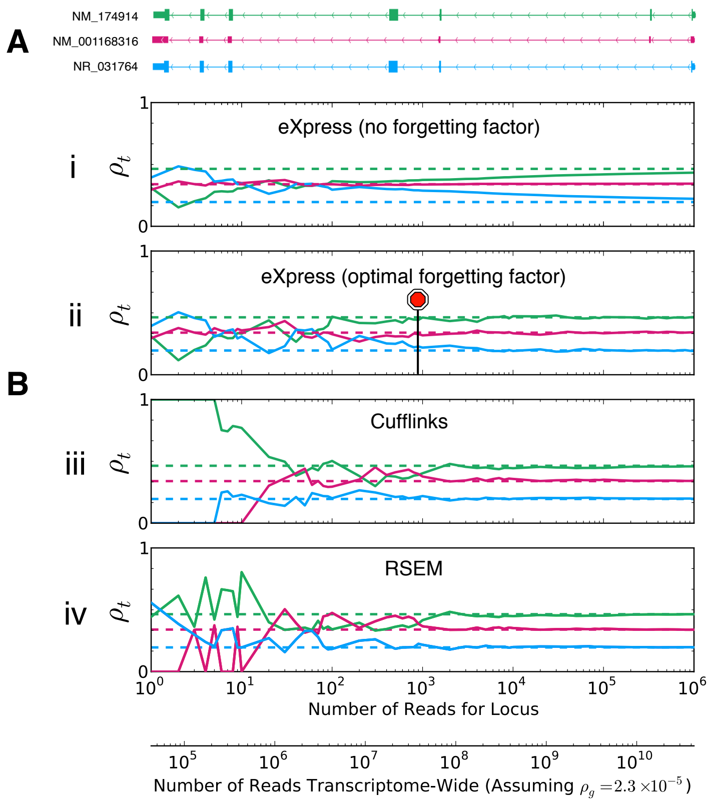


Figure 5.2: *Single gene convergence comparison.* An example of the abundance estimates produced by `eXpress`, RSEM, and `Cufflinks` for different depths of simulated data for the (A) three-isoform gene UGT3A2, as annotated in human by RefSeq. The dashed lines show the ground-truth relative abundances used for the simulation. (B*i*,*ii*) `eXpress` is only able to process each fragment once whereas (B*iii*) `RSEM` and (B*iv*) `Cufflinks` iterate over the data many times before converging to the maximum likelihood solution. Nevertheless, as more fragments are observed, all three algorithms converge toward the correct answer at approximately the same depth. In fact, `eXpress` is more robust than the batch algorithms at low depth due to its use of a prior. (B*ii*) The black line with the red octagon ("stop sign") shows where `eXpress` would automatically stop if a convergence threshold was set to 10^{-6} in terms of the Kullback-Leibler divergence between the abundance estimates at intervals of 100 fragments.

Method	w/o Bias Correction	w/ Bias Correction
`eXpress`	0.807	0.834
`RSEM`	0.791	-
`Cufflinks`	0.797	0.836

Figure 5.3: *Validation with qRT-PCR.* Spearman's rank correlation coefficients for comparisons between tested methods' abundance estimates and qRT-PCR for 907 transcripts measured by MAQC. While all methods have approximately the same accuracy, `eXpress` and `Cufflinks` benefit from the bias correction method described in Chapter 4 and in [55]. These results are concordant with the improvements due to bias correction reported in [31].

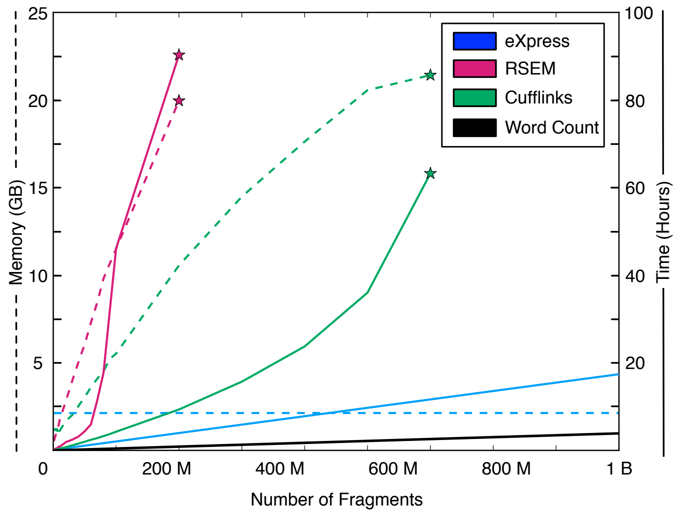


Figure 5.4: *Comparison of time and memory requirements.* The peak memory usage (dashed lines) and runtime (solid lines) for each method dependent on the number of paired-end reads input. All methods used the hg19 UCSC transcriptome annotation and were run on an 8-core Intel Xeon 2.27 GHz Mac Pro with 24 GB of RAM and 16 hyper threads. `Cufflinks` and `RSEM` were allowed 8 threads for processing, while `eXpress` only used 2. The stars represent where each of the software packages crashed or were halted due to the test machines memory constraint (24 GB). Word count is `wc`, the UNIX word count program.

Method	Runtime Slope (mpmf)	Resource Slope (cpmf)
`eXpress-D`	0.05	0.12
`eXpress`	1.8	0
`Cufflinks`	6	0
`RSEM`	27	0

Figure 5.5: *Slopes of runtime scaling.* We computed the mean slope between the runtime samples for `eXpress-D` and from Figure 5.4 to compare the scaling of the four methods in units of minutes per million fragments (mpmf). While `eXpress`, `Cufflinks`, and `RSEM` were all run on the same machine fixed at 8 cores and 24 GB RAM, the resources of `eXpress-D` were increased at a rate of 0.12 cores per million fragments (cpmf), allowing it scale in approximately constant time.

5.3 Partitioning the data

One approach to reducing the complexity and resource use of the batch EM is to partition the data and process only subsets of the data at a time, treating these subsets as independent. In Section 5.4 we will discuss a heuristic partitioning used by `Cufflinks`, but for now we introduce an "exact" partitioning that respects the independencies of the approximate (alignment-based) likelihood function.

The ambiguity graph

We can "exactly" partition the data to respect independencies in the approximated likelihood using what we call the *ambiguity graph* of the dataset.

As described in Chapter 2, in order to make the calculation and optimization of our likelihood function tractable, a read aligner is used to remove unlikely alignments from consideration, thus providing, for each fragment $f \in \mathcal{F}$, a subset of likely transcripts the fragment is derived from. In this section, we denote the mapping by $L_{\mathcal{F}\to\mathcal{T}}$, where $L_{\mathcal{F}\to\mathcal{T}}(f)$ is the set of targets that fragment $f \in \mathcal{F}$ are aligned to.

The approximation based on these alignments introduces sparsity to the inference, and allows the likelihood function to be factorized. This factorization can then be used to reduce the computation necessary for fragment assignment and abundance estimation.

In order to algorithmically leverage the sparsity of alignments, we make use of an *ambiguity graph.* In this graph, vertices represent transcripts and two vertices are connected by an edge when there is at least one ambiguous fragments aligning to the two transcripts. The ambiguity graph is defined formally as follows: It is the undirected graph $G = (\mathcal{T}, E)$ where $E = \bigcup_{f\in\mathcal{F}}\{\{u,v\}|u,v \in L_{\mathcal{F}\to\mathcal{T}}(f) \land u \neq v\}$. It is easy to show that each of the components of G define a factorization of the likelihood functions used in most RNA-Seq inference algorithms [48]. Specifically, the set of transcripts in each component can be considered independently when assigning ambiguous fragments and computing abundances.

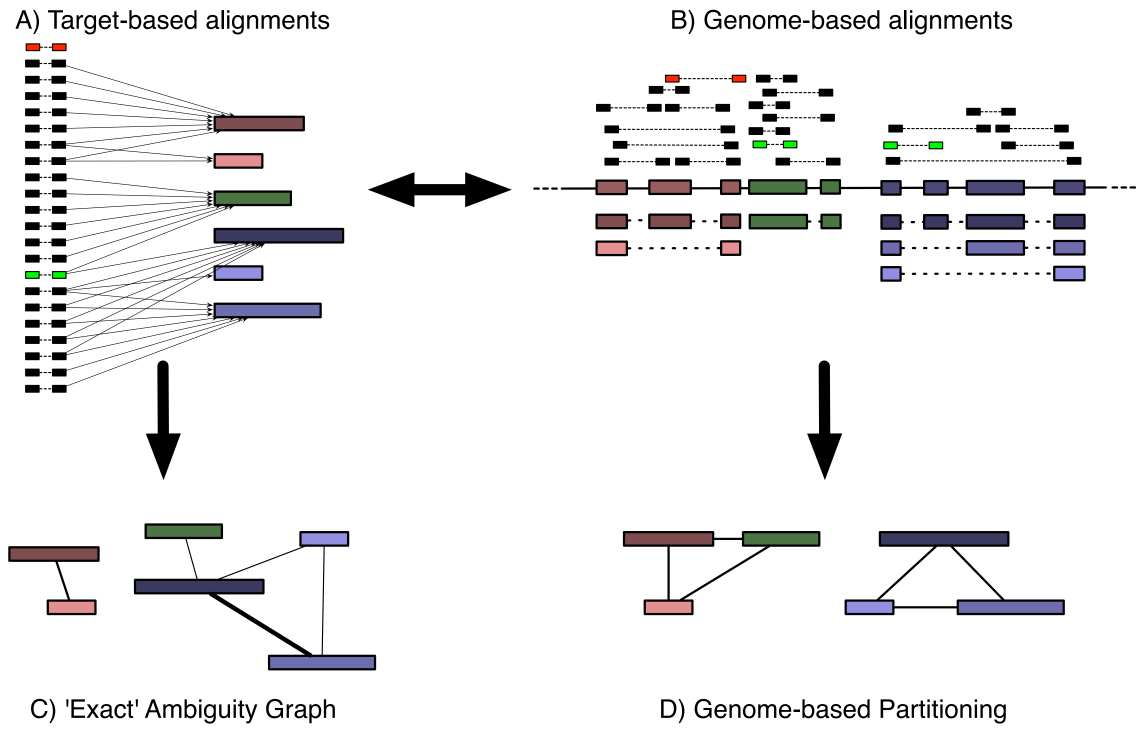

A) Target-based alignments
B) Genome-based alignments
C) 'Exact' Ambiguity Graph
D) Genome-based Partitioning

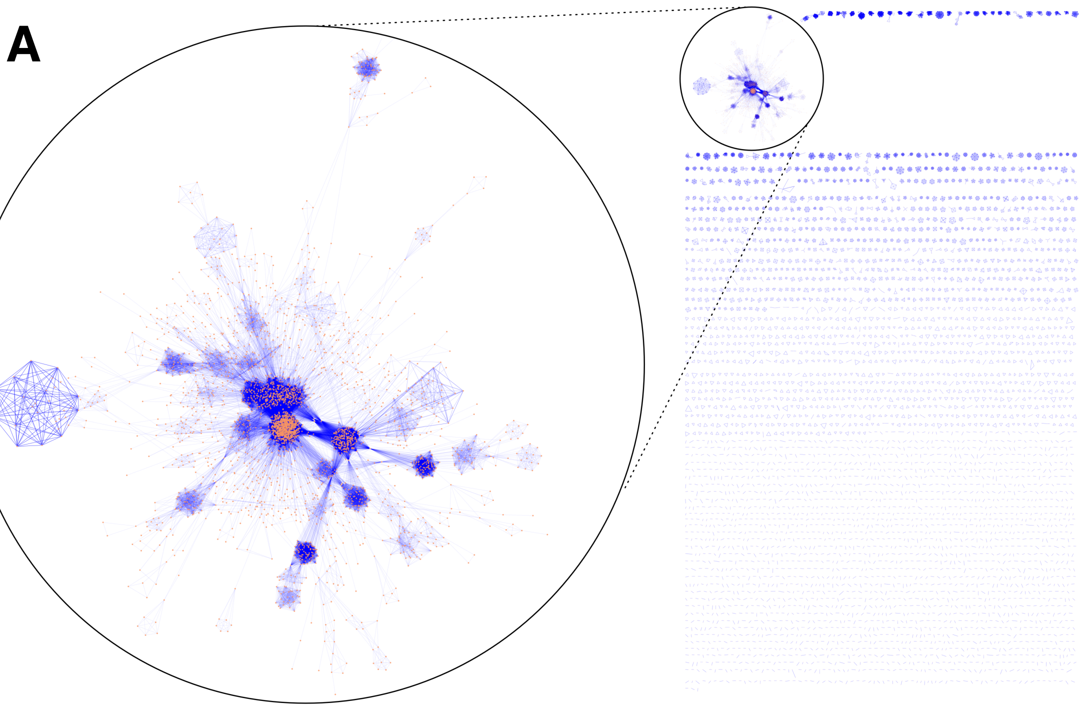
Figure 5.6: *Different alignment and partitioning methods.* (A) `RSEM` and `eXpress` require fragments to be aligned directly to the targets. (B) `Cufflinks` requires spliced alignments to the genome. Fragments may align to the genome but not the targets (red fragment) if, for example, the transcript annotations are incomplete or incorrect, which could lead to missing edges in the resulting ambiguity graph (C). Fragments aligning to multiple genomic loci (green fragment) will cause `Cufflinks` to incorrectly partition the data (D).

Figure 5.6 shows the ambiguity graph induced for an example set of alignments, and an ambiguity graph obtained for a real dataset of 60 million reads is shown in Figure 5.7.

Leveraging the ambiguity graph

Once we have defined the components of the ambiguity graph, they can be optimized sequentially using the batch algorithm, thereby reducing the memory required at any single moment to be proportional to the number of fragments in the largest component. Furthermore, smaller components can be optimized in parallel in a manner that keeps the memory use below this upper bound. We note again that because the processing respects the independencies of the data, the results will be identical to those produced by the normal batch EM algorithm.

While the size of the largest component can still be quite large (Figure 5.7B), the structure of the data visible in Figure 5.7A suggests that it can be further partitioned without discarding a significant amount of information. We explore such a technique in Chapter 6



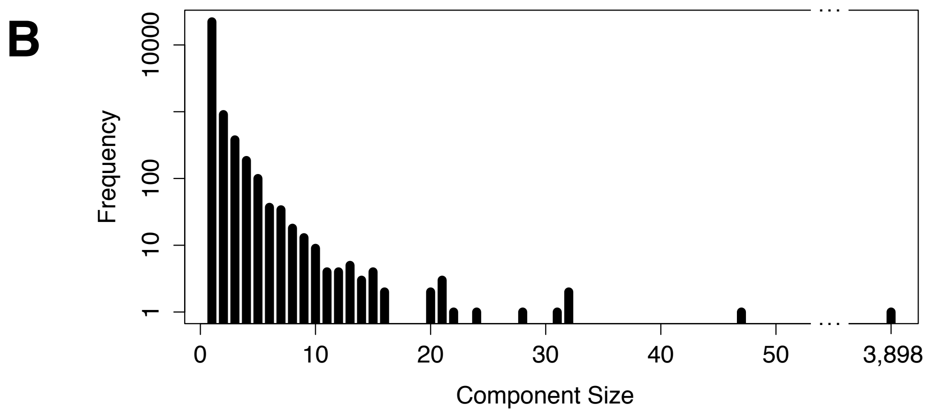


Figure 5.7: *Example ambiguity graph.* The ambiguity graph and histogram of component sizes (B) for the 60 hour time-point in [67] using approximately 30 million mapped 75 bp paired-end reads. The largest component is shown larger to illustrate its details. Note that it is partially made up of many strongly connected cliques with weak edges to the rest of the component, which is the reason for the effectiveness of the greedy partitioning algorithm we describe in Chapter 6 to reduce complexity of the update method. Edge weights are illustrated by opacity.

in the context of updating abundance estimates after re-annotation, and believe that such a method can and should be applied to the initial processing of ambiguous fragments in the future.

5.4 The heuristic partitioning approach of `Cufflinks`

5.4.1 Model

The model of Cufflinks is similar to the *eXpress Model*, except that it assumes independence between the fragment length and target sampling and lacks an explicit error model. In place of an explicit error model, there is an assumption that only the "best" alignments (the ones with the fewest mismatches) are valid. In terms of the parameters described in Chapter 2, the likelihood function appears as

$$L(\lambda, \rho, \pi, \phi|\mathcal{F}) = \prod_{f\in\mathcal{F}} \sum_{l=1}^{M_L} \sum_{t\in\mathcal{T}} \sum_{p=1}^{l(t)-l+1} \lambda_l \cdot \tau_t \cdot \pi_{p|t,l} \cdot \phi_{f|t,p,l}, \tag{5.1}$$

where ϕ assigns 1 to all alignments with the fewest mismatches for the fragment and 0 to any others.

5.4.2 Optimization

The largest difference between `Cufflinks` and the methods we've described previously is that it makes a much more extreme approximation to the likelihood function involving alignments, even beyond the mismatch heuristic described above. `Cufflinks` relies on spliced alignments to the genome instead of directly to a target set (Figure 5.6)A,B. It then breaks the set of alignments into overlapping blocks, splitting at positions where a stretch of overlaps is broken. If an annotation of features of interest is provided, these are used to preserve stretches of overlaps. In other words, the data will not be partitioned within a feature. Once partitioned in this way, `Cufflinks` then runs the batch EM algorithm separately on each block, considering only the alignments and ambiguities within each [67]. Fragments that align to multiple genomic loci are split uniformly over those sites.

In the typical case for which `Cufflinks` was developed, RNA-Seq, the features of interest are genes and their isoforms. `Cufflinks` is therefore assuming that ambiguity exists only within a gene, which allows the likelihood function to be factorized as

$$L(\lambda, \rho, \pi, \phi|\mathcal{F}) \approx \prod_{g\in\mathcal{G}} \left(\prod_{f\in\mathcal{F}} \sum_{(t,p,l)\in\hat{\mathcal{A}}_f : t\in g} \lambda_l \cdot \tau_t \cdot \pi_{p|t,l} \cdot \phi_{f|t,p,l} \right), \tag{5.2}$$

where g is a genomic locus represented by a set of transcript targets and $\mathcal{G}$ is the full set of genomic loci. With this factorization, the fragments for each gene can be optimized independently.

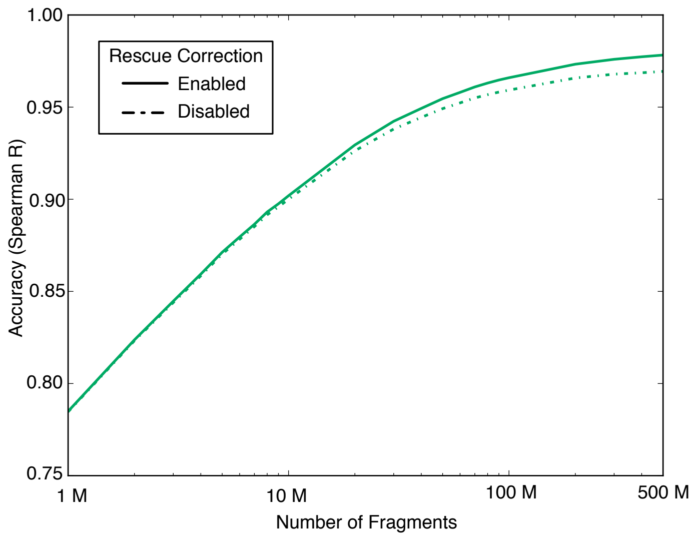


Figure 5.8: *Effect of multi-mapping reads and "rescue" correction on* `Cufflinks` *accuracy.* `Cufflinks` reduces its time and memory requirements by assuming each set of overlapping transcripts is independent of all others. Thus, a heuristic is used to disambiguate fragments that map to multiple genomic locations. The original `Cufflinks` heuristic (before v1.0, dashed line) distributed multi-mapping reads uniformly amongst the loci they mapped to. We modified the software in manner reminiscent of the "rescue method" of [44] to distribute the fragments according to the likelihood model following initial abundance estimation that uses a uniform distribution. This is equivalent to a single round of EM over the multi-mapping reads. The results show improvement at sufficiently high sequencing depth (solid line). This approach was used for all results in this chapter. Note that additional rounds of this correction will further improve the accuracy but are computationally expensive.

Performance comparison

This assumption indeed reduces the complexity and memory requirements of `Cufflinks` (Figure 5.4) but at the cost of accuracy (Figure 5.1). In fact, the results in Figure 5.1 use a slight improvement we made to the `Cufflinks` optimization that essentially uses the so-called "rescue method" of [44] to better assign multiply-aligned fragments (Figure 5.8).

In previous section (5.3), we introduced a method for finding the partition of fragments and targets that respects the independencies inherent in the approximated likelihood. Figure 5.6C,D shows how the resultant blocks differ from those based on the `Cufflinks` heuristic.

We also note that due to its use of genomic alignments and this partitioning heuristic, `Cufflinks` is more closely tied to RNA-Seq than either `RSEM` or eXpress, and no other uses for the software have been introduced as of yet.

5.5 The online EM approach of `eXpress`

In order to address the scaling issues of previous methods, we developed an online algorithm for fragment assignment based on processing data one fragment at a time. We specialized the online EM algorithm [10] to the fragment assignment problem and adapted it to work directly with estimated counts rather than relative abundances using a similar technique as described in [38]. The *eXpress model*–fully described in Chapters 2,4–includes parameters for the fragment length distribution, errors in reads (including indels), and sequence-specific biases thought to result from the fragmentation and priming steps during library preparation (see [18, 55] and Chapter 4). All of these parameters are estimated jointly with abundances enabling `eXpress` to be used for a wide range of experiments.

We will first give a basic overview of `eXpress` in Section 5.5.1 followed by mathematical details of the online optimization in Section 5.5.2 and comparisons with the above methods in Section 5.5.4. We conclude with a short discussion of the opportunities provided by the performance gains of `eXpress` in Section 5.5.5.

5.5.1 Overview

Figure 5.9 illustrates in detail how `eXpress` works. In the algorithm, each incoming fragment may map to arbitrarily many target sequences and is apportioned to the targets it maps to according to previously estimated counts. Parameter estimates for the fragment length distribution, sequence bias, and an error model for reads (including mismatches and indels)–as described in Section 2.2–are simultaneously updated. Unlike in the batch algorithm used by `Cufflinks` and `RSEM` and introduced in Chapter 3, the parameter estimates are updated after assignment of each observed fragment. To accomplish this, the parameters are treated as Dirichlet priors with the multinomial update of the assignment producing a Dirichlet posterior that is used as a prior in the next iteration.

As fragments are processed, they are assigned increasing *forgetting mass* to allow the algorithm to adapt to improving parameter estimates and partially counteract the overwhelming weight of the prior during later iterations (see Figure 5.10). This dynamic scheme–based on the theory of [10]–has favorable convergence properties that are crucial for the performance of the online algorithm (see Figure 5.2B*i*,*ii*). After an initial learning period, the auxiliary parameters are fixed due to their faster convergence (Figure 5.11), improving the efficiency of the algorithm for estimating ρ. Moreover, the convexity of the likelihood function once the auxiliary parameters are fixed guarantees that the online algorithm converges to the global maximum (see Chapter 3). More mathematical details can be found in Section 5.5.2.

To enable the use of relative abundance and count estimates in downstream applications, `eXpress` quantifies uncertainties in the estimates. Specifically, for every transcript, the posterior fragment count distribution is approximated by a shifted beta binomial distribution. The accuracy of the approximation was confirmed by simulation study. On average, we noted that counts could be estimated within 5.4% of the true value implying that for many transcripts, estimated counts obtained by `eXpress` can be used directly in differential

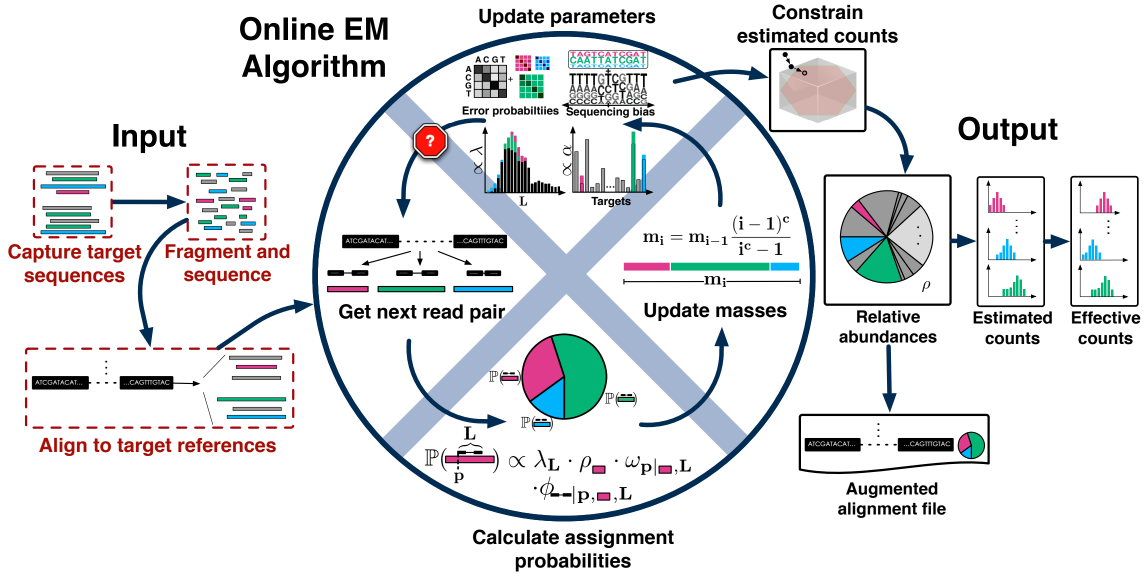


Figure 5.9: *Overview of eXpress.* The input to the program consists of reads (multiply) aligned to a set of target sequences. Either single- or paired-end reads can be processed. Fragment alignments can be streamed to eXpress, but the program can also process a SAM or BAM alignment file. If a fragment maps ambiguously to more than one target sequence, assignment probabilities are calculated for each of the constitutive alignments given previous estimates of target sequence abundances (initially a uniform prior is used). Next, a *forgetting mass* is calculated and partial "counts" are distributed to the target sequences according to the computed assignment probability. In addition to updating fractional fragment assignments to the target sequences, parameters for fragment length distribution, sequence bias, and errors in the reads are updated in a similar fashion. The updated parameters are used in the processing of the next fragment alignment. When processing of input data completes, relative abundances are calculated from the count distributions, along with distributions of estimated counts and effective counts to report the estimated number of reads that would map to each target if there were no bias. In addition, a SAM alignment file is optionally output that includes a field where the posterior probabilities of the alignments are reported. Relative abundances can be continuously examined to determine whether further sequence is needed, allowing `eXpress` to be used for real-time sequencing and analysis.

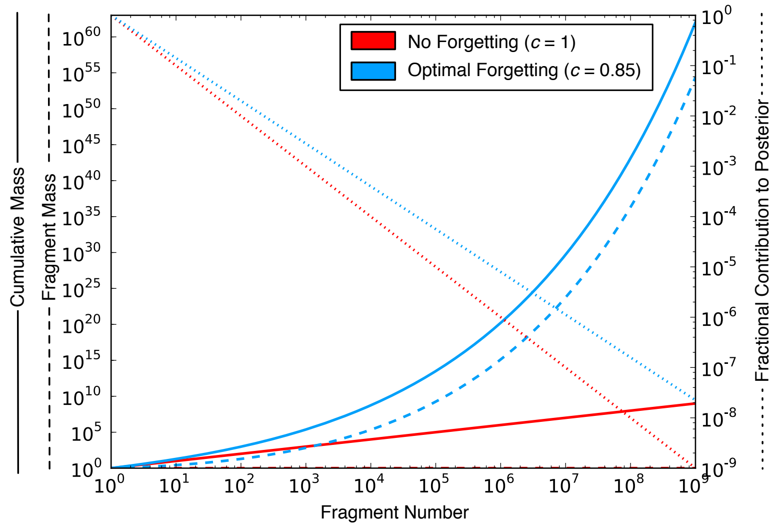


Figure 5.10: *Growth of forgetting factor.* In the simplest implementation of the online algorithm, all incoming fragments are given a mass of 1, corresponding to a forgetting factor of 1. However, as the cumulative mass grows, the fragment mass does not, and later fragment assignments have progressively smaller influence on the posterior distribution. By increasing the mass of later fragments using a forgetting factor, the fragment mass grows with the cumulative mass to reduce the effect of the prior. This allows for much faster convergence in practice, but can also cause instability if the mass grows too quickly. In this plot, we show the increase in fragment mass for a forgetting factor of 0.85, which we found empirically to give the best results in our simulations.

expression packages such as DEseq [1] that model count variability in biological replicates. When there is uncertainty in the count estimate, the count distribution can be incorporated in differential analysis [65].

When an analysis is not limited to a single pass of the data, additional information can be extracted to improve the accuracy of the resulting estimates. We tested various methods of repetitive analysis using various combinations of batch (parameter updates follow full processing of the dataset) and online EM rounds (parameter updates following each observation). The results of our tests are in Figure 5.12. Its clear that additional rounds increase the likelihood of the output results, the optimal strategy being to use an online round to seed further batch rounds.

5.5.2 Mathematical details

The online EM algorithm is an iterative algorithm that consists of computing vectors $\tau^n = \{\tau_t^n\}_{t\in\mathcal{T}}$ where $n = 1, 2, \ldots, |\mathcal{F}|$. If the fragments are ordered as $f_1, \ldots, f_{|\mathcal{F}|}$ then each

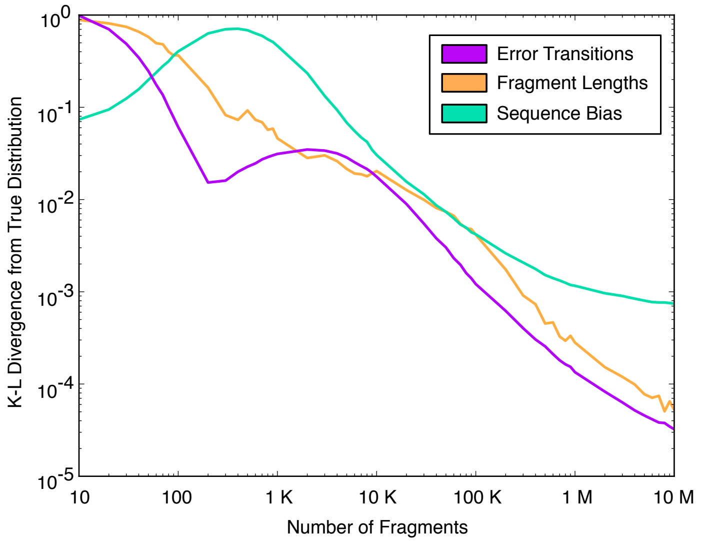


Figure 5.11: *Convergence of auxiliary parameters.* Convergence of auxiliary parameter estimates was measured by Kullback-Leibler divergence from the true distributions. In our simulations all have a divergence below 10^{-3} by 2 million observed fragments, which equates to a an average (weighted) relative ratio between the estimate and the truth of 1.001. Therefore, in our implementation we fix the auxiliary parameters and stop learning after 5 million fragments.

τ_t^n represents an estimate of the parameter τ_t after processing the fragments $f_1, \ldots, f_n$. The update procedure is given by

$$\tau^{n+1} = (1 - \gamma_{n+1})\tau^n + \gamma_{n+1}\overline{\tau}^n \tag{5.3}$$

where $\gamma_n = \frac{1}{n^c}$ for some constant $\frac{1}{2} < c \leq 1$ (called the *forgetting factor*) and

$$\overline{\tau}_t^n = P(T = t | F = f_n) = \sum_{p,l} P_{f_n}(t, p, l). \tag{5.4}$$

The probabilities in (5.4) were derived in Section 4.2.4.

Theorem 1 *[10, 38] The online EM algorithm is equivalent to stochastic gradient ascent in the space of sufficient statistics and assuming that $\frac{1}{2} < \gamma_n \leq 1$, together with mild regularity assumptions [10], converges to a local optimum.*

For fixed auxiliary parameters the likelihood in Section 4.2.4 is convex [67] and it follows that the online algorithm (also called the stepwise EM algorithm) converges to the (unique) global maximum.

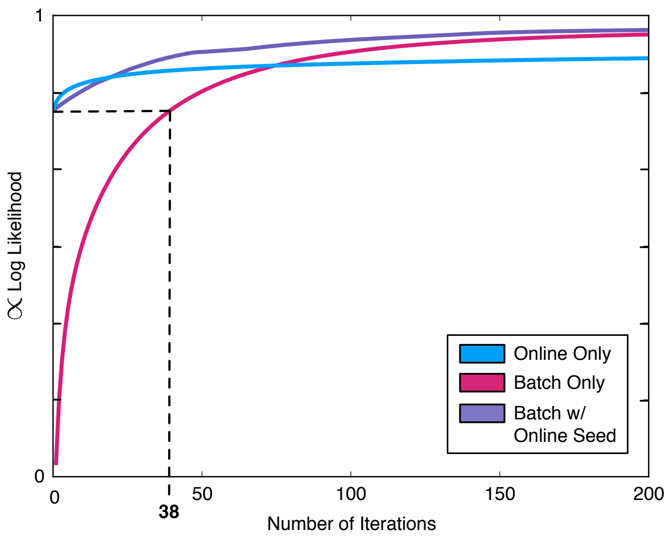


Figure 5.12: *Comparison of different iterative versions of eXpress.* The log likelihood was calculated using our model (see Section 4.2.4) on a set of 25 million simulated pair-end reads. The batch only curve (red) shows the log likelihood achieved using the standard batch algorithm where in each round all fragments are incorporated in the expectation step of the EM algorithm. The online curve (blue) shows the log likelihood achieved by repeated passes through the data as if it were additional observations (i.e., the weighting is not reset for each round). The purple curve shows a coupled method where the online algorithm is used to "seed" the parameters for the batch algorithm. The 0^{th} iteration corresponds to this seed round. Note that in this experiment, a single round of the online algorithm yields results equivalent to 38 rounds of the batch algorithm. The "batch with online seed" has superior performance to the other methods after 21 rounds.

Updating (5.3) requires $O(|\mathcal{T}|)$ operations at every step making the algorithm intractable for large numbers of target sequences. There are two reasons for this. First, computing (5.4) requires, in principle, calculation of a normalization constant that is based on a sum taken over all positions in all targets. Second, the update in (5.3) requires changing τ_t^n for all $t \in \mathcal{T}$. The first difficulty can be overcome by limiting the calculation to locations where fragments map using a heuristic alignment program such as `Bowtie` [29]. This is reasonable because the probabilities $P(T = t|F = f_n)$ are approximately zero due to the error model when a fragment does not map to a target. It is important to note that different mappers may be suitable for other types of sequencing reads (e.g. SOLiD), but the model for assignment is independent of the technology used. The second difficulty can be addressed by a change of coordinates that greatly simplifies the calculation.

We replace the τ_t^n with variables α_t^n where $n = 1, 2, \dots, |\mathcal{F}|$ and instead of iterating (5.3) we compute

$$\alpha^{n+1} = \alpha^n + m_n \overline{\tau}^n \tag{5.5}$$

where

$$m_{n+1} = m_n \left(\frac{\gamma_{n+1}}{1 - \gamma_{n+1}} \right) \frac{1}{\gamma_n} \tag{5.6}$$

is called the *forgetting mass*. When $\overline{\tau}_t = 0$ we have that $\alpha_t^{n+1} = \alpha_t^n$. Thus, the online EM algorithm scales linearly with the number of fragments analyzed, with a (small) constant number of operations per iteration.

Each vector α^n represents an estimate of the number of fragments originating from t from among the fragments $f_1, \dots, f_n$ and the α are related to the τ via

$$\tau_t^n = \frac{\alpha_t^n}{\sum_{r \in \mathcal{T}} \alpha_r^n}. \tag{5.7}$$

The α estimates can also be interpreted as parameters of Dirichlet distributions, providing a Bayesian interpretation of the online EM algorithm [38]. In `eXpress` the online EM algorithm is used to estimate the auxiliary parameters alongside the abundances.

The basic algorithm for updating the α estimates (illustrated in Figure 5.9) at step n is as follows:

1. Compute the probabilities of each alignment of the current fragment given the current parameter estimates and using

$$\hat{P}_{f_n}(t, p, l) \propto \lambda_l \cdot \frac{\hat{\alpha}_t^{n-1}}{\tilde{l}^j(t)} \cdot \hat{w}_{t,p,l}^j \cdot \hat{\phi}_{f|t,p,l}. \tag{5.8}$$

2. Increment the appropriate parameters of the α distribution by the product of the current forgetting mass and the proportion of the fragment assigned to the given target, i.e., for each target f_n aligns to,

$$\alpha_t^{n+1} \leftarrow \alpha_t^n + m_n \overline{\tau}_t^n. \tag{5.9}$$

   For any target, u, that f_n does no align to,

$$\alpha_u^{n+1} \leftarrow \alpha_u^n. \tag{5.10}$$

3. Repeat until $n = |\mathcal{F}|$.

5.5.3 Allele-specific expression

In the realm of RNA-Seq, a specialized experiment involves comparing the abundances of the transcripts from multiple haplotypes of an individual, which may differ at locations of SNPs or indels. Due to the small number of differences between the haplotype transcripts of an individual, the online estimation procedure is extremely slow to converge. Essentially useless data in the form of perfectly matching reads overwhelms the informative fragments that overlap varying sites. While not helpful in differentiating the haplotypes, these non-informative reads must be taken into account when estimating their abundances relative to other transcripts.

We have implemented a special procedure for estimating the abundances of haplotype transcripts. When enabled, the method keeps track of any informative fragments relative to the transcripts and computes the relative abundances between the haplotypes using the closed form, maximum likelihood solution. A faux transcript is presented to represent the haplotypes relative to other transcripts, with these abundances computed in the usual, online manner.

5.5.4 Performance comparison

We first examine the convergence of `eXpress` for a three-isoform gene in Figure 5.2. While `eXpress` appears to be converging towards the truth in (Figure 5.2B*i*), it will require a huge number of reads, especially compared to `RSEM` and Cufflinks. However, when a good forgetting factor is used to weight the observations, `eXpress` is able to rapidly converge to the truth (Figure 5.2B*ii*). While choosing too large of a forgetting factor leads to slow convergence (as in Figure 5.2B*i*), choosing too small of a factor leads to instability causing `eXpress` to never converge. Nevertheless, having empirically selected a forgetting factor, we expect good convergence behavior as shown in Figure 5.2B*ii*. Note that while `eXpress` is able to converge to the truth with approximately the same amount of data as `RSEM` and Cufflinks, it is only examining each fragment once where as `RSEM` and `Cufflinks` require many iterations.

Returning to Figure 5.4, most striking is the performance of `eXpress`, whose running time is linear in the number of fragments while requiring memory proportional only to the size of the transcriptome. This is similar to the UNIX word count `wc` program that simply counts the number of characters in a file. However, counting the number of fragments mapping to target sequences (without fragment assignment) cannot be used as a proxy for abundance [65].

In terms of accuracy, Figure 5.1A shows that `eXpress` is competitive with `RSEM` and outperforms `Cufflinks` (for reasons discussed in Section 5.4) when simulating without fragmentation bias. When sequence-specific positional bias is simulated (Figure 5.1B), both `eXpress` and `Cufflinks` are more accurate than `RSEM`, which does not model it. Figure 5.3 backs up the simulation of bias based on the model in [55] and Chapter 4 by showing the correction implemented in `Cufflinks` and `eXpress` also improves results for real data.

5.5.5 Discussion

In practical terms, the combined speed and accuracy of `eXpress` means that it can be used in the analyses of much deeper sequencing experiments than previously possible. Moreover, the ability of `eXpress` to accurately assign fragments using a streaming algorithm means that it is compatible with sequencing technologies that produce reads incrementally. Although current high-throughput sequencing technologies such as Illumina produce reads in parallel while serially adding bases, novel single molecule technologies promise to produce reads incrementally [8]. In a dynamic sequencing pipeline, `eXpress` could be coupled directly to a sequencer and be used to estimate abundances of target sequences in real time as individual fragments are sequenced.

We examined the use of stopping criteria based on the relative increments of the global likelihood or the local likelihood for a group of target sequences and found that `eXpress` can automatically determine when sufficiently many reads have been processed to guarantee convergence (Figure 5.2B*ii*), thus avoiding the complicated issue of choosing a sequencing depth. We note that such an approach to high-throughput sequencing also eliminates the need for storing read sequences, providing an alternative to cloud-based bioinformatics [63].

5.6 The distributed batch EM approach of `eXpress-D`

While these solutions have all used algorithms and heuristics to deal with the memory issue, another approach in computer science to dealing with the increasing size of datasets is to scale up the compute resources. We have implemented the batch EM optimization of the *eXpress model* for a distributed computing environment using the open source Apache Spark [73] cluster computing system in a software package called `eXpress-D`.

5.6.1 Background

While these solutions have all used algorithms and heuristics to deal with bounded computer memory resources, another approach is to handle the increasing size of datasets by scaling up the compute resources. It is currently infeasible for every small lab to purchase machines with enough RAM to fully analyze today's datasets using the batch EM algorithm. However, large clusters of compute nodes are now available for relatively low cost with pay-by-use cloud platform services, such as Amazon's Elastic Compute Cloud (EC2). Developing software to take advantage of the distributed resources on clusters of commodity machines is nontrivial, as issues such as failure recovery and communication must be dealt with [20].

MapReduce is an abstraction that allows developers to access the power of large distributed commodity clusters without having to explicitly handle details such as data partitioning, work scheduling, and software and hardware failures. The MapReduce programming model involves a series of calls to primitive map and reduce methods, with reordering and grouping allowed between. MapReduce was originally conceived by Google [12] in conjunction with the Google File System (GFS) [16], a fault tolerant, distributed file system–the “disk” that MapReduce utilizes. Both inspired open-source counterparts that compose the core Apache Hadoop project: Hadoop MapReduce and the Hadoop Distributed File System (HDFS) [61].

`Myrna` [27] applies Hadoop MapReduce to the analysis of RNA-Seq data, using Hadoop to count the unique alignments in an experiment. The map phase iterates through the alignments, emitting a tuple identifying the transcript that each fragment is aligned to. In the reduce phase, the unique alignments for each transcript are accumulated to produce the total counts. Since the fragments can be processed independently in the map phase, Hadoop can distribute the fragments randomly to multiple nodes. In the reduce phase, Hadoop can be set to automatically assign tuples for each transcript to the same node, allowing the accumulations to occur in an independent, distributed manner.

This method could be also extended to handle ambiguous mappings by implementing the EM algorithm using many iterations of MapReduce. The map phase would correspond to the E-step, in which a tuple is emitted for each alignment specifying the target and the probability that it is origin of the fragment based on the likelihood model and a set of global parameter estimates. The reduce phase would correspond to the M-step, in which the probabilistic assignments would be accumulated and the values normalized to produce the updated maximum likelihood parameters estimates for use in the subsequent iteration.

However, the problem with implementing the EM algorithm using Hadoop MapReduce is that the system is not tailored for data reuse. In Hadoop, the dataset being scanned is re-read from disk before every map step, and the results of intermediate computations are written to temporary files after the map. In our application, EM would be implemented as a map task. This means that the alignments would have to be loaded from disk before the E-step and a partial set of probabilistic assignments would have to be written to disk after the M-step. Then, on a single node executing a reduce task, partial sets of probabilistic assignments are fetched from the temporary files on map nodes and loaded into memory for rendezvous and normalization. An on-disk file containing likelihood parameters would also be updated during this reduce step. Thus, disk operations done in Hadoop’s map and reduce tasks create a significant bottleneck.

The approach we take instead is to use Apache Spark, an open-source framework that provides in-memory, fault-tolerant cluster computing by implementing *resilient distributed datasets* (RDDs) [73]. Spark is an alternative compute engine to Hadoop that implements the MapReduce abstraction by allowing users to apply map and reduce functions over RDDs. In conjunction with the a distributed file system–such as GFS, HDFS, or Amazon’s S3–Spark handles all issues of fault tolerance and partitioning across the cluster nodes. Unlike MapReduce, however, once a subset of the data is read from the filesystem into memory, it can be made to persist in the RAM of the compute nodes, allowing an application to efficiently scan it throughout many iterations.

Furthermore, Spark provides two types of shared variables based on common use cases that are well suited for the workflow of the EM implementation: broadcast variables and accumulable variables. A broadcast variable is a read-only piece of data that is distributed to all worker nodes. An accumulable variable references an append-only data structure that is updated by each worker node's local process and then fully combined by the process running on the master node. Broadcasted and append-only data structures both persist in-memory. `eXpress-D` utilizes these shared variables to distribute and update parameters and accumulate probabilistic assignments. The following section contains more detail on the implementation.

When given enough RAM, consecutive map executions, broadcasts, and accumulations can avoid disk spilling, which makes Spark particularly well-suited for the EM algorithm [21]. By implementing the EM algorithm for ambiguous fragment assignment using Spark, in conjunction with Amazon S3 for persistent storage, we can easily scale the method to very large datasets by combining the resources of multiple compute nodes, providing in-memory storage of alignment data while also taking advantage of large-scale parallel computations.

5.6.2 Method

Our implementation maximizes the likelihood of the *eXpress Model* (Chapters 2,4).

Preprocessing with `eXpress`

By default, the distributed file system partitions a dataset stored as text using line breaks to delineate discrete units of processing. In our case, a discrete unit is the collection of alignments of a single fragment for the alignment file, and the name and sequence of a single target for the target file. Since the commonly used formats for alignments and targets (SAM and FASTA, respectively) do not conform to this standard, we must pre-process the files to produce inputs that can be partitioned by the file system. At the same time, we wish to make our input files as small as possible to reduce the time required for network transfers.

To achieve these goals, we modified `eXpress`–which has already been optimized for parsing the standard SAM and FASTA files–to produce input files compatible with our method. The format of these new files are newline-delimited, serialized Protocol Buffers, which are encoded in base64 to ensure no newline characters appear in the serialization itself. The Protocol Buffer specification is shown in Figure 5.14 for both alignments and targets. We have avoided including any unnecessary or redundant information and compressed nucleotide sequences to byte arrays, requiring approximately 2 bits per nucleotide. The resulting files are significantly smaller than the original binary SAM (BAM) and FASTA files.

Once the input files are loaded into HDFS or S3 on the cluster, our application can be run on Spark to begin fragment assignment. Figure 5.13 outlines the procedure, which is described in more detail in the following subsections.

Preprocessing on Spark

The input files are parsed by Spark and loaded into the memory of the slave nodes as RDDs. The per-alignment indices for accessing the relevant elements of the error and bias Markov chain parameter matrices are then precomputed and stored in a transformed RDD.

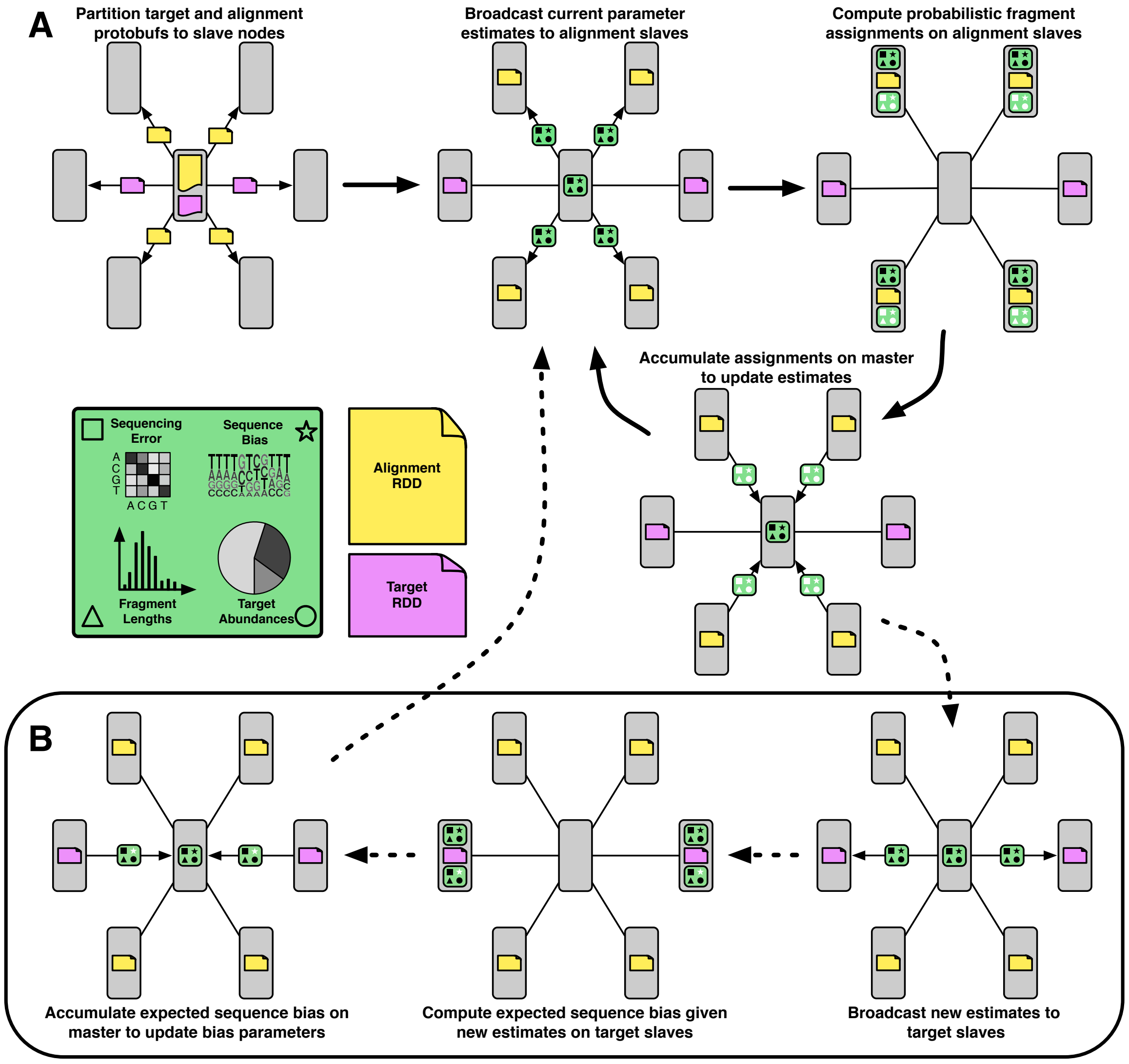

A
Partition target and alignment protobufs to slave nodes
Broadcast current parameter estimates to alignment slaves
Compute probabilistic fragment assignments on alignment slaves
Accumulate assignments on master to update estimates
Sequencing Error
Sequence Bias
A C G T
A C G T
Fragment Lengths
Target Abundances
Alignment RDD
Target RDD
B
Accumulate expected sequence bias on master to update bias parameters
Compute expected sequence bias given new estimates on target slaves
Broadcast new estimates to target slaves

Figure 5.13: *Overview of `eXpress-D`.* The top portion (A) shows the procedure for running the distributed batch EM algorithm with Spark [73] ignoring sequence-specific bias. First blocks of partitioned alignments (yellow) and targets (magenta) are distributed to the slave nodes and loaded into RDDs. An initial set of parameter estimates (green with black symbols) are broadcast to the slaves with alignments. The alignments on each slave are probabilistically assigned and new parameter estimates are partially accumulated (green with white symbols) on each node. These are then sent back to the master to be fully combined and re-broadcast for the next round. When sequence-specific bias is enabled, additional processing (B) takes place between some rounds. The parameter estimates are sent to the slaves with targets and the expected sequence bias given the current abundance estimates are accumulated and normalized on the master node to produce updated weights (see Section 4.2). Auxiliary parameters (error, bias, and fragment lengths) are fixed after 1000 rounds. The estimation procedure stops when convergence of the abundance parameters is reached.

Field	Type	Description
Fragment		
name	string	Unique query name of fragment in SAM file
paired	bool	Boolean specifying if both ends were sequenced
alignments	FragmentAlignments	Collection of alignments for fragment
FragmentAlignment		
target_id	uint32	ID of target aligned to (index in SAM header)
read_l	ReadAlignment	Alignment information for 5' (left) read, if exists
read_r	ReadAlignment	Alignment information for 3' (right) read, if exists
ReadAlignment		
first	bool	Boolean specifying if this end was sequenced first
left_pos	unit32	0-based left endpoint of alignment to reference
right_pos	unit32	0-based right endpoint of alignment to reference
mismatch_indices	byteArray	Positions in read that differ from reference
mismatch_nucs	byteArray	Nucleotides in read at mismatches, 2 bits/nuc
Target		
name	string	Unique name of target sequence
id	uint32	Index of target in SAM header
length	uint32	Number of nucleotides in target sequence
seq	byteArray	Nucleotides of target sequence, 2 bits/nuc

Figure 5.14: *Specification of alignment and target Protocol Buffers.* `eXpress` pre-processes the input data (SAM/BAM and FASTA file) and converts it to a format that is compatible with the distributed file system's partitioning scheme. The information for each target and fragment are put into a space-efficient Protocol Buffer, keeping only the information necessary for optimization, which is then serialized and encoded in base64. Each target or fragment takes up exactly one line in the file created for input into `eXpress-D`.

Each partition of the transformed RDD is approximately 700 megabytes and stores about 1 million fragments.

Processing without bias correction

The algorithm for processing without bias correction is depicted at the top of Figure 5.13. The current target abundance, error substitution Markov chain, and fragment length distribution estimates (all initially set to be uniform) are broadcast to the slave nodes storing alignment RDDs. Given these distributions, the fragments on each slave are probabilistically assigned to the aligned targets using the likelihood function from Section 2.5. The appropriate categories of the latent distributions are incremented by the posterior probabilities of the assignments at each slave node to produce new empirical distributions. These counts are then accumulated by the master node and Laplace smoothing is applied before they are normalized. The updated parameter estimates are then broadcast to the slave nodes and the procedure is repeated until convergence is detected (see below).

Processing with bias correction

Previous work demonstrates that significant improvements in accuracy can be attained by modeling sequence-specific bias (Chapter 4 and [55, 31]). We have included a bias correction mode (enabled by default) to take advantage of these improvements, as illustrated at the bottom of Figure 5.13. The primary algorithm remains the same as outlined above with the addition of Markov chain parameters modeling the sub-sequences surrounding the 5' and 3' fragment ends. Estimates of these parameters are broadcast to the slaves, used in the likelihood calculation (Section 4.2.4), and updated empirically, similar to the other hidden parameters. Instead of probabilities, the bias parameters are ratios of the observed to expected frequencies of these sub-sequences and are used as weights in the likelihood function. The observed frequencies are accumulated empirically along with the other parameters as described above, but the expected frequencies must be computed by sliding windows along the target sequences and counting the occurrences of various sub-sequences weighted by the current target abundance and fragment length parameter estimates. To make these repeated updates efficient, we broadcast the current model parameter estimates to the slave nodes storing target RDDs and have them compute local frequencies based on sliding windows over the RDDs in memory. The frequencies are then accumulated by the master node, allowing the bias weights to be updated before the next iteration.

Freezing of auxiliary parameters

We define the auxiliary parameters to be all parameters of the model except for the target abundance parameters, which are the main parameters of interest. As previously discussed in Chapter 3, there are two reasons for freezing the auxiliary parameters after a suitable number of iterations:

1. The auxiliary parameters can be estimated accurately much earlier than the target abundance parameters since they are fewer in number. Otherwise the algorithm will be wasting a significant amount of time unnecessarily updating their distributions at later iterations.

2. The model is only convex given fixed auxiliary parameters. Since we repeat the EM steps until convergence is reached, we want a guarantee that processing will not continue indefinitely. Given fixed auxiliary parameters, the likelihood function is log-linear, and the EM algorithm is guaranteed to converge to the maximum likelihood solution.

We therefore have chosen to use the following auxiliary parameter update scheme: The parameters are updated at every iteration for the first 20 and are then only updated every 100 iterations until 1000 iterations are reached, at which point they are frozen.

Numerical stability

To avoid underflow, all probabilities distributions are logged before being used in likelihood computations, which has the extra benefit of allowing the use of faster additions instead of multiplications. The assignment probabilities are exponentiated before incrementing the empirical distributions, since there is no concern of numerical instability in the update step.

Convergence detection

We halt the algorithm when convergence of the target sampling probabilities is detected in a manner similar to [34]. The parameters are considered to have converged when all targets with a sampling probability of at least 10^{-7} have a relative change of no more than 10^{-2} between two consecutive iterations.

5.6.3 Results

Cluster and experiment setup

For running experiments, we used Amazon EC2 clusters comprising m3.2xlarge instances, each of which has 8 virtual CPUs and 30 GB of memory. A virtual CPU is rated at 3.25 EC2 Compute Units (ECU), which is roughly equivalent to a 1.0-1.2 GHz 2007 Xeon processor. Even though 30 GB may seem excessive, we found that it was necessary to avoid full, costly garbage collection runs by the Java Virtual Machine (JVM) that Scala runs on, which could delay each iteration by tens of seconds.

A cluster was launched for subsets of various sizes of each test dataset. Starting from 3 slave nodes used for 50 million and fewer fragments, the number of slave nodes used increases proportionally with the dataset size, until we reach 60 slave nodes used for 1 billion fragments. Each set of fragments is broken down to partitions of approximately 1 million fragments, the size of which is 128 MB when stored on disk and 700 MB when stored in memory as Java objects. The partitions are stored using Amazon's S3 persistent store, and for `eXpress-D` executions is cached on a slave assigned by the Spark scheduler. To measure how runtimes scale with increasing dataset sizes and cluster resources, we executed `eXpress-D` four times on each cluster for every dataset and report the average of those runs on that cluster. Furthermore, runs over datasets simulated with and without bias were done sequentially on the same cluster. We also used only trials where no Spark processes were interrupted due to disconnected instances, or other machine component failures.

Performance comparisons

Figure 5.1 reveals that `eXpress-D` outperforms all other methods compared for data simulated both with and without bias. This is unsurprising since it combines the exact generative model with the full batch EM algorithm for optimization, while the other methods make various approximations in one or the other. `eXpress` and `Cufflinks` use a complete

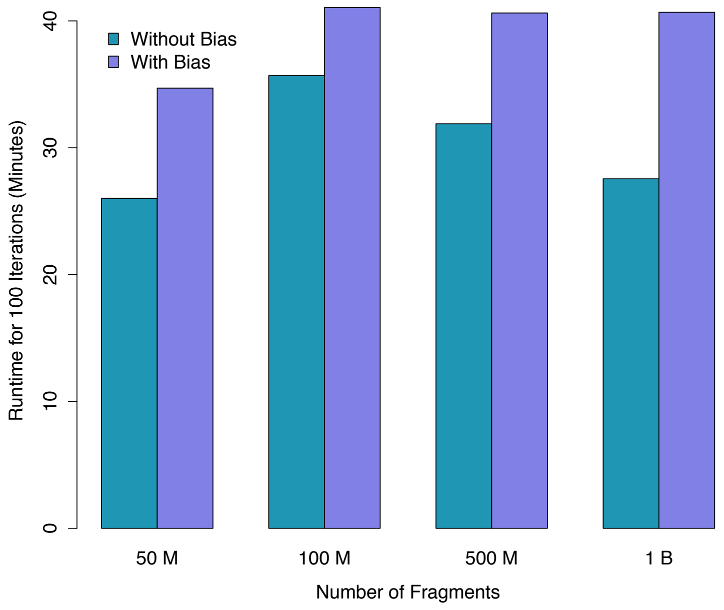


Figure 5.15: ***eXpress-D*** *runtimes.* Average time required for 100 iterations on EC2 for different amounts of input data running on data simulated with (purple) and without (teal) sequence-specific bias. In the latter case, the timing is for iterations after the first 20, which require a constant 30 minutes to learn the bias model. The cluster size is scaled as 3 slave nodes (6 cores) for each 50 million fragments. The results show that `eXpress-D` running on Spark maintains constant runtime when resources are scaled linearly with the amount of the data.

model including bias, but `eXpress` optimizes with the *online* EM algorithm and `Cufflinks` assumes independence between genomic loci. `RSEM` optimizes with the batch EM algorithm but does not model sequence-specific bias. These approximations are made to help the algorithms process large datasets on a single machine, but by taking advantage of the cloud, `eXpress-D` does not need to sacrifice accuracy to scale.

In terms of speed and resource use, Figure 5.15 shows that `eXpress-D` can provide constant runtime if the number of nodes are increased linearly with the size of the input datasets. We found that with one CPU core per 1 million fragments, `eXpress-D` could execute 100 iterations in approximately 30 minutes without bias correction and 40 minutes with bias correction. There is also a constant 30 minute total overhead for learning the bias model during the first 20 iterations. Each run requires approximately 500 iterations to converge, meaning that only 4 hours would be required to process a billion fragments using 60 slave nodes (480 cores). This is just twice is long as is taken by the online EM algorithm of `eXpress`, which can't take advantage of parallelism and has linear scaling in time.

Although it is impossible to directly compare timings across different machines, recall that we found `RSEM` to be unable to complete the processing of more than 200 million fragments on a typical desktop server with 24 GB of RAM or 800 million fragments on a server with 512 GB of RAM (Figures 5.4,5.1), which is more than is available to many labs. Also, we show in Figure 5.5 that `eXpress`, `Cufflinks`, and `RSEM` scaled with slopes that range from 1.8 minutes per million fragments (mpmf) to 27 mpmf on datasets that were successfully processed. Since `eXpress-D` runs in the cloud, it is not limited by the resources on a single machine and can easily scale to a billion reads with essentially no change in the time required.

5.6.4 Discussion

The distributed implementation of `eXpress-D` allows us to combine the full model of `eXpress` with the batch EM algorithm of `RSEM` to provide the best results in the least amount of time for large datasets. A simple extension to `eXpress-D` that also parallelizes the read alignment and pre-processing steps–similar to what is done in `Myrna` and `Crowssbow`[28]–would greatly improve performance and move the full analysis pipeline to the cloud.

As more genomic data moves to the cloud for storage, tools that are able to take advantage of distributed environments and frameworks–such as Spark–will become more widely used and help remove the barriers to large-scale integrative analysis of high-throughput sequencing projects.

Chapter 6

Updating Estimates After Changes to Target Set

The target abundances from high-throughput sequencing data requires a time-intensive step of mapping reads to the target set, followed by an optimization procedure for deconvolution of multi-mapping reads. These procedures are essential for downstream analysis such as differential expression in the case of RNA-Seq, which this chapter will focus on without loss of generality. In cases where it is desirable to adjust the underlying target set, for example upon the discovery of novel isoforms or errors in existing annotations, current pipelines must be rerun from scratch. This makes it difficult to update abundance estimates after re-annotation, or to explore the effect of changes in the transcriptome on analyses.

We present a novel efficient algorithm for updating abundance estimates upon a change in the target set that does not require re-analysis of the entire dataset. Our approach is based on a fast partitioning algorithm for identifying transcripts whose abundances may depend on the added or deleted targets, and on a fast follow-up approach to re-estimating abundances for all targets. We demonstrate the effectiveness of our methods by showing how to synchronize RNA-Seq abundance estimates with the daily RefSeq incremental updates. Thus, we provide a practical approach to maintaining relevant databases of RNA-Seq derived abundance estimates even as annotations are being constantly revised.

Our methods are implemented in software called `ReXpress` and are freely available, together with source code, at http://bio.math.berkeley.edu/ReXpress/.

The research contained in this chapter originally appeared in [54]. While the text focuses on the specific example of RNA-Seq, the method applies for any type of target set.

6.1 Introduction

Two major bottlenecks in RNA-Seq analysis are the mapping of reads to transcripts, which is a prerequisite for quantification and differential analysis, and abundance estimation following mapping. The latter step is particularly complex when multi-mapping reads need to be resolved, which is necessary for estimating isoform-level abundances, or when genes have been duplicated [65]. Popular programs for multi-read assignment, such as `Cufflinks` (Section 5.4, [67]) and `RSEM` (Section 5.2, [31]), have large memory and time requirements (see Figure 5.4). Alternative approaches, such as `eXpress` (Section 5.5, [53]), which uses a streaming algorithm for assignment, are faster with a low memory footprint but must still re-process all the data from scratch when the underlying annotation is adjusted. For large datasets, such as the 3.5 billion reads of [17], a complete run of read mapping with `Bowtie` [29], followed by abundance estimation with `eXpress`, would take approximately 11 days (with 44 cores used for the mapping).

In cases where an annotation of transcripts in a genome may change after mapping, current analysis pipelines require re-mapping of all reads followed by a complete recomputation of abundances [59, 66]. This has made it time-consuming and impractical to determine the effects of the addition of possibly novel transcripts on results or the impact of removal of transcripts that appear to be incorrect. Moreover, in cases of model organisms, it has resulted in the "freezing" of analyses with respect to specific annotation sets, even though re-annotation efforts are resulting in continuous changes to "reference" transcriptomes [47].

The problem we solve in this paper is how to update quantification of transcript abundances in cases where annotations change without remapping all reads to all transcripts and running abundance estimation procedures from scratch. This problem is non-trivial for two reasons:

1. Multi-mapping: Frequently reads map to multiple transcripts, and therefore the removal or addition of transcripts may change the posterior probabilities associated to read mappings. In particular, the addition of a single transcript may require requantification of many other related transcripts.

2. Abundance estimates from RNA-Seq are relative and not absolute: Since RNA-Seq abundance estimates are relative, a change in the abundance estimate of a single transcript affects all other transcripts.

Given a change in the underlying transcripts, we show that abundance estimates can be updated by a procedure that only involves mapping reads to a small subset of the transcripts and by recomputing assignment probabilities of multi-mapping reads for a similarly small set (Figure 6.1). This is made possible by isolating a small relevant subset of transcripts using a partitioning algorithm on a graph constructed from read alignments. When abundance estimation is subsequently performed using a fast online algorithm, the updating of estimates is particularly fast when the change to the underlying annotation is small.

An implication of this result is that it is possible to easily update RNA-Seq abundance estimates for annotations that are continuously updated, as is the case with the nightly Reference Sequence (RefSeq) updates. RefSeq is a large database of sequences that includes widely used reference transcripts for many organisms. RefSeq is updated nightly to reflect improvements in annotations, and although the changes are small, we show that they can affect abundance estimates in RNA-Seq analyses. Our results demonstrate that it is pos-

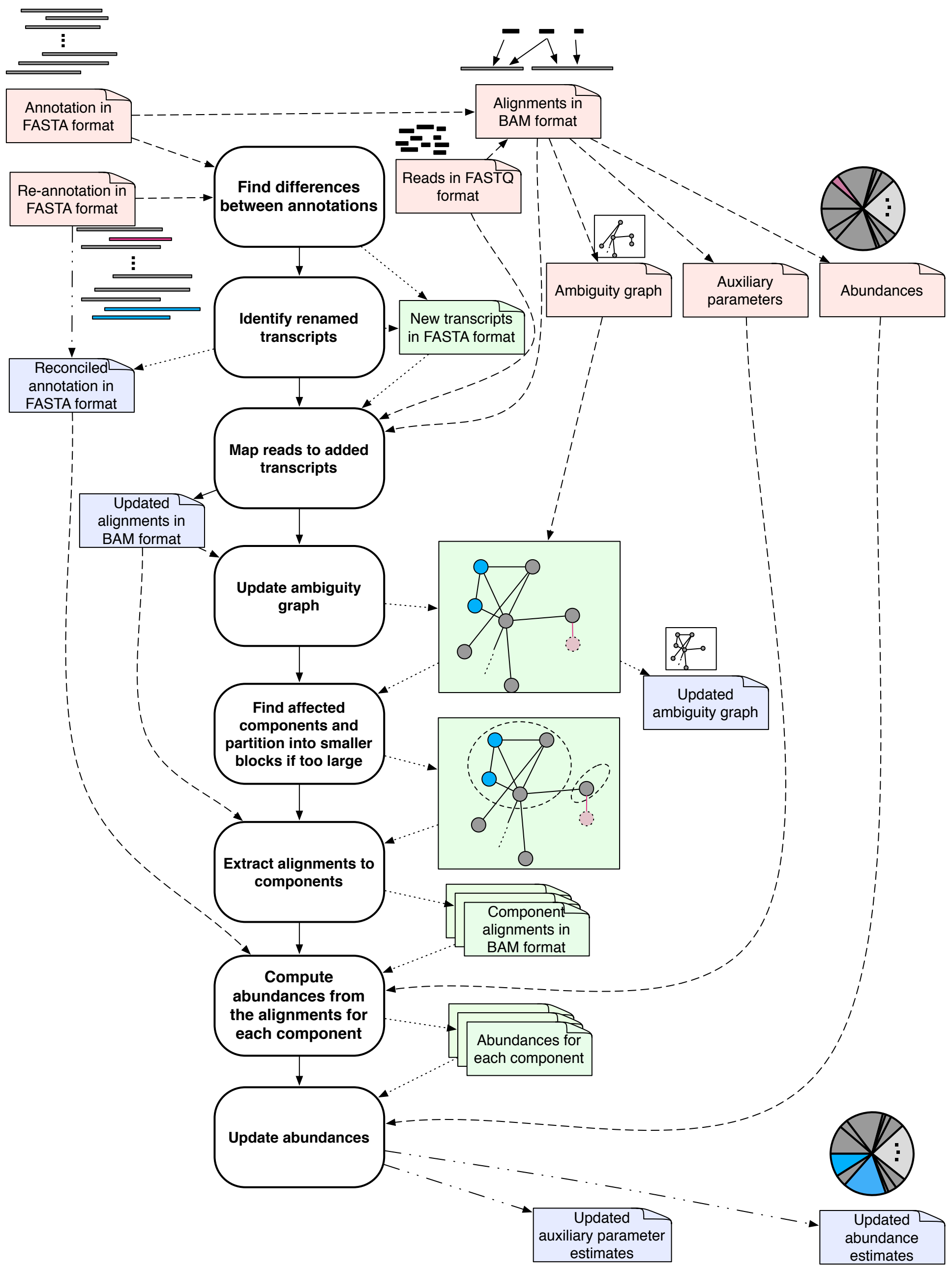

Annotation in FASTA format
Alignments in BAM format
Re-annotation in FASTA format
Find differences between annotations
Reads in FASTQ format
Ambiguity graph
Auxiliary parameters
Abundances
Identify renamed transcripts
New transcripts in FASTA format
Reconciled annotation in FASTA format
Map reads to added transcripts
Updated alignments in BAM format
Update ambiguity graph
Updated ambiguity graph
Find affected components and partition into smaller blocks if too large
Extract alignments to components
Component alignments in BAM format
Compute abundances from the alignments for each component
Abundances for each component
Update abundances
Updated auxiliary parameter estimates
Updated abundance estimates

Figure 6.1: *Overview of* ReXpress*.* Reads are initially aligned to a set of known transcript sequences and these alignments are used to probabilistically assign multi-mapping reads and to estimate abundances of the transcripts. The result is a set of relative abundances, for example in FPKM units. When a new annotation is given, differences are identified. Reads are mapped to any added transcripts and the ambiguity graph, where vertices correspond to transcripts and edges correspond to pairs of transcripts to which reads have mapped ambiguously, is updated (deleted transcripts in red and added transcripts in blue). The "affected" transcripts whose abundance must be recomputed are obtained from a partitioning in the graph. Finally, the subset of affected transcripts have their abundances recomputed using the relevant reads and abundances for the transcriptome are recomputed.

sible, with our algorithm, to analyze an RNA-Seq dataset by building up the annotation one transcript at a time. In particular, our tool `ReXpress` allows scientists to routinely update abundance estimates for RNA-Seq analyses to reflect best possible results at any time. Although `ReXpress` is designed to work with formats produced by the `eXpress` RNA-Seq quantification tool, the program is general and suitable for use with many mapping and abundance estimation methods.

6.2 Methods

6.2.1 Incremental adjustment of abundance estimates

We begin by describing the adjustments that are required to update an RNA-Seq analysis with respect to a re-annotation. An outline of the algorithm is provided in Figure 6.1. We will assume that there already exists an initial annotation, alignments, abundances, and ambiguity graph (see Section 5.3). In our software implementation, we assume that the files are in `eXpress` format [53], but `eXpress` itself does not have to be used to generate the output.

Given an updated FASTA file containing a re-annotation, the newly added and deleted transcripts are detected. In some cases, re-annotation can involve simply renaming existing transcripts, and this case is checked for and ignored if detected (after names/identifiers are correctly updated). A modified transcript can always be described in terms of a transcript deletion followed by an addition. For each difference between annotations the nature of the edit is recorded.

The set of reads are then aligned to the added transcripts, and the ambiguity graph is updated with new nodes representing these transcripts and edges induced by the new alignments. The transcripts and alignments associated with independent components of the ambiguity graph containing added and deleted transcripts are extracted. The abundances for all transcripts in these components are then re-quantified separately using the updated annotation. Finally, the new annotation, alignments, abundances (for all transcripts), and ambiguity graph are output to be used with the next re-annotation update. Below is a more formal description that explains the steps in detail. Proofs of correctness follow trivially from the factorization of the standard likelihood function used in RNA-Seq, and are omitted.

6.2.2 Mathematical details

We require two fields from the output of an RNA-Seq quantification program after it has been used to estimate abundances for a set of transcripts $\mathcal{T}$: the estimates $\hat{\rho}^{\mathcal{T}}$ and the *ambiguity graph* (defined in Section 5.3) of $\mathcal{T}$ which we denote by $G = (\mathcal{T}, E)$. We assume that $\mathcal{T}'$ consists of $\mathcal{T}$ with the addition of a set of transcripts $\mathcal{A}$ and the deletion of a set of transcripts $\mathcal{D}$ so that $\mathcal{T}' = (\mathcal{T} \cup \mathcal{A}) \setminus \mathcal{D}$. Finally, we will need the stored alignments from $\mathcal{F}$ to $\mathcal{T}$, which we denote by $L_{\mathcal{F}\to\mathcal{T}} = \{f \to \{t | t \in \mathcal{T} \text{ and } f \text{ aligns to } t\}\}$.

To simplify the presentation we explain separately the case of adding transcripts ($\mathcal{T}' = \mathcal{T} \cup A$) and the case of deletion ($\mathcal{T}' = \mathcal{T} \setminus \mathcal{D}$). The general case of addition and deletion of transcripts can be handled by following addition by deletion, or together by combining the procedures (details omitted). Furthermore, for simplicity, in the description below we restrict the exposition to the case of addition/deletion of a single transcript.

- Given a set of transcripts $\mathcal{T}$, let t' be a transcript with $t' \notin \mathcal{T}$. The updating of estimates when t' is added to the annotation is performed as follows:

  1. Align the reads in $\mathcal{F}$ to t' and denote the subset of reads of $\mathcal{F}$ that align to t' by $\mathcal{F}' \subseteq \mathcal{F}$. Denote the alignments of $\mathcal{F}'$ as $L_{\mathcal{F} \to t'}$.
  2. Extract the read alignments for the reads in $\mathcal{F}'$ from $L_{\mathcal{F} \to \mathcal{T}}$ and denote as $L_{\mathcal{F}' \to T} = \{f \to L_{\mathcal{F} \to \mathcal{T}}(f) \text{ for all } f \in \mathcal{F}'\}$. In addition, denote by $\mathcal{S} = \bigcup_{f \in \mathcal{F}'} L_{\mathcal{F} \to \mathcal{T}}(f)$ the set of transcripts in $\mathcal{T}$ that appear in $L_{\mathcal{F}' \to T}$.
  3. Create the updated ambiguity graph $G' = (\mathcal{T} \cup t', E \cup \{\{t', v\} \text{ for all } v \in \mathcal{S}\})$.
  4. Let $\mathcal{B} = \{t : f \text{ is in the same component as } t', t \neq t'\}$. Extract the alignments in $L_{\mathcal{F} \to \mathcal{T}}$ that consist of a read mapping to a transcript in $\mathcal{B}$ as $L_{\mathcal{F} \to \mathcal{B}} = \{f \to L_{\mathcal{F} \to \mathcal{T}}(f) \text{ for all } f \in \mathcal{F} | L_{\mathcal{F} \to \mathcal{T}}(f) \subseteq \mathcal{B}\}$.
  5. Merge the alignments to create $L_{\mathcal{F} \to \mathcal{B} \cup t'} = L_{\mathcal{F} \to \mathcal{B}} \cup L_{\mathcal{F} \to t'}$.
  6. Perform quantification on the set of transcripts $\mathcal{B} \cup t'$ using the alignments $L_{\mathcal{F} \to \mathcal{B} \cup t'}$. This produces a set of estimates $\{\hat{\rho}'_t\}_{t \in \mathcal{B} \cup t'}$.
  7. Compute $\hat{\rho}^{\mathcal{T}}_{\mathcal{B}} = \sum_{t \in \mathcal{B}} \hat{\rho}^{\mathcal{T}}_t$. Set $\hat{\rho}^{\mathcal{T}'}_t = \hat{\rho}^{\mathcal{T}}_{\mathcal{B}} \times \hat{\rho}'_t$ for all $t \in \mathcal{B} \cup t'$.

- Deletion is performed via a similar procedure. Let t' be a transcript with $t' \in \mathcal{T}$.

  1. Let $\mathcal{B}$ be the component in G that contains t'.
  2. Extract the alignments from $L_{\mathcal{F} \to \mathcal{T}}$ that contain reads mapping to transcripts in $\mathcal{B}$, denoted by $L_{\mathcal{F} \to \mathcal{B}} = \{f \to L_{\mathcal{F} \to \mathcal{T}}(f) \text{ for all } f \in \mathcal{F} | L_{\mathcal{F} \to \mathcal{T}}(f) \subseteq \mathcal{B}\}$.
  3. Remove the alignments of reads to t' from $L_{\mathcal{F} \to \mathcal{B}}$ as $L_{\mathcal{F} \to \mathcal{B} \setminus t'} = \{f \to L_{\mathcal{F} \to \mathcal{B}}(f) \setminus t' \text{ for all } f \in \mathcal{F}\}$.
  4. Perform quantification on the set of transcripts $\mathcal{B} \setminus t'$ using the alignment file $L_{\mathcal{F} \to \mathcal{B} \setminus t'}$. This produces a set of estimates $\{\hat{\rho}'_t\}_{t \in \mathcal{B} \setminus t'}$.
  5. Compute $\hat{\rho}^{\mathcal{T}}_{\mathcal{B}} = \sum_{t \in B} \hat{\rho}^{\mathcal{T}}_t$. Set $\hat{\rho}^{\mathcal{T}'}_t = \hat{\rho}^{\mathcal{T}}_{B} \times \hat{\rho}'_t$ for all $t \in \mathcal{B} \setminus t'$.
  6. Create the updated ambiguity graph $G' = (\mathcal{T} \setminus t', E \setminus \{\{t', v\} \text{ for all } v \in \mathcal{B}\})$

Note that in the rare case where this a change in the total number of aligned fragments after the addition or deletion of a target, and additional step is required to renormalize the relative abundance between components. This step is trivial and fast, and the details are omitted.

6.2.3 Improving performance by approximating the affected set

There is another issue that can hurt performance in practice: the affected component $\mathcal{B}$ can be very large (see Figure 5.7). In typical RNA-Seq experiments, as much as one fifth of all transcripts can lie in a single component of the ambiguity graph [53]. These components typically consists of large gene families and multiple isoform genes that share common sequence. To improve performance, it is therefore desirable to restrict the re-quantification to a smaller subset without sacrificing important information in the form of fragment alignments. We do this by partitioning a weighted generalization of the ambiguity graph, obtained by the addition of edge weights representing the number of ambiguous alignments between each pair of transcripts. For a given mapping $L_{\mathcal{F}\to\mathcal{T}}$ and induced ambiguity graph G, we let the weight between two transcripts u, v be $w(\{u,v\}) = \sum_{f\mathcal{F}} \mathbf{1}(\{u,v\} \subseteq L_{\mathcal{F}\to\mathcal{T}}(f))$. Given these weights, we wish to partition around t' such that the total weight of edges crossing the partition cut is small compared to the weight of edges inside the block. Moreover, it is desirable that the block containing t' is small.

Many sophisticated objective functions and algorithms exist for partitioning graphs [5]. A detailed exploration of the applications of these methods to our problem is outside the scope of this paper. Instead, to demonstrate the feasibility of a partitioning scheme for improving the performance of our method with large components, we chose to use the greedy approach outlined below, which is motivated by the objective of removing edges that correspond to the "least informative" alignments.

First, we define the density of a block S, $d(S)$, as the total weight of edges incident to a node in the block and a node outside of the block divided by the total weight of edges incident to the nodes in the block. Intuitively, this is the ratio of edges crossing the cut to all of the edges touching nodes in the block. Formally,

$$d(S) = \frac{\sum_{u\in S, j\in\bar{S}} w(u,v)}{\sum_{u\in S, v\in\mathcal{T}\setminus S} w(u,v) + \sum_{(u,v)\in S} w(u,v)}. \tag{6.1}$$

Our objective is to find, for a given transcript t', a block S that contains t such that $d(S) < \theta$ for a given threshold $0 < \theta \leq 1$. We do so using the following greedy update.

1. Begin with $S = \{t'\}$.

2. Iteratively add node $u = \text{argmax}_{u\in\mathcal{T}} w(\{t', u\})$ to S until $d(S) < \theta$.

It is easy to show that for any valid θ, this algorithm will terminate. As we show in the following section, the method is empirically both fast and accurate.

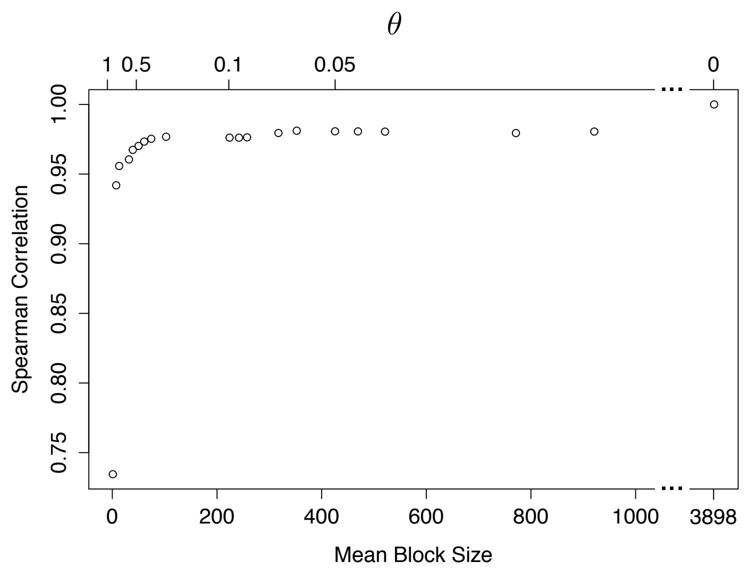


Figure 6.2: *Tradeoff between approximate affected component size and accuracy for the greedy partitioning algorithm.* 250 transcripts were randomly selected with replacement from the largest component in Figure 5.7. For 20 values of θ, each transcript was removed from the dataset and subsequently re-added using `ReXpress`. The abundance for the transcript computed by `ReXpress` was then correlated (using Spearman rank correlation) with the initial results for those transcripts using the full dataset. While smaller partitions result in a reduction of this correlation, the computational time is greatly reduced. For $\theta < 0.1$, the affected component size is reduced by nearly a factor of 20, while the accuracy is still within the expected error of the `eXpress` method from Section 5.5.

6.3 Results

6.3.1 Accuracy of partitioning approximation

To validate the performance of our greedy partitioning algorithm, we randomly selected with replacement 250 transcripts from the largest partition (3,898 transcripts) in the initial RefSeq time-point and simulated their addition at some earlier date. Each selected transcript was filtered from the FASTA, alignment file, and ambiguity graph, and was then re-added using a single step of our algorithm. We show the results of this update using 20 different values of θ in Figure 6.2. There is a clear tradeoff between the accuracy of the approximation and the size of the resulting block selected by different values of θ. We note that for $\theta < 0.1$, the correlation is already reasonably close to the accuracy of the `eXpress` algorithm demonstrated in [53] and Chapter 5.

6.3.2 Application to RefSeq incremental update

To demonstrate the effectiveness of our approach, we applied it to the large RNA-Seq dataset produced for [67], which consisted of RNA-Seq performed on C2C12 which is a mouse myoblast cell line. The data was first analyzed in 2010, but since then the mouse RefSeq annotation has been updated numerous times. Specifically, as a proof-of-concept, we applied `ReXpress` (our implementation of the methods above) to 34 days of the RefSeq incremental update (RIU), which is a daily update of the RefSeq annotation database (see Methods, Figure 6.3A).

Using 24 free cores, 644 minutes were required for the initial `Bowtie2` alignment, 505 minutes for 20 repetitions of abundance estimation with `eXpress`, and 11 minutes for building the ambiguity graph. Across the entire month of RefSeq updates, a size 3910 component was affected seven times, while components of size 15 or less were affected 37 times.

Each subsequent update required, on average, 55 minutes to complete our re-annotation pipeline. This is compared with the approximately $644 + 505 = 1149$ minutes that would be required for alignment and abundance estimation from scratch with `Bowtie2` and `eXpress` after each re-annotation.

The abundance estimates for the final time point had a Spearman rank correlation of $R^2 = 0.994$ with those calculated from scratch. The small discrepancy is due to the fact that the online EM method in `eXpress` approximates the maximum likelihood solution, and therefore is not expected to be exact.

Because some of the transcripts added and deleted over the time period affected the large components in the ambiguity graph, we also ran the analysis using the greedy partitioning scheme described above ($\theta = 0.1$). While the speed of the updates was greatly improved by the partitioning by reducing the size of the (approximate) affected components (Figure 6.3B), the results were nearly identical.

6.4 Discussion

Despite the difficulties in storing, processing and distribution of high-throughput sequence data [58], repositories such as GEO have led to an explosion in publicly available genome-wide expression data. However, numerous technical challenges that arise in re-using data have limited the utility of publicly archived RNA-Seq reads [57].

Our results show that it is possible to efficiently update RNA-Seq abundance estimates upon re-annotation, thus removing a major obstacle to re-using publicly available data. This should prove to be particularly useful in newly sequenced organisms whose annotations are not stable and undergo periodic revision, and also in human cancer transcriptomics where structural alterations can be tumor specific [3, 72]. We also believe that `ReXpress` will be particularly useful for sequencing centers providing analysis services. Instead of producing one-time output, it should now be possible to refresh analyses as annotations improve, without expensive hardware or compute time needed as user bases and datasets

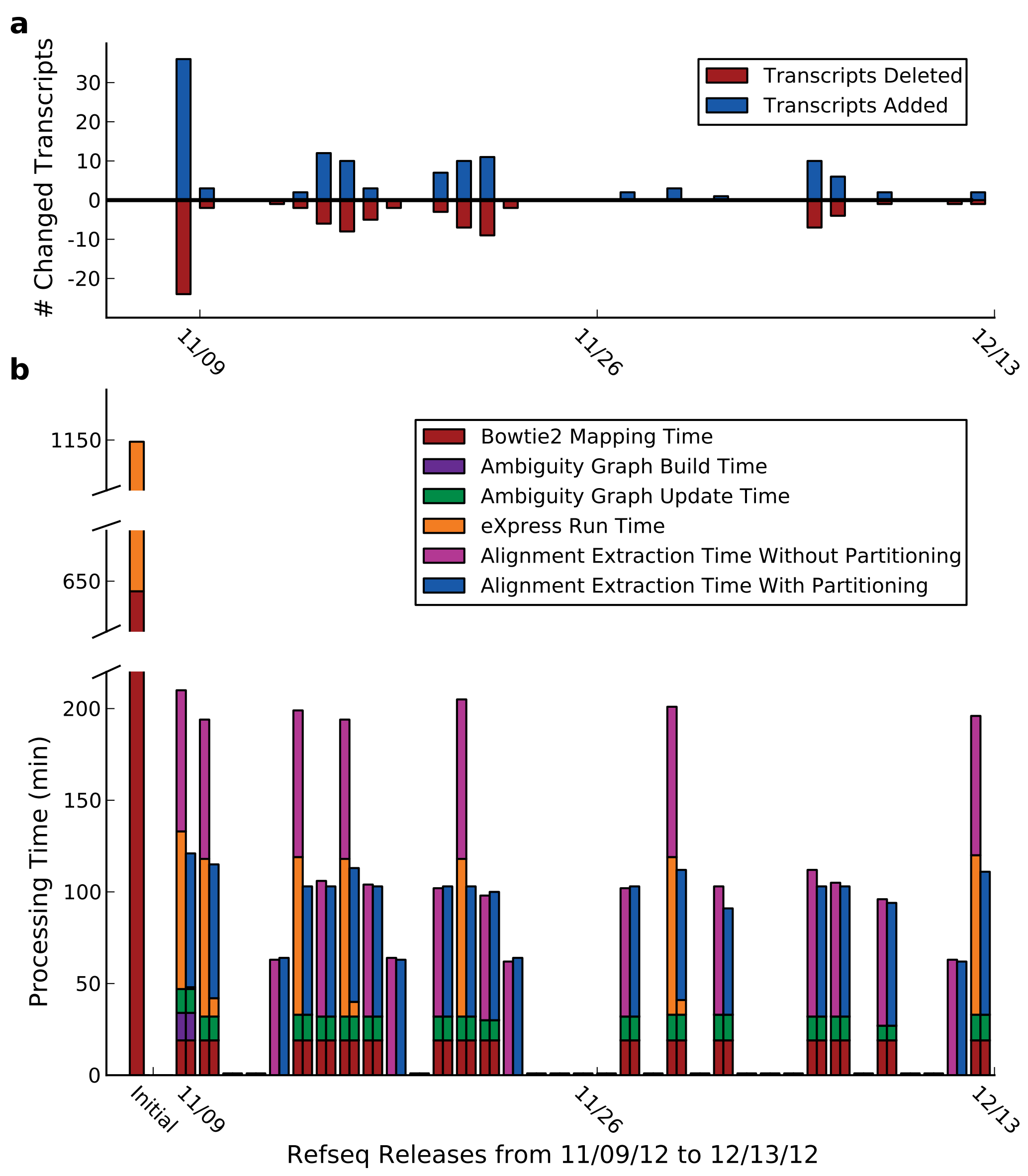


Figure 6.3: *Updates to the mouse RefSeq transcriptome over the course of 34 days.* (A) Numbers of transcripts added and deleted over the 34-day period. Transcripts that kept the same name but changed sequence were treated as an addition and a deletion. (B) `ReXpress` run time, in minutes, on each RefSeq update, with and without partitioning. Initial run time consists of `Bowtie2` alignment time (24 cores) and `eXpress` abundance estimation time (3 cores), without `ReXpress`. Partitioning was done when a changed transcript was part of a component larger than 300 transcripts, which occurred 7 times over the 34-day period.

grow.

Other applications of our work include a randomized approach to optimization of transcriptome assembly in conjunction with abundance estimation [36, 37, 43], and the development of an RNA-Seq quantification database for publicly available datasets that is automatically updated as annotations improve.

Moreover, our work on component identification in and partitioning of the ambiguity graph can be used to develop more efficient batch methods for abundance estimation. A common issue in the commonly used batch EM solutions [67, 31] is the necessity of iterating over a large number of reads, which has a memory bottleneck as shown in Chapter 5. As mentioned in Section 5.3, a better solution for the memory bottleneck in the batch method is to iterate over approximately independent partitions of the ambiguity graph whose associated reads can be fit into memory. Since most components are often small, only the largest will need to be partitioned as in our method above. The blocks can then be processed in parallel only a single machine or distributed over a cluster.

Finally, in conjunction with the streaming algorithm for quantification in [53], the present method provides an online algorithm in both the reads and the targets in any setting where probabilistic assignment of multi-mapping reads is a bottleneck in analysis of high-throughput sequencing data.

Chapter 7

Detecting RNA-DNA Differences

RNA-DNA differences (RDDs) are known to be produced by the enzymatic "editing" of RNA after transcription, which serves a clear biological function in the modification of proteins produced by specific genes. Recent studies have also discovered, to varying degrees, non-canonical forms of RDD with unclear function or mechanism. The explosion of DNA- and RNA-Seq data has led researchers to query new datasets for the discovery and validation of RDDs, but a rigorous statistical analysis has yet to be introduced. Current methods rely on heuristics and ad-hoc thresholds, and the discrepancies in their results have caused major disagreements in the interpretation of the same underlying data as to the number and types of edits that actually occur [26, 39, 52, 49]. Here we lay the groundwork for a more rigorous statistical method through the extension of the *eXpress model* described in Chapters 2,4 and using the online EM algorithm of `eXpress` (Section 5.5 and [53]) that we hope can put an end to the current chaos in the field.

7.1 Background

With the success of recently published tools for the detection of genomic variation in the large-scale sequencing efforts such as the 1000 Genomes project, researchers have begun to turn their attention to a less explored form of variation: RNA-DNA differences (RDDs). An RDD is defined as a site in RNA that differs with the DNA from which it was transcribed. A common form of this variation is due to RNA editing (Figure 7.1), a post-transcriptional process whereby the RNA sequence is modified. Typically, this process is performed by ADAR enzymes, which convert adenosine to inosine (translated as guanosine by the ribosome), known as A$\rightarrow$I(G) editing. Another, more rare form of editing is catalyzed by APOBECs, which edit cytidine to uridine (C$\rightarrow$U) in a similar manner. Furthermore, recent studies making use of RNA-Seq and applying methods similar to those used for genome variant discovery have provided convincing evidence for other non-canonical forms of editing, especially transitions [35, 51].

That non-canonical editing occurs is clear. However, there is a large controversy sur-

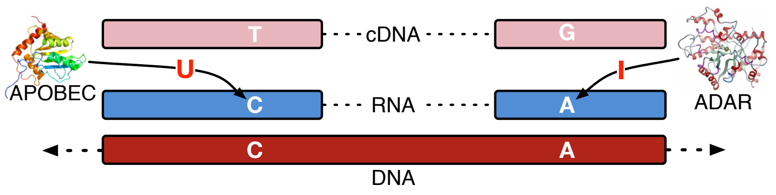


Figure 7.1: *Canonical RNA editing.* RNA editing leads to RNA-DNA Differences (RDDs) and is known to occur in two canonical ways. In C→U editing, a cytosine base is deaminated into a uridine. In A→I editing, an adenosine base is deaminated into an inosine by an ADAR enzyme and is treated as a guanine in translation.

rounding how widespread these RDDs are in practice. According to one study [35], non-canonical editing is nearly as common as canonical editing. Nevertheless, multiple other analyses of the same data [26, 39, 52] as well as newer datasets [51] have concluded otherwise. Thus, variable approaches to the analysis of these data relying on heuristics and ad-hoc thresholds are producing widely varying results, leading to the conclusion that a more principled method, based on a statistically-sound generative model, is needed [49].

In the following sections, we indicate potential faults in previous analyses that may lead to false positives. We then describe our proposed generative model and likehood-based method for a high-throughput sequencing experiment meant to detect RDDs, which we have implemented as an extension to `eXpress` with only minor engineering challenges. Finally, we will present evidence that our method can accurately identify RDDs.

7.2 Previous Methods

We focus on the two largest studies to date using RNA-Seq for detecting RDDs, one by Li *et al.*[35] and one by Peng *at al.*[51].

7.2.1 Li *et al.* [35]

In their paper, Li *et al.* sequenced the transcriptomes of 27 individuals and compared with their genome data from the 1000 Genomes Project to find RDDs. After analyzing this data, they announced the discovery of 28,766 RDD events, over 10,000 of which were non-canonical. This finding was disputed by other researchers in three separate commentaries [26, 39, 52] for four primary reasons: uneven distribution of events among read positions, strand specificity, calls at known polymorphic sites, and the existence of paralogous genes with sequences matching the proposed edit. The first two are signs of systematic errors in the sequencing and library preparation [41], and the latter two are due to a combination of incomplete or incorrect reference sequences and mapping errors.

Systematic sequencing errors

Uneven distribution of claimed RDD events within reads are demonstrated in all three commentaries. These likely false positives land near the ends of reads, especially on the 5' end of the fragment. Several explanatory hypotheses are proposed including the possibility of systematic sequencing errors [52], overhanging ends of mis-aligned spliced reads [26], and binding of non-exact random hexamer primers [26], which best explains the 5' bias. Two of the commentaries [26, 52] also point out that a large number of the proposed events are supported by reads in only one direction. This has been found to be a strong indicator of a systematic error in Illumina sequencers [41], and is a likely explanation for some of the unexpected findings.

Reference and mapping errors

Many of the alleged false positives in [35] were attributed to errors in the reference as well as mapping errors that were not accounted for.

In the first case, [26, 39] point out that many of the sites were at positions with the reference nucleotide was determined using very low coverage DNA-Seq (as low as four reads). Therefore, a miscalled reference base would appear to be an RDD when compared to deeper coverage RNA-Seq data. Furthermore, [52] adds that many of the RDD sites are known to be polymorphic in the human population, which, combined with the low-coverage genotyping calls explains many of the alleged false positives.

All three commentaries [26, 39, 52] also argue that limiting alignments to an incomplete transcriptome reference and only considering unique alignments below some threshold led to further false positives. By remapping the reads to the full genome, the commentators demonstrated that many multiply align to duplicated genes, some of which are not annotated in the transcriptome. Without knowledge of these multiple mappings, the original authors substituted inexact false alignments for the true origin of the reads, thus believing the mismatches to be RDDs. Furthermore, reads covering non-annotated splice junctions will commonly align to other isoforms of the gene with errors in the last few bases where a different exon is spliced. [26, 39] found that this artifact appeared repeatedly at sites reported as RDDs by [35]. This issue is further exacerbated by the fact that the genome references themselves are incomplete so a full transcriptome reference may only be possible using a *de novo* assembly of the RNA-Seq data itself.

7.2.2 Peng *et al.* [51]

In their paper, Peng *et al.* deeply sequence the transcriptome of a Chinese individual whose genome had previously been sequenced [69]. Aware of the controversy surrounding [35], the authors set out to build a series of filters to remove likely false positives and avoid the pitfalls of the previous publication. The pipeline focused on removing likely PCR duplicates, low coverage and repetitive sites, sites with SNVs, sites likely to receive erroneous mappings issues, strand biased sites, sites with similar sequence repeated in the genome, sites of known variation from other sources (SNPs), sites with multiple types of inferred edits, and sites with 100% of the reads differing between the DNA- and RNA-Seq.

The results in this paper were shown to be much more in line with the expected results based on what is known about RNA editing. Only a small number of non-canonical sites were called and many of the canonical sites matched known sites from previous studies. Furthermore, the authors reanalyzed the data from [35], producing results that also more closely matched the expectations of [26, 39, 52].

The apparent success of this method is noteworthy. Nevertheless, the heuristic nature of the approach makes uncertainty difficult to measure and leads to the possibility of introducing unintended biases into the analysis.

7.3 Method

We have developed a principled likelihood-based approach for RDD detection using a generative model, which addresses the problems in the previous methods, especially those of [35].

7.3.1 Generative model

The generative model used is an extension of the *eXpress model* of Chapters 2,4 and [53] and is shown in Figure 7.2. The observed variables are a set of RNA-Seq fragments, a set of DNA-Seq fragments, and a reference genome. The individual genome is generated from a set of reference genomes and then produces haplotype transcripts that may or may not be edited from the genome sequence. The two sequencing experiments proceed independently, beginning by sampling a fragment length followed by the selection of a transcript given the sampled length, and bias and abundance parameters in the case of RNA-Seq, or only given the length and bias in the case of DNA-Seq, as abundances are assumed to be uniform. Next a position is selected in the transcript given the sampled length and transcript, and bias parameters specific to the experiment. Finally errors are inserted based on the experiment's error distribution. Below, more detail is provided on how we deal with the issues pointed out in the previous section.

Systematic sequencing errors

As in `eXpress` (Section 5.5 and [53]), this model uses a Markov chain to model the likelihood of specific base substitutions given position in the read and sequence context. The model can easily be learned with a relatively small amount of data and should be able to help discriminate between the likelihood of a base difference being due to sequencing or priming errors as opposed to a real RDD. The ends of the fragments are modeled separately based on sequence order, since the sequencing device is biased towards different errors on each read.

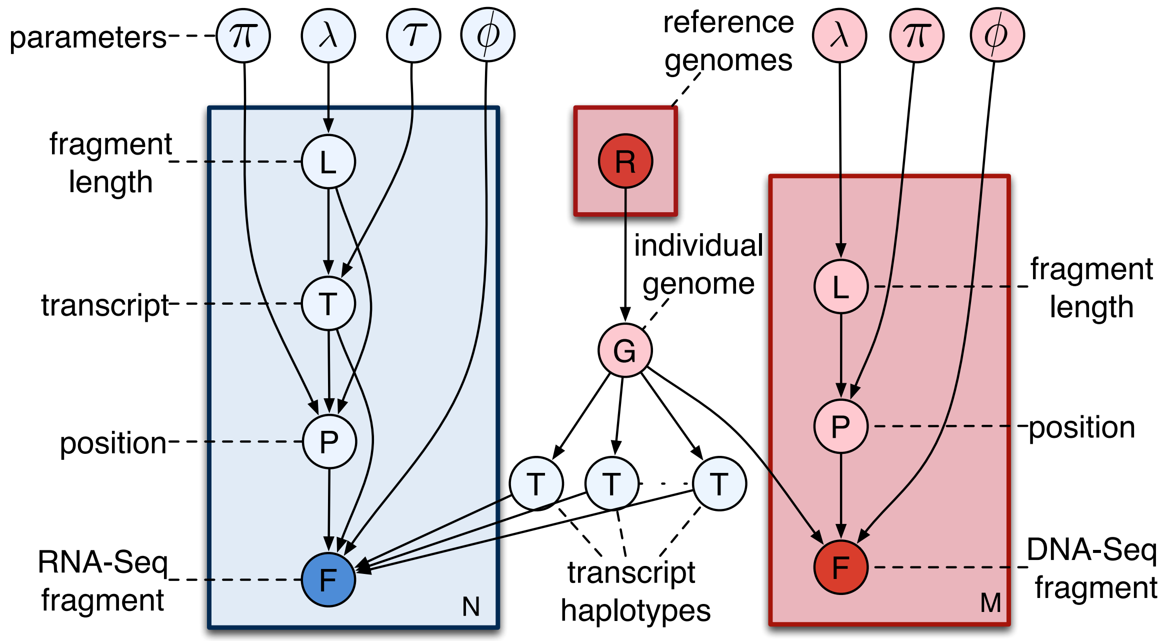


Figure 7.2: *Graphical representation of generative model for RDD detection experiment.* Here we assume a stochastic process generates an individual's diploid genome from a set of reference genomes, thereby producing a set of transcript haplotypes, possibly including edits. During a DNA-Seq experiment (red), sequenced fragments are generated from the genome based on distributions for fragment length, sequencing biases, and sequencer errors. During an RNA-Seq experiment (blue), sequenced fragments are generated from the transcript haplotypes based on distributions for fragment length, transcript abundances, sequencing bias, and sequencer errors. Our method attempts to find the set of parameters that maximize the likelihood given this model. Observed random variables are shown in bold.

However, it may also be useful to further subdivide the reads based on whether they are the 5' or 3' end of the fragment in order to capture errors due to the priming of inexact sequences [26], although this is not yet implemented.

Based on the findings of [41] that different upstream motifs are features of systematic error, the uniform strandedness at an error locus appears to be due to the different upstream sequences in reads from reverse complemented fragments. In this case, the context-dependent model described above should be able to properly assign a correct error likelihood without explicitly taking read directionality into account. Nevertheless, it may be necessary to model other features of systematic error, for example by including base qualities in the error model.

Reference and mapping errors

By allowing for the possibility that any transcript could potentially produce any observed fragment, our model can avoid the types of mapping errors described above. Of course, we do not wish to consider every possible alignment, but a very liberal mapping procedure combined with comparing multiple alignments with different mapping scores will make us less susceptible to error.

Furthermore, as the genome itself is treated as a latent variable that is being inferred from both the DNA-Seq and RNA-Seq data, we are much less sensitive to the effect of inaccurate reference sequences. Our method learns, albeit with some uncertainty, the correct base at each position, taking into account sequencing errors and biases. By requiring all haplotypes of an individual as inputs, we further reduce the number of false positives at multi-allelic sites.

7.3.2 Likelihood function

Since our method (below) allows us to decouple the two sequencing experiments, we begin by focusing on the the likelihood function from Section 2.5 for the RNA-Seq experiment,

$$L(\lambda, \rho, \pi, \phi | \mathcal{F}) = \prod_{f \in \mathcal{F}} \sum_{l=1}^{M_L} \sum_{t \in \mathcal{T}} \sum_{p=1}^{l(t)-l+1} \lambda_l \cdot \tau_{t|l} \cdot \pi_{p|t,l} \cdot \phi_{f|t,p,l}, \tag{7.1}$$

and expand it in the following manner. First, note that in `eXpress`, ϕ is made up of a first order Markov chain for base substitution probabilities due to sequencing errors as well as an independent model for insertions and deletions. For simplicity, we assume that the post-transcriptional modification and sequencing do not result in indels. We represent the substitution probabilities for the 5' read as

$$\epsilon^{5,i}_{a|b,c} := \mathbb{P}(f^5_i = a | \tilde{f}^5_{i-1} = b, \tilde{f}^5_i = c), \tag{7.2}$$

and similarly $\epsilon^{3,i}_{a|b,c}$ for the 3' end, where $\tilde{f}$ represents the true sequence of the fragment (no sequencing or priming errors), the superscript denotes an end of paired-end read, and the subscript is an index into the sequence. Furthermore, we define

$$\sigma^{t,i}_b := \mathbb{P}(t_i = b), \tag{7.3}$$

where t_i is the base at index i in the sequence of target t. Then, letting $\mathcal{B} = \{\text{'A', 'C', 'G', 'T'}\}$ be the set of possible bases and assuming independence between the ends,

$$\phi_{f|t,p,l} = \phi_{f^5|t,p,l} \cdot \phi_{f^3|t,p,l} \tag{7.4}$$

$$\phi_{f^5|t,p,l} = \prod_{i=1}^{l(f^5)} \sum_{a \in \mathcal{B}} \sigma^{t,p+i-1}_b \sum_{b \in \mathcal{B}} \sigma^{t,p+i}_b \cdot \epsilon^{5,i}_{f^5_i|a,b}, \tag{7.5}$$

with a similar equation for $\phi_{f^3|t,p,l}$.

We modify the calculations of the bias weights for each potential fragment alignment $w_{t,p,l}$, and the effective length of a transcript t, $l(t)$, from Section 4.2 in a similar way so that the expected distribution, for example, is computed taking base uncertainty into account.

The likelihood for the DNA-Seq data is nearly identical, except the transcript abundances (ρ) used to compute the sampling probabilities (τ) are fixed to be uniform, i.e. $\rho_t = |\mathcal{T}|^{-1}$ for all $t \in \mathcal{T}$.

7.3.3 Optimization of likelihood

Online updates

The likelihood is optimized in the same manner as `eXpress`, using the online EM algorithm from Section 5.5. The only required modifications are in how to learn and update the parameters mentioned in the previous section (ϵ, σ, π). The parameters are represented by Dirichelet random variables and updated by the multinomial assignments of fragment alignments, as ϵ and π are estimated in `eXpress`. However, due to the fact that the target sequences are now latent variables, the updates to ϵ become

$$\epsilon^{5,i}_{a|b,c} \leftarrow (1-\gamma_n)\epsilon^{5,i}_{a|b,c} + \gamma_n \cdot \mathbb{1}(f^5_i = a) \sum_{(t,p,l)\in\hat{\mathcal{A}}_f} \hat{P}_f(t,p,l) \cdot \sigma^{t,p+i-1}_b \cdot \sigma^{t,p+i}_c. \tag{7.6}$$

The update for π is similar, but is for a third order Markov chain.

If we let

$$\hat{\mathcal{A}}_f(t,i) := \{(r,p,l) \in \hat{\mathcal{A}}_f : t = r \wedge ((p < i < p+l(f^5)-1) \vee (p+l-l(f^3)-1 < i < p+l-1))\} \tag{7.7}$$

be the set of alignments of fragment f that cover position i of transcript t, and let $n_{t,i}$ be the number of fragments thus far in the experiment that cover position i. Then the update for $\sigma^{t,i}_b$ is

$$\sigma^{t,i}_b \leftarrow \left(1 - \frac{2 - \sum_{(t,p,l)\in\hat{\mathcal{A}}_f(t,i)} \hat{P}_f(t,p,l)}{n_{t,i}}\right)\sigma^{t,i}_b + \frac{1}{n_{t,i}} \sum_{(t,p,l)\in\hat{\mathcal{A}}_f(t,i)} \hat{P}_f(t,p,l)$$
$$\cdot \frac{\sum_{j=1}^{l(f^5)} \mathbb{1}(i = p+j-1)\mathbb{P}(\tilde{f}^5_j = b|f^5_j,(t,p,l)) + \sum_{j=1}^{l(f^3)} \mathbb{1}(i = p+l-j)\mathbb{P}(\tilde{f}^3_j = b|f^3_j,(t,p,l))}{\max(1, \sum_{j=1}^{l(f^5)} \mathbb{1}(i = p+j-1) + \sum_{j=1}^{l(f^3)} \mathbb{1}(i = p+l-j))}, \tag{7.8}$$

where

$$\mathbb{P}(\tilde{f}^5_j = c|f^5_j = a, f = (t,p,l)) = \frac{\sum_{b\in\mathcal{B}} \sigma^{t,p+j-2}_b \sigma^{t,p+j-1}_c \epsilon^{5,j}_{a|b,c}}{\sum_{d\in\mathcal{B}}\sum_{b\in\mathcal{B}} \sigma^{t,p+j-2}_b \sigma^{t,p+j-1}_d \epsilon^{5,j}_{a|b,d}}. \tag{7.9}$$

Note that in place of γ_n, we are using $1/n_{t,i}$ as the update mass. This is due to the sparsity of the informative fragments for any given σ parameter. The first term of 7.8 provides a correct weighting of the current value based on this update schedule. In the second term, we sum over each position in both ends of all fragments that overlap with position i, accumulating the posterior probabilities that any observed bases at position i were produced by there being a b nucleotide in the actual fragment sequence. We use indicator functions here instead of indexing directly into the fragment sequences since, although the fragments are known to overlap i, it is not guaranteed that the base was included in the sequenced reads. The denominator in the second term is so that we don't double count the evidence a single fragment provides if both ends cover position i.

Priors

Since we are using the online EM algorithm with vary sparse updates for the sequence parameters, the prior used in this case is very important. A uniform prior would cause all positions to look equally likely in terms of the assignment and would ignore the information provided in the reference sequence. A deterministic prior based on the reference would cause the distribution to be permanently fixed, reducing our model to that in `eXpress`.

Instead, we bias our prior towards the reference nucleotide and give non-zero, uniform probability to the other bases. How close the reference base is to 1, along with how strong the prior is (in terms of the Dirichlet parameters) will determine how easily the distribution is allowed to shift given observed alignments. Because we are taking possible sequencing errors and alternative alignments into account, it is reasonable to assume that a weaker prior is necessary and sufficient to provide good sensitivity and specificity. However, this is a parameter of our method that may require more tuning in the future.

7.3.4 Method

Given the generative model and an optimization technique, we now discuss two methods for detecting RDD in different cases.

Known haplotypes

Assume that we know the true haplotypes of an individual based on some other method for variant detection, as in both [35, 51]. We now treat G (the genome) as an observed variable, which decouples the DNA-Seq and RNA-Seq portions of our generative model. Therefore, we can use the online EM algorithm with the updates described in Section 5.5.2 and above to learn a posterior distribution on the bases at each site given only the RNA-Seq data.

Aside from the posterior distribution, we also want to determine the significance of the difference from the reference at each site. To do so, we first define $O^{t,i}$ as a multinomial random variable for the frequency of observed (probabilistically assigned) fragment bases aligned to transcript t at position i. We then compute the MLE as the number of observations of each base in our experiment,

$$\hat{o}_b^{t,i} = \sum_{f \in \mathcal{F}} \sum_{(t,p,l) \in \mathcal{A}(f,t,i)} \hat{P}_f(t,p,l) \cdot [\mathbb{1}(p < i < p + l(f^5) - 1) \cdot \mathbb{1}(b = f^5_{i-p+1}) + \mathbb{1}(p + l - l(f^3) - 1 < i < p + l - 1) \cdot \mathbb{1}(b = f^3_{p+l(f^3)-i})], \tag{7.10}$$

as well as the number of observations of the most observed non-reference base,

$$\hat{o}^{t,i}_{\max} = -\min_{b \in \mathcal{B}} (1 - \mathbb{1}(b = t_i)) \cdot \hat{o}_b^{t,i}, \tag{7.11}$$

the total number of observations at this position,

$$\hat{o}^{t,i} = \sum_{b \in \mathcal{B}} \hat{o}_b^{t,i}, \tag{7.12}$$

and the expected supporting fragments for each base given the true reference base and the positions and probabilities of the fragment alignments,

$$\mathbb{E}(O_b^{t,i} | t_i) = \sum_{f \in \mathcal{F}} \sum_{(t,p,l) \in \mathcal{A}(f,t,i)} \hat{P}_f(t,p,l) \cdot [\mathbb{1}(p < i < p + l(f^5) - 1) \cdot \epsilon^{5,i-p+1}_{b|t_{i-1},t_i} + \mathbb{1}(p + l - l(f^3) - 1 < i < p + l - 1) \cdot \epsilon^{3,p+l(f^3)-i}_{b|t_{i-1},t_i}]. \tag{7.13}$$

We can now compute our p-value as the probability that any non-reference base would be observed as often as the most commonly observed non-reference base, given the true reference. Thus,

$$\text{p-value}_{t,i} = \sum_{b \in \mathcal{B}: b \neq f_{t,i}} \mathbb{P}(O_b^{t,i} \geq \hat{o}^{t,i}_{\max}), \tag{7.14}$$

where $O_b^{t,i} \sim Normal(np, np(1-p))$ for $n = \hat{o}^{t,i}, p = \mathbb{E}(O_b^{t,i} | t, \mathcal{F}) / \hat{o}^{t,i}$.

Note that the above test implicitly takes into account the depth of coverage. We are careful to correct for multiple testing using a Bonferroni correction to reduce the number of false positives.

General case

The more general case is where we do not know the true haplotypes. We require a reference genome along with DNA-Seq or Exome-Seq reads. Since the observed RNA-Seq and DNA-Seq data are independent given G, we would expect that if we inferred G from the DNA-Seq and RNA-Seq reads independently, we would learn the same distribution within some uncertainty due to our optimization method, sequencing errors, etc. Therefore, our method in this case is to do this comparison.

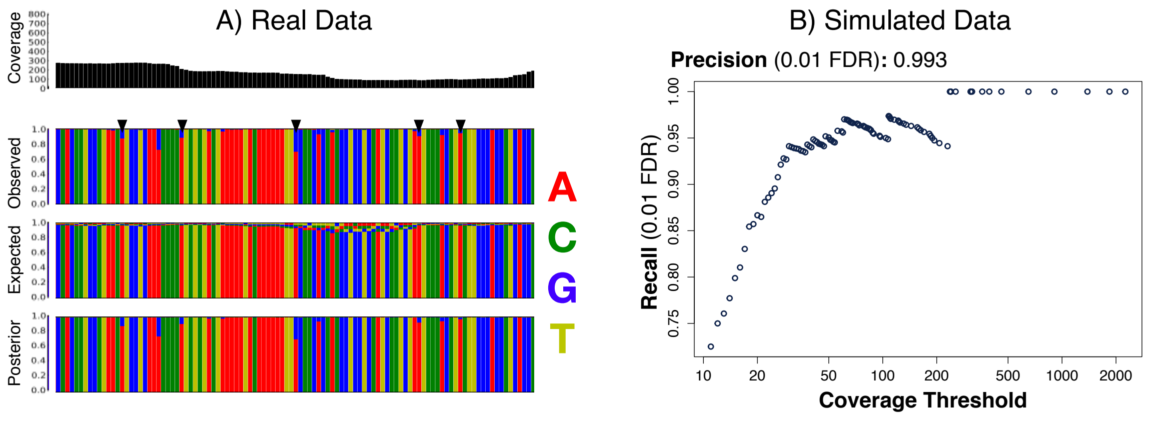


Figure 7.3: *Results for real and simulated data.* (A) The posterior, expected, and observed distributions of bases for a portion of a transcript (NM_001199739) learned with the data from [51]. The black arrowheads denote positions with significant p-values (after Bonferroni correction) according to our method. All are A→I edits and were validated by [51]. (B) Simulations of 50 million paired-end reads and 200 randomly selected RDD sites shows near-perfect precision and excellent recall for significant sites (p-value < 0.01) after Bonferroni correction. Recall improves with coverage and is perfect for all sites with a depth of at least 237 reads.

We again divide the model and infer the transcript sequences given the two datasets in turn. The RNA-Seq data is processed as above. The DNA-Seq data has a similar model except the transcript abundances are fixed to be uniform. This produces two sets of posterior distributions on the bases at all positions.

We now wish to determine if the differences in these distributions are significant. However, we must also take into account a possible difference in coverage between the two experiments. Therefore, we compute our p-value using the bootstrap. For each position in each experiment we sample from the posterior the number of bases covering that position. For each pair of samples we use Fisher's exact test to find a p-value and then take the average p-value over some large number, B, of samples. This average p-value can then be used as a significance score for the position, after correction for multiple testing.

Note that this more general case also applies when a distribution over the genome sequence is provided by some alternative means, without the need to re-infer the genome.

7.4 Results

Figure 7.3 presents initial results for our method. In (A) we show an example of the output of our method on the data analyzed by Peng *et al.*[51]. The positions that we identify as significant in this selection match their findings but in a more statistically-sound manner. A full re-analysis of the data should be completed shortly.

In (B) we present results for a synthetic dataset where we randomly selected 200 sites to be “edited” in the reference. We then processed 50 million transcriptome-wide synthetic reads generated with errors under the *eXpress model*. Our precision was nearly perfect after Bonferroni correction and our recall was excellent even at low coverage. Furthermore, we had perfect recall for any site with at least 237 reads. This analysis using synthetic data lends great support to our method and also provides useful information for determining a coverage threshold in real analyses.

7.5 Discussion

We have presented a likelihood-based method for discovering sites of RDD using a generative model that captures various aspects of the sequencing experiment. We have also described how we believe this model addresses many of the issues that led to false positives in [35] without resorting to the heuristic filtering of [51].

However, there are a few issues that still need to be better addressed by our method. For example, we have not described a way to prevent some of the mapping errors that plagued the analysis of [35]. One solution may be to include the entire genome as a pseudo-transcript in the transcriptome set used in our mappings. This would allow us to better resolve the probabilistic assignment of fragments that come from repeat regions or non-annotated isoforms.

Furthermore, we may need to learn a separate error model based on the fragment end (5’ or 3’) in additional to the sequencing order, as mentioned previously. Based on the observed importance of relative quality scores in calling systematic errors [41], it may be necessary to include these in our error model along with the read position.

Our initial results provide a proof-of-principle for statistically sound detection of RDDs, however a complete analysis of an entire genome is beyond the scope of this thesis.

Chapter 8

Conclusion

In the preceding chapters we have presented an overview of the ambiguous fragment assignment problem, with a focus on RNA-Seq (Chapter 1), a generative model for a typical sequencing experiment called the *eXpress model* (Chapter 2), an EM-based optimization procedure for estimating the hidden parameters of our model (Chapter 3), an improvement to the model and estimation procedure for capturing the effects of sequence-specific fragmentation bias (Chapter 4), multiple ways of scaling the optimization to larger datasets including partitioning the data, using the online EM algorithm of `eXpress` and using cluster computing with `eXpress-D` (Chapter 5), an efficient method for updating estimates after a change in the target sequences (Chapter 6), and a new method for detecting RNA-DNA Differences in RNA-Seq experiments (Chapter 7).

While we have primarily focused on RNA-Seq in this document, only simple modifications are necessary to use `eXpress` and associated methods for other high-throughput sequencing experiments. For example, we have already added a feature to `eXpress` that provides ability to assign multi-reads in ChIP-Seq experiments. This procedure uses binned portions of the genome as target sequences and allows for fragment count estimates of “neighboring” targets to be used in the formula for probabilistic assignment. This procedure is similar to [11] except that it includes models for errors and bias. Metagenomics provides an even more straightforward application whereby full genomes are treated as target sequences. Other applications could include ribosomal profiling [22], which has similarities with RNA-Seq, and copy number variation detection [14], which has similarities with ChIP-Seq.

As more methods are introduced that take advantage of modern sequencing technologies for digital counting and datasets continue to grow larger, the ability to scale solutions to the fragment assignment problem will become even more critical. Furthermore, as new technologies are introduced, likelihood models must be able to adapt to the peculiarities of different platforms while remaining general enough to be widely applicable. In this document, we have described a model and methods that we have shown to be general, precise, adaptable, and scalable to the coming influx of high-throughput sequencing data.

Bibliography

[1] S Anders and W Hüber. Differential expression analysis for sequence count data. *Genome Biology*, 11:R106, 2010.

[2] CD Armour, JC Castle, R Chen, T Babak, P Loerch, S Jackson, JK Shah, J Dey, CA Rohl, JM Johnson, and CK Raymond. Digital transcriptome profiling using selective hexamer priming for cDNA synthesis. *Nature Methods*, 6:647–649, 2009.

[3] YW Asmann, BM Necela, KR Kalari, A Hossain, TR Baker, JM Carr, C Davis, JE Getz, G Hostetter, X Li, SA McLaughlin, DC Radisky, GP Schroth, HE Cunliffe, EA Perez, and EA Thompson. Detection of redundant fusion transcripts as biomarkers or disease-specific therapeutic targets in breast cancer. *Cancer Research*, 72:1921–1928, 2012.

[4] KF Au, H Jiang, L Lin, Y Xing, and WH Wong. Detection of splice junctions from paired-end RNA-Seq data by SpliceMap. *Nucleic Acids Research*, 38:4570–4578, 2010.

[5] CE Bichot and P Siarry, editors. *Graph Partitioning.* John Wiley and Sons, Inc., 2011.

[6] D Blei, A Ng, and M Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[7] JR Bradford, Y Hey, T Yates, Y Li, SD Pepper, and CJ Miller. A comparison of massively parallel nucleotide sequencing with oligonucleotide microarrays for global transcription profiling. *BMC genomics*, 11(1):282, 2010.

[8] D Branton, DW Deamer, A Marziali, H Bayley, SA Benner, T Butler, M Di Ventra, S Garaj, A Hibbs, X Huang, SB Jovanovich, PS Krstic, S Lindsay, XS Ling, CH Mastrangelo, A Meller, JS Oliver, YV Pershin, JM Ramsey, R Riehn, GV Soni, V Tabard-Cossa, M Wanunu, M Wiggin, and JA Schloss. The potential and challenges of nanopore sequencing. *Nature Biotechnology*, 26:1146–1153, 2008.

[9] JH Bullard, E Purdom, KD Hansen, and S Dudoit. Evaluation of statistical methods for normalization and differential expression in mRNA-Seq experiments. *BMC Bioinformatics*, 11:94, 2010.

[10] O Cappé and E Moulines. On-line expectation-maximization algorithm for latent data models. *Journal of the Royal Statistical Society*, 71:593–613, 2009.

[11] D Chung, PF Kuan, B Li, R Sanalkumar, K Liang, EH Bresnick, C Dewey, and S Keles. Discovering transcription factor binding sites in highly repetitive regions of genomes with multi-read analysis of ChIP-Seq data. *PLOS Computational Biology*, 7:e1002111, 2011.

[12] J Dean and S Ghemawat. MapReduce: simplified data processing on large clusters. In *Proceedings of the 6th conference on Symposium on Operating Systems Design & Implementation - Volume 6*, OSDI'04, pages 10–10, Berkeley, CA, USA, 2004. USENIX Association.

[13] AP Dempster, NM Laird, and DB Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

[14] J Duan, JG Zhang, HW Deng, and YP Wang. Comparative studies of copy number variation detection methods for next-generation sequencing technologies. *PloS one*, 8:e59128, 2013.

[15] S Fleige and MW Pfaffl. RNA integrity and the effect on the real time qRT-PCR performance. *Molecular Aspects of Medicine*, 27:126–139, 2006.

[16] S Ghemawat, H Gobioff, and ST Leung. The Google file system. In *ACM SIGOPS Operating Systems Review*, volume 37, pages 29–43. ACM, 2003.

[17] BR Graveley, AN Brooks, JW Carlson, MO Duff, JM Landolin, L Yang, CG Artieri, MJ van Baren, N Boley, BW Booth, JB Brown, L Cherbas, CA Davis, A Dobin, R Li, W Lin, JH Malone, NR Mattiuzzo, D Miller, D Sturgill, BB Tuch, C Zaleski, D Zhang, M Blanchette, S Dudoit, B Eads, RE Green, A Hammonds, L Jiang, P Kapranovand L Langton, N Perrimon, JE Sandler, KH Wan, A Willingham, Y Zhang, Y Zou, J Andrews, PJ Bickel, SE Brenner, MR Brent, P Cherbas, TR Gingeras, RA Hoskins, TC Kaufman, B Oliver, and SE Celniker. The developmental transcriptome of drosophila melanogaster. *Nature*, 471:473–479, 2010.

[18] KD Hansen, SE Brenner, and S Dudoit. Biases in Illumina transcriptome sequencing caused by random hexamer priming. *Nucleic Acids Research*, 38:1–7, 2010.

[19] T Hashimoto, MJ de Hoon, SM Grimmond, CO Daub, Y Hayashizaki, and GJ Faulkner. Probabilistic resolution of multi-mapping reads in massively parallel sequencing data using MuMRescueLite. *Bioinformatics*, 19:2613–2614, 2009.

[20] U Hoelzle and LA Barroso. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines.* Morgan and Claypool Publishers, 1st edition, 2009.

[21] T Hunter, T Moldovan, M Zaharia, S Merzgui, J Ma, MJ Franklin, P Abbeel, and AM Bayen. Scaling the mobile millennium system in the cloud. In *Proceedings of the 2nd ACM Symposium on Cloud Computing*, page 28. ACM, 2011.

[22] NT Ingolia, S Ghaemmaghami, JRS Newman, and JS Weissman. Genome-wide analysis in vivo of translation with nucleotide resolution using ribosome profiling. *science*, 324:218–223, 2009.

[23] H Jiang and W Wong. Statistical inferences for isoform expression in RNA-Seq. *Bioinformatics*, 25:1026–1032, 2009.

[24] Y Katz, ET Wang, EM Airoldi, and CB Burge. Analysis and design of rna sequencing experiments for identifying isoform regulation. *Nature methods*, 7(12):1009–1015, 2010.

[25] M Kircher and J Kelso. High-throughput DNA sequencing– concepts and limitations. *BioEssays*, 32:524–536, 2010.

[26] CL Kleinman and J Majewski. Comment on Widespread RNA and DNA sequence differences in the human transcriptome. *Science*, 335:1302, 2012.

[27] B Langmead, KD Hansen, JT Leek, et al. Cloud-scale rna-sequencing differential expression analysis with myrna. *Genome Biology*, 11:R83, 2010.

[28] B Langmead, MC Schatz, J Lin, M Pop, and SL Salzberg. Searching for SNPs with cloud computing. *Genome Biology*, 10:R134, 2009.

[29] B Langmead, C Trapnell, M Pop, and SL Salzberg. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biology*, 10:R25, 2009.

[30] J Levin, X Adiconis, M Yassour, D Thompson, M Guttman, M Berger, L Fan, N Friedman, C Nussbaum, A Gnirke, and A Regev. Development and evaluation of RNA-Seq methods. *Genome Biology*, 11:P26, 2010.

[31] B Li and C Dewey. RSEM: accurate transcript quantification from RNA-Seq data with or without a reference genome. *BMC Bioinformatics*, 12(1):323, 2011.

[32] B Li, V Ruotti, RM Stewart, JA Thomson, and CN Dewey. RNA-Seq gene expression estimation with read mapping uncertainty. *Bioinformatics*, 26:493–500, 2010.

[33] H Li and R Durbin. Fast and accurate short read alignment with Burrows-Wheeler Transform. *Bioinformatics*, 25:1754–1760, 2009.

[34] J Li, H Jiang, and WH Wong. Modeling non-uniformity in short-read rates in RNA-Seq data. *Genome Biology*, 11:R50, 2010.

[35] M Li, IX Wang, Y Li, A Bruzel, AL Richards, JM Toung, and VG Cheung. Widespread RNA and DNA sequence differences in the human transcriptome. *Science*, 333:53–58, 2011.

[36] W Li, J Feng, and T Jiang. IsoLasso: a LASSO regression approach to RNA-Seq based transcriptome assembly. *Journal of Computational Biology*, 18:1693–1707, 2011.

[37] W Li and T Jiang. Transcriptome assembly and isoform expression level estimation from biased RNA-Seq reads. *Bioinformatics*, 28:2914–2921, 2012.

[38] P Liang and K Klein. Online EM for unsupervised models. *Proceedings of NAACL*, pages 611–619, 2009.

[39] W Lin, R Piskol, MH Tan, and JB Li. Comment on Widespread RNA and DNA sequence differences in the human transcriptome. *Science*, 335:1302, 2012.

[40] S Marguerat and J Bähler. RNA-Seq: from technology to biology. *Cellular and Molecular Life Sciences*, 67:569–579, 2010.

[41] F Meacham, D Boffelli, J Dhahbi, D Martin, M Singer, and L Pachter. Identification and correction of systematic error in high-throughput sequence data. *BMC Bioinformatics*, 12:451, 2011.

[42] P Meinicke, KP Asshauer, and T Linger. Mixture models for analysis of the taxonomic composition of metagenome. *Bioinformatics*, 27:1618–1624, 2011.

[43] AM Mezlini, EJM Smith, M Fiume, O Buske, G Savich, S Shah, S Aparicion, D Chiang, A Goldenberg, and M Brudno. ireckon: Simultaneous isoform discovery and abundance estimation from RNA-Seq data. *Genome Research*, 23:519–529, 2012.

[44] A Mortazavi, B Williams, K McCue, and L Schaeffer. Mapping and quantifying mammalian transcriptomes by RNA-Seq. *Nature Methods*, 2008.

[45] M Nicolae, S Mangul, II Mandoiu, and A Zelikovsky. Estimation of alternative splicing isoform frequencies from rna-seq data. *Algorithms for Molecular Biology*, 6(1):9, 2011.

[46] M Nicolae, S Mangul, I Măndoiu, and A Zelikovsky. Estimation of alternative splicing isoform frequencies from RNA-Seq data. *Algorithms in Bioinformatics*, 6293:202–214, 2010.

[47] CA Ouzounis and PD Karp. The past, present and future of genome-wide re-annotation. *Genome Biology*, 3, 2002.

[48] L Pachter. Models for transcript quantification from RNA-Seq. *arXiv*, 2011.

[49] L Pachter. A closer look at RNA editing. *Nature Biotechnology*, 30:246–247, 2012.

[50] B Paaniuc, N Zaitlen, and E Halperin. Accurate estimation of expression levels of homologous genes in RNA-Seq experiments. In Bonnie Berger, editor, *Research in Computational Molecular Biology*, volume 6044 of *Lecture Notes in Computer Science*, pages 397–409. Springer Berlin / Heidelberg, 2010.

[51] Z Peng, Y Cheng, BCM Tan, L Kang, Z Tian, Y Zhu, W Zhang, Y Liang, X Hu, X Tan, J Guo, Z Dong, Y Liang, L Bao, and J Wang. Comprehensive analysis of RNA-Seq data reveals extensive RNA editing in a human transcriptome. *Nature Biotechnology*, 30:253–260, 2012.

[52] JK Pickrell, Y Gilad, and JK Pritchard. Comment on Widespread rna and dna sequence differences in the human transcriptome. *Science*, 335:1302, 2012.

[53] A Roberts and L Pachter. Streaming fragment assignment for real-time analysis of sequencing experiments. *Nature Methods*, 1:71–73, 2013.

[54] A Roberts, L Schaeffer, and L Pachter. Updating RNA-Seq analyses after re-annotation. *Bioinformatics*, 29:1631–1637, 2013.

[55] A Roberts, C Trapnell, J Donaghey, JL Rinn, and L Pachter. Improving RNA-Seq expression estimates by correcting for fragment bias. *Genome Biology*, 12:R22, 2011.

[56] G Robertson, M Hirst, M Bainbridge, M Bilenky, Y Zhao, T Zeng, G Euskirchen, B Bernier, R Varhol, A Delaney, N Thiessen, OL Griffith, A He, M Marra, M Snyder, and S Jones. Genome-wide profiles of STAT1 DNA association using chromatin immunoprecipitation and massively parallel sequencing. *Nature Methods*, 4:651–657, 2007.

[57] J Rung and A Brazma. Reuse of public genome-wide gene expression data. *Nature Reviews Genetics*, 14:89–99, 2013.

[58] A Sboner, XJ Mu, D Greenbaum, RK Auerbach, and MB Gerstein. The real cost of sequencing: higher than you think! *Genome Biology*, 12:125, 2011.

[59] S Schultheiss, G Jean, J Behr, R Bohnert, P Drewe, N Görnitz, A Kahles, P Mudrakarta, VT Sreedharan, G Zeller, and G Rätsch. Oqtans: a Galaxy-integrated workflow for quantitative transcriptome analysis from NGS data. *BMC Bioinformatics*, 12:A7, 2011.

[60] L Shi, LH Reid, WD Jones, R Shippy, JA Warrington, SC Baker, PJ Collins, F de Longueville, ES Kawakasi, KY Lee, et al. The MicroArray Quality Control (MAQC) project shows inter- and intraplatform reproducibility of gene expression measurements. *Nature Biotechnology*, 24:1151–1161, 2006.

[61] K Shvachko, H Kuang, S Radia, and R Chansler. The Hadoop distributed file system. In *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*, pages 1–10. IEEE, 2010.

[62] S Srivastava and L Chen. A two-parameter generalized Poisson model to improve the analysis of RNA-Seq data. *Nucleic Acids Research*, 38:e170, 2010.

[63] LD Stein. The case for cloud computing in genome informatics. *Genome Biology*, 11:207, 2010.

[64] M Taub, D Lipson, and TP Speed. Methods for allocating ambiguous short-reads. *Communications in Information and Systems*, 10:69–82, 2010.

[65] C Trapnell, DG Hendrickson, M Sauvageau, L Goff, JL Rinn, and L Pachter. Differential analysis of gene regulation at transcript resolution with RNA-Seq. *Nature Biotechnology*, 31:46–53, 2013.

[66] C Trapnell, A Roberts, L Goff, G Pertea, D Kim, DR Kelley, H Pimentel, SL Salzberg, JL Rinn, and L Pachter. Differential gene and transcript expression analysis of RNA-seq experiments with TopHat and Cufflinks. *Nature Protocols*, 7:562–578, 2012.

[67] C Trapnell, BA Williams, G Pertea, A Mortazavi, GK, MJ van Baren, SL Salzberg, BJ Wold, and L Pachter. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nature Biotechnology*, 28:511–515, 2010.

[68] ET Wang, R Sandberg, S Luo, I Khrebtukova, L Zhang, C Mayr, SF Kingsmore, GP Schroth, and CB Burge. Alternative isoform regulation in human tissue transcriptomes. *Nature*, 456:470–476, 2008.

[69] J Wang, W Wang, R Li, Y Li, G Tian, L Goodman, W Fan, J Zhang, J Li, J Zhang, Y Guo, B Feng, H Li, Y Lu, X Fang, H Liang, Z Du, D Li, Y Zhao, Y Hu, Z Yang, H Zheng, I Hellmann, M Inouye, J Pool, X Yi, J Zhao, J Duan, Y Zhou, J Qin, L Ma, G Li, Z Yang, G Zhang, B Yang, C Yu, F Liang, W Li, S Li, D Li, P Ni, J Ruan, Q Li, H Zhu, D Liu, Z Lu, N Li, G Guo, J Zhang, J Ye, L Fang, Q Hao, Q Chen, Y Liang, Y Su, A San, C Ping, S Yang, F Chen, L Li, K Zhou, H Zheng, Y Ren, L Yang, Y Gao, G Yang, Z Li, X Feng, D Kristiansen, GK Wong, R Nielsen, R Durbin, L Bolund, X Zhang, S Li, H Yang, and J Wang. The diploid genome sequence of an Asian individual. *Nature*, 456:60–65, 2008.

[70] B Wold and RM Myers. Sequence census methods for functional genomics. *Nature Methods*, 5:19–21, 2005.

[71] Y Xing, T Yu, YN Wu, M Roy, J Kim, and C Lee. An expectation-maximization algorithm for probabilistic reconstructions of full-length isoforms from splice graphs. *Nucleic Acids Research*, 34:3150–3160, 2006.

[72] D Yorukoglu, F Hach, L Swanson, CC Collins, I Birol, and SC Sahinalp. Dissect: detection and characterization of novel structural alterations in transcribed sequences. *Bioinformatics*, 28:i179–i187, 2012.

[73] M Zaharia, M Chowdhury, T Das, A Dave, J Ma, M McCauley, M Franklin, S Shenker, and I Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pages 2–2. USENIX Association, 2012.