

Spring 5-18-2012

Software for Estimation of Human Transcriptome Isoform Expression Using RNA-Seq Data

Kristen Johnson

Follow this and additional works at: <https://scholarworks.uno.edu/td>



Part of the [Numerical Analysis and Scientific Computing Commons](#), and the [Other Computer Sciences Commons](#)

Recommended Citation

Johnson, Kristen, "Software for Estimation of Human Transcriptome Isoform Expression Using RNA-Seq Data" (2012). *University of New Orleans Theses and Dissertations*. 1448.
<https://scholarworks.uno.edu/td/1448>

This Thesis is protected by copyright and/or related rights. It has been brought to you by ScholarWorks@UNO with permission from the rights-holder(s). You are free to use this Thesis in any way that is permitted by the copyright and related rights legislation that applies to your use. For other uses you need to obtain permission from the rights-holder(s) directly, unless additional rights are indicated by a Creative Commons license in the record and/or on the work itself.

This Thesis has been accepted for inclusion in University of New Orleans Theses and Dissertations by an authorized administrator of ScholarWorks@UNO. For more information, please contact scholarworks@uno.edu.

Software for Estimation of Human Transcriptome Isoform Expression Using RNA-Seq Data

A Thesis

Submitted to the Graduate Faculty of the
University of New Orleans
in partial fulfillment of the
requirements for the degree of

Master of Science
in
Computer Science

by

Kristen Johnson

B.S. University of New Orleans, 2007

May 2012

@Copyright 2012, Kristen Johnson

Acknowledgments

I would like to thank my professors for all the knowledge they have passed on to me since I began my studies in Computer Science. Most importantly I give special thanks to my advisor, Dr. Dongxiao Zhu, for sharing his knowledge of bioinformatics as well as his support and advice throughout the course of my graduate studies and research. Dr. Zhiyu Zhao, who got me started in bioinformatics research and who I have enjoyed working with since. Dr. Christopher Summa, for his endless advice and good humor. And finally, Dr. Jaime Nino, for teaching me everything about programming I needed to know, just as he promised.

I would also like to thank everyone in our Bioinformatics group, especially Nan Deng. Nan taught me everything she knew about this topic and helped make this thesis possible. I wish to thank her for her patience with answering my endless questions.

Finally, I would like to thank my family and friends who have always supported me. My love and gratitude go to my mom and siblings for keeping me focused and understanding why I am always busy, my aunt Kyle for getting me back on my feet, and my best friends Sancy and Jeff for keeping me there. I would also like to thank my boyfriend, Brendan, for never getting tired of me picking his brain, always making me laugh, and making me as confident as he is by always supporting me. Lastly I wish to thank my grandmother, quite possibly the most important person I have had in my life so far, for being my guide, best friend, and inspiration to stay strong and make the most of my life.

Table of Contents

List of Figures	vi
Abstract	vii
Chapter 1: Introduction	1
1.1: Motivation.....	1
1.2: Objectives	1
1.3: Contributions	2
1.4: Overview.....	3
Chapter 2: Background Information	4
2.1: Genome and Transcriptome.....	4
2.2: Central Dogma of Molecular Biology	4
2.3: Alternative Splicing	5
2.4: Next-Generation Sequencing.....	6
2.5: RNA-Seq Technology.....	8
2.5.1: Short Read Alignment	10
2.6: Isoform Expression Level Estimation	11
2.6.1: Calculation of NP_i and EV_i Via EM-Like Algorithm	12
2.7: Expectation Maximization (EM) Algorithm	13
Chapter 3: Related Work	19
3.1: Enhanced Read Analysis of Gene Expression (ERANGE).....	19
3.2: Cufflinks	20
3.3: IsoformEx	21
3.4: RNA-Seq by Expectation Maximization (RSEM).....	22
3.5: IsoEM.....	25
3.6: Read Assignment via EM (RAEM).....	27
Chapter 4: Software	29
4.1: Introduction.....	29
4.2: Supporting Classes.....	29
4.2.1: Gene	29
4.2.2: mRNA.....	30
4.2.3: Exon	30
4.2.4: EmInfo	31
4.2.5: ExonReadLocationFilter	31
4.3: Input Files and File Processing Classes.....	32
4.3.1: GFF File and GffProcessor	33
4.3.2: SAM File and ReadsFileProcessor	34
4.3.3: RPKM File and RpkmFileReader.....	35
4.4: Main Class: MultireadIsoformLevelEstimator (MILE).....	36
4.4.1: Step One: Data Structure Initialization.....	37
4.4.2: Step Two: Isoform Estimation using the EM Algorithm.....	37

4.5: Critique and Comparison of Programs	41
4.5.1: GWIE Pros and Cons	41
4.5.2: ChromIE Pros and Cons	42
Chapter 5: Results	44
5.1: iGWIE Results	45
5.2: ChromIE Results.....	47
Chapter 6: Conclusion and Future Works.....	50
References.....	52
Vita.....	55

List of Figures

Figure 1. Central Dogma of Molecular Biology and Alternative Splicing	5
Figure 2. RNA-Seq Workflow Illustrated Using Illumina Sequencer	9
Figure 3. UML Diagrams of Classes Gene, mRNA, and Exon	30
Figure 4. UML Diagrams of Classes EmInfo and ExonReadLocationFilter	32
Figure 5. UML Diagrams of File Processing Classes	35
Figure 6. UML Diagrams of MILE	36
Figure 7. E and M Steps of GWIE and ChromIE	40
Figure 8. iGWIE Intermediate Results	45
Figure 9. iGWIE Predicted Copy Proportions vs. Real Copy Proportions	45
Figure 10. iGWIE Predicted Read Proportions vs. Real Read Proportions	46
Figure 11. ChromIE Analysis of Chromosome X	47
Figure 12. ChromIE Predicted Copy Proportions vs. Real Copy Proportions	47
Figure 13. ChromIE Predicted Read Proportions vs. Real Read Proportions	48
Figure 14. Isoform Proportion Estimates with Read Coverage	49

Abstract

The goal of this thesis research was to develop software to be used with RNA-Seq data for transcriptome quantification that was capable of handling multireads and quantifying isoforms on a more global level. Current software available for these purposes uses various forms of parameter alteration in order to work with multireads. Many still analyze isoforms per gene or per researcher determined clusters as well. By doing so, the effects of multireads are diminished or possibly wrongly represented. To address this issue, two programs, GWIE and ChromIE, were developed based on a simple iterative EM-like algorithm with no parameter manipulation. These programs are used to produce accurate isoform expression levels.

Keywords

RNA-Seq

Transcriptome Quantification

Isoform Expression

Multireads

Expectation-Maximization (EM) Algorithm

Chapter 1: Introduction

1.1: Motivation

Estimation of gene expression levels from RNA-Seq data can be achieved with greater accuracy through estimation of the transcript abundance of each gene. This estimation is a useful biological tool for understanding transcriptional regulation and gene functionality. Increasing understanding of the transcriptome is important for determining the characteristics of both normal development in cells, as well as the progression of various diseases [2, 38].

As a result of the use of high-throughput, next-generation sequencing technologies, most notably RNA-Seq, data can be obtained that provides a more absolute coverage of transcript levels than previous microarray technologies. However, due to the size and complexity of this data, programs must be designed to aid biomedical researchers in its interpretation. The large amount of data produced by RNA-Seq is a typical problem faced by bioinformatics researchers. Efficient methods for storage, retrieval, and processing must be designed to further research in this field [38, 39, 40].

The software designed for this thesis presents a simple command line interface that merely requires the input paths for three files. From this, it will perform direct isoform expression calculation without the need for confusing parameter setting or understanding, as is common with many popular bioinformatics software tools. Despite the drawbacks of computational time due to large input data files, the trend towards faster, more powerful computers supports that this approach can be useful because of its simplicity and accuracy.

1.2: Objectives

The initial objective of this thesis was to extend the work previously done in [8] to design and implement a program that would encompass all reads. Previously, only separate reads are

considered during the estimation. Separate reads are those reads that map only to one gene, meaning multiple mappings were discarded from the calculation.

An additional goal was to broaden the level of analysis from a gene based level to a more global level in order to obtain a more accurate analysis of all isoforms simultaneously. Previously, only the isoforms of each gene are compared to each other. This ignores the statistical contribution of multiple mappings to multiple genes.

1.3: Contributions

The main contributions of this thesis are the achievement of the stated objectives. In order to achieve these goals, two programs were designed and implemented using the programming language Java. Both programs consider all reads, not just separate reads during the isoform expression estimation. Each program achieves estimation at a more general, global level than the gene based approach in [8].

Genome Wide Isoform Estimation (GWIE) was designed to include all reads and to analyze them at the global level. This means that all reads are assigned to the isoforms that they could possibly belong to, which could fall under multiple genes. During the estimation, over 90,000 isoforms of all genes are compared using the EM algorithm.

Chromosomal Isoform Estimation (ChromIE) was designed with the hope to achieve faster results than GWIE, while incorporating all reads but analyzing them per chromosome. This achieves multiple read mapping inclusion and estimation at a level higher than the gene based approach in [8]. However, in this exchange for quicker result production, ChromIE is not as global as the genome wide approach of GWIE. For example, while computing the estimated levels for the isoforms of chromosome X, ChromIE compares 3,989 expressed isoforms with

831,053 short reads to these isoforms. For chromosome one, ChromIE compares 11,215 expressed isoforms with 1,900,016 short reads.

In addition to these software based contributions, additional work was contributed to two publications. For [8], test programs were implemented and used for results verification. Other party software was run to further support the findings of [8] to fulfill reviewer revision suggestions before publication. For [43] the de novo transcriptome assemblers, SPATA and Trans-ABYSS, were run using both human and mouse data in order to analyze performance and accuracy.

1.4: Overview

This thesis is organized into six chapters. Chapter one consists of a brief introduction and the objectives, motivation, and contributions behind this work. Chapter two consists of a discussion of the background information necessary for understanding why the software was designed for this thesis. Chapter three covers software that computes isoform expression levels like the software of this thesis. Chapter four describes in detail the software designed for achieving the goals of this thesis. Chapter five discusses the intermediate results of the two programs. Chapter six ends this thesis with conclusions and future goals concerning this work.

Chapter 2: Background Information

2.1: Genome and Transcriptome

The genome of an organism constitutes all of the biological information needed to build and maintain that organism. This biological information is represented through deoxyribonucleic acid (DNA), which is further segmented into chromosomes. These chromosomes can further be divided into multiple genes. These genes carry the instructions necessary for producing ribonucleic acid (RNA) or protein molecules [1, 3]. For the purposes of this thesis, chromosomes one through twenty-two, X, and Y are considered but only the protein coding genes of these chromosomes, 21,280 in total, are evaluated [9].

The transcriptome refers to the set and amount of all transcripts, which include all forms of RNA molecules, in a cell at a given time. This time could be during a stage of normal development or a certain condition, such as disease. This leads to the biological importance of understanding the transcriptome in order to determine the progression of both normal development and diseases and is particularly useful in the study of cancer [1, 8, 39, 40]. This thesis evaluates the expression of only 100,298 possible mRNAs [9].

2.2: Central Dogma of Molecular Biology

According to the central dogma of molecular biology, DNA encodes for RNA, which encodes for protein. Though there are special cases, such as RNA encoding for RNA, of most importance to the study of the transcriptome is the flow of information from DNA to RNA [6]. Transcription is the transference of DNA information to a messenger RNA (mRNA) molecule. For eukaryotic cells, a primary transcript called pre-mRNA undergoes further processing, such as alternative splicing, before translation from mRNA to protein. This splicing can cause the availability of multiple transcript isoforms from genes, thus increasing the complexity of this

dogma [16]. The thesis software aims to evaluate the expression levels of these transcripts in order to determine which isoforms of which genes are most highly expressed at a given time within a cell, or within a certain type of tissue sample. Figure 1 below shows the flow of genetic information through both the central dogma and alternative splicing.

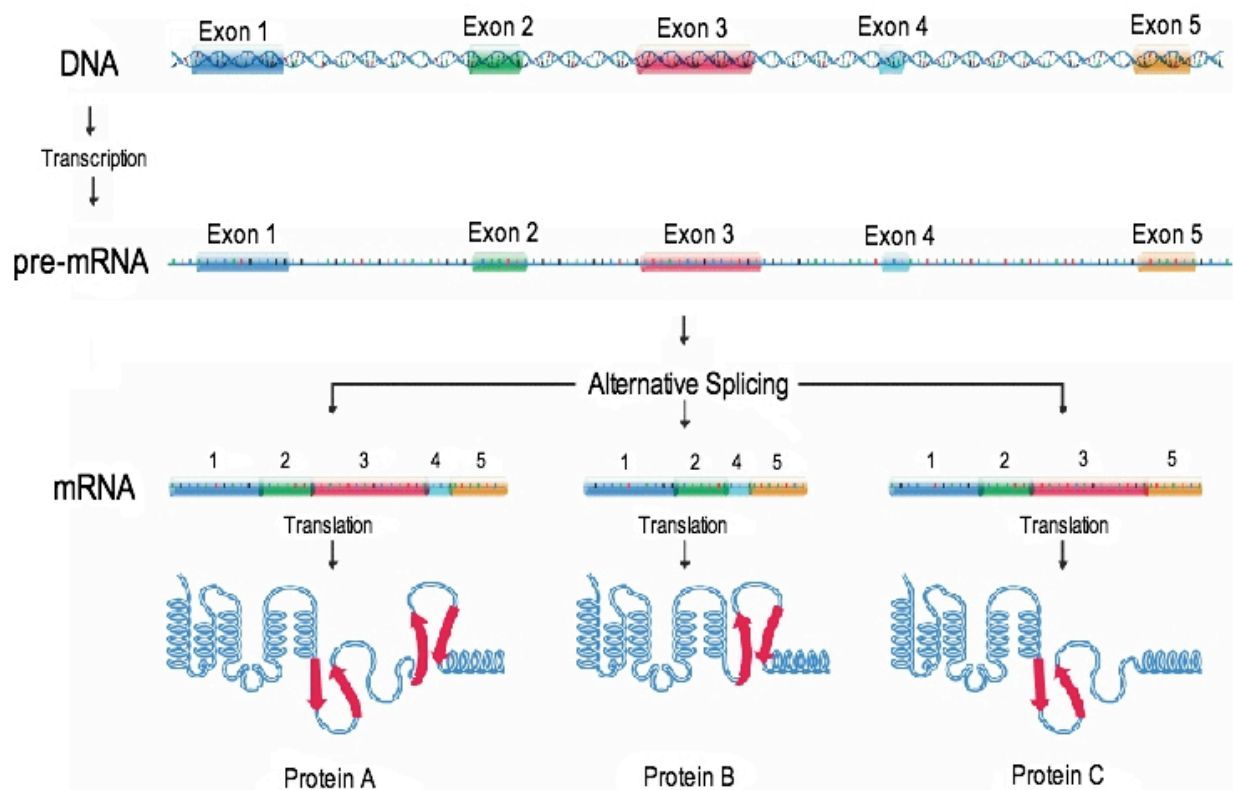


Figure 1. Central Dogma of Molecular Biology and Alternative Splicing. The flow of genetic information via the central dogma can be seen via the arrows. DNA is transcribed into pre-mRNA which undergoes alternative splicing to produce mRNA. Different isoforms of mRNA are translated into different proteins. This figure was modified from [2].

2.3: Alternative Splicing

While the genome is more fixed, with the exception of mutations, the transcriptome is highly variable, dynamic, and dependent on external environmental conditions, such as tissue types. Alternative splicing is considered a key factor in the cause of the transcriptome, cellular, and functional complexity found in eukaryotes. Due to limited coverage and sensitivity of first-

generation sequencing techniques, the extent of alternative splicing found in humans was not fully explored until the advent of next-generation sequencers [24, 26, 40].

Transcripts undergo processing via splicing to remove introns. This is done as the precursory step to creation of messenger RNA (mRNA) molecules, which only contain exons. Alternative splicing not only increases the range of the transcriptome, but also produces multiple transcripts from a single gene because of the splicing variants produced due to inclusion or exclusion of exons, as can be seen in Figure 1 [40].

2.4: Next-Generation Sequencing

Next-generation sequencers can generate over tens of millions of short reads per experiment. These short reads are generated from a library of nucleotide sequences and are commonly referred to as the high-throughput data output by next-generation sequencers. The produced reads can range in length, from as short as 25 base pairs (bp) to as long as 200 bp [21, 23, 24, 30].

Prior to next-generation technologies, methods that used microarrays, Sanger sequencing, serial analysis gene expression (SAGE), or cap analysis gene expression (CAGE) were used to determine full-length cDNA sequences. Use of Sanger sequencing based methods dominated DNA sequencing for over three decades. However, through the effort to understand the human genome, it was realized that greater throughput was necessary. Around 2004 to 2005, development of inexpensive, non-Sanger based sequencing technologies by 454 Life Sciences and the lab of George Church began a sequencing revolution. Despite initial negative attitudes towards aspects such as sequencing validity, read length, and manageability of the volume of data generated by these new sequencers, the technology continued to develop due to the fact that

most Sanger based technologies suffered similar problems during their initial development [21, 23, 30].

The most commonly used next-generation sequencers of today include the Illumina Genome Analyzer, HiSeq (Illumina/Solexa), Applied Biosystems' Solid Sequencing (SOLiD), and Roche/Life Sciences' 454 Sequencing (454 Sequencing). These different platforms acquire the same information and can be applied in similar or different research areas. For instance, Illumina has been used recently for sequencing mammalian transcriptomes. SOLiD has been used for profiling stem cell transcriptomes. While 454 sequencing has been used to discover single nucleotide polymorphisms (SNPs) in corn [21, 23, 30].

Each approach employs a different sequencing chemistry. Illumina uses polymerase based sequence-by-synthesis, SOLiD uses ligation based sequencing, and 454 Sequencing uses pyrosequencing. The parallelized version of pyrosequencing used by 454 Sequencing uses emulsion PCR in combination with detection of added nucleotides to nascent DNA via light generated by luciferase to generate reads. Illumina's detection is based on the combination of bridge amplification and reversible dye-terminators. SOLiD uses emulsion PCR prior to sequencing by annealing and ligation of oligonucleotides [21, 23, 30].

Each sequencer has varying performance rates for paired end separation, mb per run, read length, running time, and cost. 454 is the fastest, taking only several hours, and cheapest for cost per run, but most expensive for cost per mb. SOLiD takes approximately five days to sequence paired end data and is the most expensive. Illumina can take between one to two weeks and is more expensive than 454 Sequencing but less expensive than SOLiD. Though these three sequencers remain the most popular and widely used currently, several other next-generation sequencers are available for use as well [30].

2.5: RNA-Seq Technology

RNA-Seq technology is the use of high-throughput data produced by the next generation sequencing technologies (as described in section 2.4) to classify the transcriptome of individual cells, specific tissues, or entire organisms [23]. Next-generation tools provide excellent genome coverage and base level resolution, which allows for efficient measurement of transcriptome data in an experimental setting. It allows precise measurement of the levels of transcripts. RNA-Seq has allowed the expansion of the study of transcriptomes, an important biological achievement because transcriptomes provide information concerning complexity, gene regulation, and protein information. Using RNA-Seq experiments to study the transcriptome has thus become widely used for studying diseases, such as cancer, as well as embryonic development and stem cells. The most common goals of RNA-Seq experiments are to identify novel transcripts, identify splice junctions, detect alternative splicing, and quantify isoform expression [16, 23, 33, 38, 39, 40].

In a typical RNA-Seq experiment, RNA samples are fragmented into a given length. These fragments are then converted into cDNA for priming. After this step, the cDNA is then converted into a nucleotide library that will be sequenced by a next-generation sequencer to produce reads. These reads, which range in base pair length, are then mapped back to the reference genome of interest. Reads usually map to the exons of a genome or to the exon-exon junctions of a transcriptome [16, 24, 40]. An example of the steps of an RNA-Seq experiment is illustrated below in Figure 2, which depicts the workflow using an Illumina sequencer [13].

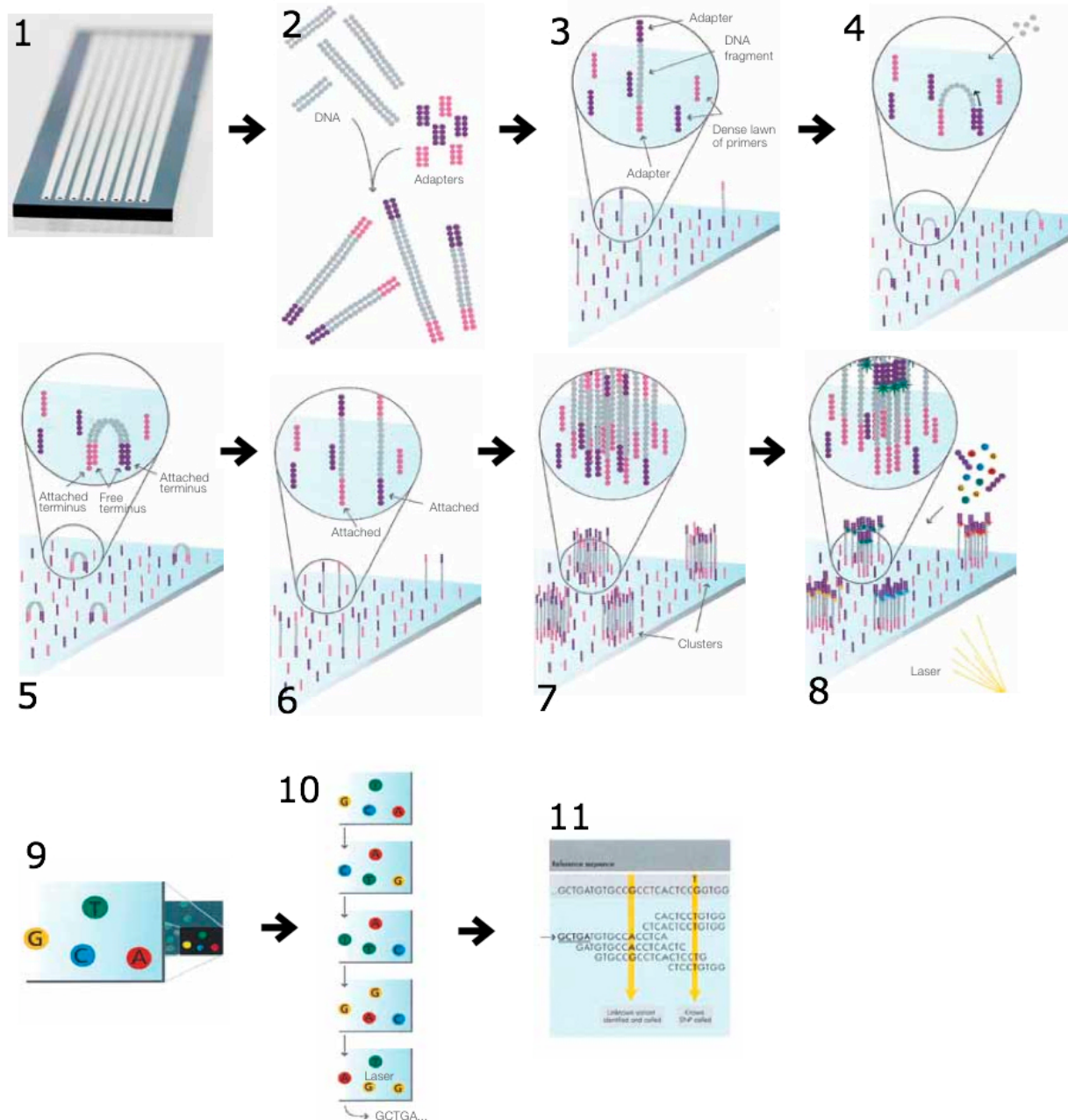


Figure 2. RNA-Seq Workflow Illustrated Using Illumina Sequencer. The above steps (1 – 11) show the steps taken during sequencing using an Illumina Sequencer. This figure is available on the Illumina website for researcher use and was pieced together using [13]. (1) The eight-lane flow cell of an Illumina Sequencing System which can hold several samples at a time. (2) DNA is randomly fragmented and ligate adapters are added to both ends. (3) Single-strand fragments are bound to the surface of the flow cell channels. (4) Bridge amplification is performed by adding unlabeled nucleotides and enzyme. (5) The enzyme uses the nucleotides added in step (4) to build double-stranded bridges. (6) The double-stranded bridges are denatured into single-stranded templates. (7) Complete amplification is performed by generating several million clusters of double-stranded DNA in the channels. (8) The sequencing cycle begins by

determining the first base. (9) Laser excitation is performed in (8) and the emitted fluorescence is used to determine the base. The sequencing cycle (steps 8 and 9) is repeated as shown in (10) for each base in order to determine the sequence of a fragment. (11) The sequence found in (10) is then aligned to a reference to identify any differences.

2.5.1: Short Read Alignment

The typical first step of most RNA-Seq experiments is to align short reads produced by next-generation sequencers to a reference genome [40]. For this thesis, the short read aligner Bowtie and splice junction mapper TopHat were used to produce the Sequence Alignment/Map (SAM) file used by the software.

The SAM format can be used for both single or paired-end reads from various sequencers. SAM files are broken into a header and alignment section, with the latter consisting of eleven required fields and additional optional fields. The required fields consist of information concerning the read pair name, a bitwise flag indicating pair, strand, mate, etc., a reference name, position, mapping quality, CIGAR string describing matches, insertions, etc., mate's reference name and position, insert size, sequence, and quality. Similar to the SAM format, is the Binary Alignment/Map (BAM) format. BAM is a compressed binary version of the SAM format used for performance improvement [19]. The SAM format is described in more detail in section 4.3.2.

TopHat maps spliced junctions of RNA-Seq reads to large genomes via Bowtie. Bowtie aligns large amounts of short reads of DNA sequences to large genomes. After alignment, TopHat identifies splicing junctions located between exons. TopHat outputs four files: `accepted_hits.bam`, `junctions.bed`, `insertions.bed`, `deletions.bed`. Of these, GWIE and ChromIE only use the `accepted_hits.bam` file directly, which is a list of read alignments in SAM format. Bowtie can also output directly in SAM format when using optional flags at the command line interface [34, 35]. Using the read alignment file, read IDs and the exons they were aligned to can

be used to evaluate the expression of isoforms based upon the number of reads that map to the exons of the isoforms.

A common problem with quantifying the transcriptome is that of multireads, i.e. a short read is mapped to multiple locations in the reference genome. Unique reads are short reads that map to a unique location in the genome. The latter reads are the easiest to handle in most programs. Multireads increase the complexity of transcript quantification due to the various approaches of how to assign these multireads and to which isoforms they truly belong.

2.6: Isoform Expression Level Estimation

The estimation of gene expression levels is an important biological research goal and has been accomplished via transcriptome isoform level measurement. The most popular method used to estimate expression levels is the formula for the number of mapped reads per kilobase of exons per million mapped reads (RPKM). The RPKM calculation is as follows:

$$RPKM = \frac{10^9 * (M_{reads})}{(L_{exons}) * (T_{reads})}$$

where M_{reads} is the total number of short reads that were mapped to exons, L_{exons} is length of exons, and T_{reads} is the total number of short reads. RPKM is used for evaluation of single end reads, which are the type used for study in this thesis. There also exists FPKM, which is similarly derived, and is used for computing values for paired end reads [24, 33, 40].

From this RPKM calculation, a useful formula for finding the RPKM of an expressed isoform can be derived and was obtained from [8]. This formula is used by both GWIE and ChromIE to output the final results of estimated proportions, or expression levels, for each isoform analyzed. The isoform RPKM (iRPKM) is calculated as follows:

$$iRPKM = \frac{NP_i * EV_i * L_e * gRPKM_i}{L_i}$$

where NP_i is the normalized probability of the isoform, EV_i is the expression value of the isoform, L_e is the length of the exons, $gRPKM_i$ is the RPKM value of the gene that the isoform belongs to, and L_i is the length of the isoform.

2.6.1: Calculation of NP_i and EV_i Via EM-Like Algorithm

A brief description of the basic Expectation Maximization (EM) algorithm is provided in section 2.7 and more details of the algorithm use within GWIE and ChromIE is provided in section 4.4.2. Here, the EM-like algorithm E and M steps as described in [8] and used by GWIE and ChromIE are shown below. These steps are used to calculate the normalized probability (NP_i) and expression value or isoform probability (EV_i) of a given isoform as described in the formula given to calculate $iRPKM$ in section 2.6.

$$\text{First is the E step: } z_{r,i}^{(t+1)} = \frac{v_{r,i} * p_i^{(t)}}{\sum_{i=1} v_{r,i} * p_i^{(t)}}, \forall r, i. \text{ Here } z_{r,i}^{(t+1)} \text{ represents the value being}$$

calculated for NP_i for a read, r , and isoform, i , at iteration $(t+1)$ and is done for all reads and isoforms. $v_{r,i}$ represents the initial value given to that read to isoform pair r, i in the calculation.

If the read maps to that isoform, the initial value is $\frac{1}{L_i}$ where L_i is the length of that isoform. If

the read does not map to that isoform, the initial value is zero. $p_i^{(t)}$ represents the mixture proportion of that isoform at iteration t . In GWIE and ChromIE this value is initially set to be

$\frac{1}{E_i}$, where E_i is the total number of expressed isoforms, for all read to isoform pairs, instead of

initializing it with a random value as done in some EM approaches.

Next is the M step: $n_i^{(t+1)} = \sum_{r=1}^N z_{r,i}^{(t+1)}, \forall i$ and $p_i^{(t+1)} = \frac{n_i^{(t+1)}}{N}, \forall i$. $n_i^{(t+1)}$ represents the sum of all

probabilities calculated in the previous E-step of iteration t for all isoforms. $p_i^{(t+1)}$ represents the probability or isoform proportion. The EM algorithm iterates between these E and M-steps until convergence between $p_i^{(t+1)}$ and $p_i^{(t)}$ [8].

2.7: Expectation Maximization (EM) Algorithm

The Expectation Maximization (EM) algorithm appears throughout many disciplines of science, sometimes under differently named variations. These variations are based upon changes within the probabilistic model used to maximize the conditional likelihood of the data. Example fields of EM algorithm application include language processing, probability re-estimation, computer science, and bioinformatics.

There are many deterministic variants of the EM algorithm that attempt to speed up the algorithm by speeding up the convergence rate or simplifying the computations within the algorithm steps. Examples of these variant algorithms include Classification EM (CEM) that maximizes the likelihood of the parameters and the labels [12], Accelerated EM (AEM) that uses scoring optimization in the M step [29], and Parameter-Expanded EM (PX-EM) that uses a “covariance adjustment” based on parameter expansion of the complete data to improve the M step [20]. Further examples include the Baum-Welch algorithm, which is a Generalized Expectation-Maximization (GEM) algorithm that uses the forward-backward algorithm, the inside-outside algorithm, a generalized forward-backward algorithm, and the Expectation Conditional Maximization (ECM) a GEM that replaces the M step with a conditional maximization (CM) step [22]. GEM algorithms are based on increasing the Q-function, $Q(\theta|\theta(t))$ below, instead of maximizing it in each M step [14, 22].

Stochastic EM variants attempt to replace more difficult computations and thus tend less towards getting stuck in the local solution. This is done by replacing the Q-function, which can sometimes be an integral with no closed solution, with an approximated Q-function. The approximation is typically formed using a simulation of the conditional distribution of the missing data. Examples include the Stochastic EM (SEM), Stochastic Approximation type EM (SAEM), and Monte Carlo EM (MCEM) algorithms [14].

Though the EM algorithm can appear under different names with the use of modified parameters, the focus of this section will be the core aspects of the EM algorithm as first described by Dempster, Laird, and Rubin in their 1977 paper, and as these aspects are used within GWIE and ChromIE [11, 28].

The EM algorithm is an iterative method used in statistics to find the maximum likelihood estimators (MLEs) of the parameters in a given statistical model that depends on unobserved (latent) variables. Thus, the EM is typically applied under two scenarios. One is when the data set has missing values, such as the expression levels of isoforms. The other is when it will be too hard to optimize the likelihood function but the function could be simplified by assuming values for the missing parameters [4, 28].

As an iterative method, the EM algorithm creates a sequence of approximate solutions to a class of problems. This sequence is continuously improving. The EM algorithm works by alternating between the E and M steps. The expectation (E) step is based on the current estimate of the parameters ($\theta(t)$ below). It involves the computation of the expected value of the log likelihood (Q below). The maximization (M) step is the computation of parameters that will maximize the expected log likelihood value found in the previous E step. These estimated

parameters are used to find the latent variable distribution for the next E step [4, 5, 7, 11, 28, 32, 44].

Several sources contributed to the following formula and variable uses and explanation including [4, 5, 7, 11, 28, 32, 44]. Though each paper expresses the algorithm in slightly different ways mathematically, they are generally all the same and [32] is perhaps the most clear and straightforward to understand. From [32] we have the following notation for variables used during the derivation of the EM algorithm:

X	Set of observed variables
Z	Set of unknown/unobserved/latent variables
θ	Parameters we want to estimate
$\theta(t)$	Estimate of parameters at iteration t
$\ell(\theta)$	Marginal log-likelihood of observed data: $\log p(x \theta)$
$\log p(x, z \theta)$	Complete log-likelihood (when Z is known)
$q(z x, \theta)$	Arbitrary averaging distribution (can vary to produce EM variants)
$Q(\theta \theta(t))$	Expected complete log-likelihood: $\sum_z q(z x, \theta) \log p(x, z \theta)$
$H(q)$	Entropy of distribution: $q(z x, \theta)$
Θ_{ML}	Maximum Likelihood: $\arg \max \theta \log p(x \theta)$

The mathematical formulas that follow are also taken from [32]. In [32] derivation of the EM algorithm begins with the inequality:

$$\ell(\theta) = \log p(x|\theta) = \log \sum_z p(x, z|\theta) \quad (1)$$

$$= \log \sum_z q(z|x, \theta) \frac{p(x, z|\theta)}{q(z|x, \theta)} \quad (2)$$

$$\geq \sum_z q(z|x, \theta) \log \frac{p(x, z|\theta)}{q(z|x, \theta)} \equiv F(q, \theta) \quad (3)$$

In (2) above, $q(z|x, \theta)$ is an arbitrary averaging distribution over Z . This is used to develop the many EM algorithm variants. The EM algorithm then maximizes the lower bound in (3): $F(q, \theta)$. This leads to the initial E and M steps [32]:

$$\textbf{E-step} : q^{(t+1)} = \arg \max_q F(q, \theta^{(t)}) \quad (4)$$

$$\textbf{M-step} : \theta^{(t+1)} = \arg \max_{\theta} F(q^{(t+1)}, \theta) \quad (5)$$

The next step is to start with any initial parameter $\theta(0)$ and iterate through the E and M steps until $\theta(t)$ converges on the local solution. In equation (4) above $\theta = \theta(t)$ must be fixed and optimized. This leads to: $q^{(t+1)} = p(z|x, \theta(t))$. Equation (5) above uses a fixed q with the knowledge that q depends on $\theta(t)$ to compute the following [32]:

$$\ell(\theta) \geq F(q, \theta) \quad (6)$$

$$= \sum_z q(z|x, \theta) \log \frac{p(x, z|\theta)}{q(z|x, \theta)} \quad (7)$$

$$= \sum_z q(z|x, \theta) \log p(x, z|\theta) - \sum_z q(z|x, \theta) \log q(z|x, \theta) \quad (8)$$

$$= Q(\theta|\theta^{(t)}) + H(q) \quad (9)$$

From equations (6) through (9) above, it shows how maximizing $F(q, \theta)$ is the same as maximizing the expected complete log-likelihood value, $Q(\theta | \theta(t))$. This leads to the full E and M steps as given in [32] as well as other sources including [4, 5, 7, 11, 28, 44]:

$$\textbf{E-step} : \text{Compute } Q(\theta | \theta^{(t)}) = E_{p(z|x, \theta^{(t)})}[\log p(x, z | \theta)] \quad (10)$$

$$\textbf{M-step} : \theta^{(t+1)} = \arg \max_{\theta} E_{p(z|x, \theta^{(t)})}[\log p(x, z | \theta)] \quad (11)$$

Though the author did not show it above, the M step is equivalent to the following:

$$\theta^{(t+1)} = \arg \max_{\theta} Q(\theta | \theta^{(t)})$$

Though useful in many ways, the EM algorithm does have some drawbacks. For instance, there is no guarantee that the sequence of approximated solutions will converge to find the maximum likelihood estimator (MLE). The value found could depend on the initial values the algorithm starts with. There are many ways to try to get around this problem, including “random restart” where different, random initial values are tried for $\theta(t)$. Other typical problems include the slowing of the convergence rate after the first few iterations and the possibility of needing a large number of iterations, as required by the programs GWIE and ChromIE. The former can be handled depending on the problem type by trying some of the EM variations listed previously [11, 28, 32].

Despite these drawbacks, the EM algorithm is useful for many reasons. Each iteration of the E and M steps improves the value of the marginal log likelihood, $\ell(\theta)$. Most importantly it is easier to implement than other methods, such as gradient-based methods, that use derivatives to find maximum likelihood estimators (MLEs). The EM algorithm works best when the amount of missing data and the dimensions of the data are small. These attributes can also help to speed up

the E step of the algorithm [28, 32]. However, this does not apply for most bioinformatics problems, whose data sets can be Gigabytes large in magnitude and the size of missing data can be comparably large.

Chapter 3: Related Work

This chapter focuses on programs that can often be found in journal articles pertaining to the topic of isoform level expression estimation. Though there are many programs that aim to accurately quantify isoform expression, this chapter will concentrate on those programs that are most popular or use some form of EM-like algorithm in order to perform the quantification.

3.1: Enhanced Read Analysis of Gene Expression (ERANGE)

ERANGE, developed in 2008 by Mortazavi et. al [24] is one of the earlier works concerning transcriptome quantification. The authors promoted the benefits of the use of high volume RNA-Seq for transcriptome measure and study and in order to demonstrate its power and usefulness developed ERANGE. In [24] they point out common bioinformatics problems which must be handled: the difficulty in acquiring enough computational power for certain mapping tasks and reads that map to multiple locations in the genome, which they call ‘multireads’. The authors also promoted the use of the RPKM measure to quantify transcript levels in their experiment, accepting transcript levels within $\pm 5\%$ RPKM of the expected value [24].

A typical run of ERANGE begins with mapping unique reads to an expanded genome consisting of the genome and splices, calculating a preliminary RPKM value, reallocating ambiguous reads based on RPKM values, organizing reads that group together but do not map to any known exons to nearby exons or parts of exons, and then calculating the RPKM values for each gene. After this step multireads are handled. Based on the RPKM value of a gene, the probability of the multiread belonging to that gene is calculated. Multireads are then allocated to whichever gene has the greatest probability of having that multiread. A final RPKM value for the gene is then calculated [24].

An important feature of [24] is that the authors point out the importance of using multireads within calculations. Most software at that time simply discarded multireads, which the authors argued results in lower expression or no expression at all for genes with highly similar sequences, such as paralog genes. The authors also found that by including multireads in their study, they were able to increase the transcriptome quantification of their sample from 28% to over 30%. However, only multireads with between two and ten hits were used. Multireads with more than ten occurrences were still discarded from the calculations [24].

3.2: Cufflinks

Cufflinks is a popular program developed by the authors of TopHat and BowTie. In [36] the authors restate the problem of the effect of multireads on transcript assembly and abundance estimation, as well as the reliance on gene annotations that might be incomplete. Cufflinks uses paired end data with the goal of identifying novel transcripts and probabilistically assigning reads, as well as analyzing differential promoter use and differential splicing, two key features that indicate the regulatory behavior within a sample. It was originally designed to analyze transcriptional and splicing regulation, but can be applied to a variety of RNA-Seq problems [36].

For the experiment conducted in [36], cDNA sequence fragments were aligned to the mouse genome via TopHat. When “bundles” of alignments overlap, they are assembled together. This is done via a directed acyclic graph, which constructs paths based on similarity. This step is done to reduce run time and memory usage because bundles are found to only contain fragments from a few genes. Then the abundance of the assembled transcripts is estimated using the probability that observing each fragment is a linear function of the abundance of the transcript it may have originated from. Additionally, fragment lengths are unknown, but Cufflinks uses the

distributions of those lengths to assign reads to isoforms. The abundance is calculated via a maximization function that assigns a likelihood value to all sets of isoform abundances [36].

The results of Cufflinks include recovering over 13,000 known isoforms and over 12,000 new isoforms from known genes. Within these new isoforms, Cufflinks was also able to identify that 58% of them had novel splice junctions. Abundance results are reported using FPKM and reads within 15% of their final FPKM value can be determined to be “moderately expressed” even before all reads are incorporated. Based on their results, the authors were able to conclude that most of the unknown isoforms found were from the transcriptome, meaning the annotation for mouse that was used was incomplete. The experiment consisted of simulated data and the authors state that the high accuracy of Cufflinks is due to its ability to identify novel transcripts as well as perform abundance estimation [36].

One slight drawback of Cufflinks is that in [36] it states that the “model incorporates minimal assumptions about the sequencing experiment”, yet it never elaborates on what these assumptions are exactly. Researchers should know the recommended way to perform sequencing before using a method to yield the best analysis of the reads, if there are certain steps that are assumed to take place.

3.3: IsoformEx

Kim et. al [16] developed IsoformEx to handle the problem of short and alternative exons. IsoformEx employs a “weighting scheme” where non-overlapping exons, called “discriminative exon slices”, are kept whole and carry the greatest weight, and overlapping exons are divided into smaller “slices” in which larger slices are given a greater weight than smaller slices. Similar to the authors of ERANGE [24], the authors of [16] point out the computational difficulty of determining isoform level expression by calculating the counts of all

reads that map to all exons. IsoformEx can be used to find both gene and isoform level expression estimates from RNA-Seq reads.

The algorithm of IsoformEx consists of identifying the isoforms of a gene, add any extra isoforms that might be overlapped to the first set, keep discriminative exon slices or split overlapping exons into smaller slices, compute the RPKM value for each exon slice, and then solve the weighted non-negative least squares using the RPKM values of the exon slices.

In [16] the authors evaluated the accuracy of IsoformEx using both simulated and real data. It is more beneficial to use simulated data to evaluate the accuracy of the program because the true expression levels of the isoforms will be known, whereas with real data they will not. However, it is also useful to use real data when comparing programs. The authors compare their program to RSEM and Cufflinks for simulated data and add qRT-PCR for real data. Reads were aligned to the hg18 version of the human genome from UCSC with the splice junction library from the UCSC transcriptome and Bowtie was used for alignment. The authors report that for simulated data IsoformEx has lower error and higher correlation for estimating transcript expression levels; for real data IsoformEx performs closer to qRT-PCR than Cufflinks. The authors attribute the high performance of IsoformEx to its unique weighting scheme allowing for more accurate results [16].

3.4: RNA-Seq by Expectation Maximization (RSEM)

RSEM is a software package for both gene and isoform quantification. In this paper too, the authors restate the major challenge of handling multireads. They also find that single-end reads provide more accurate estimates for gene level abundance, though RSEM can work on paired end reads as well. A key feature of RSEM is that it does not require a reference genome, users can alternatively provide a set of reference transcript sequences. These sequences are

similar to those produced by de novo transcriptome assemblers. The lack of need for a reference genome allows transcriptome quantification to be performed for animals who have yet to be sequenced [17].

Only two scripts are required to run RSEM, one reason why the authors boast the user-friendliness of this program. Step one is the generation and pre-processing of transcript sequences. Step two is the estimation of abundances and creation of credibility intervals based on read alignments via Bowtie to the reference transcripts. The authors state that aligning to a set of reference transcripts instead of genome is less complicated, applicable to species with usable transcriptomes that may not have sequenced genomes, and is much faster because the transcriptome is smaller than the genome. However, like other command line based programs, RSEM has several options that the user can choose to describe input or vary program output in order to increase performance [17].

RSEM outputs one file for isoform level estimates and another for gene level estimates. Instead of the typical RPKM value, RSEM either uses the fraction of transcripts produced by an isoform or gene or multiplies that value by 10^6 to get transcripts per million (TPM). According to the authors, TPM is preferable to using RPKM or FPKM because TPM is independent of the mean expressed transcript length, making it more comparable across samples. In addition to the two primary files RSEM can also display RNA-Seq data as tracks in genome browsers, output alignments in BAM format weighted by the probability that it is the true alignment, convert the BAM file to WIG, and display the sequence model. RSEM is highly functional and can even use a Bayesian version to produce confidence intervals and perform read simulation [17].

RSEM results in better estimation when given multiread alignments and without the use of selecting only alignments within a certain range of each read. The goal is for RSEM to choose

the correct alignments instead of the aligner. The only requirement the authors suggest is to have aligners suppress reporting reads with a large number of valid alignments. This is used to help reduce the running time and memory usage of RSEM [17].

RSEM was compared to IsoEM, Cufflinks, rQuant, and the previous version of RSEM for estimates of gene abundance, global isoform abundance, and gene level isoform abundance. For single and paired end reads, RSEM and IsoEM had greater performance than Cufflinks and rQuant. The authors suggest that this is because the latter programs require genome alignment, not alignment to a transcript set, as well as the way the programs handle multireads. Cufflinks fractionally assigns them, while rQuant does not specify if or how they are handled. RSEM and IsoEM also performed similarly for paired end data, but RSEM was found to be slightly better with single end data. For this analysis, both simulated and real data were used, for the same reasons as stated in [16]. A comparison on running time and memory usage was also performed. RSEM requires time and memory linearly to the number of read alignments. It was found that IsoEM, the only Java based program, required the largest amount of memory, up to 14 GB, whereas other programs required only one to two GB [17].

Despite the thoroughness of this approach, it does have a few drawbacks. Despite the advantages of aligning to a transcript set instead of the genome, doing so depends heavily on the quality of the transcript set. As shown in [43], popular de novo assemblers are only around 60% accurate. This means that a large portion of the transcriptome will be missing for RSEM to work with. Second there are several modifications applied to their M step of the EM algorithm depending on proportions of fragments from isoforms and the number of unfiltered fragments. Because of the dependency on transcript reference sets, these numbers could be inaccurate. Also, instead of iterating down to a low cutoff, the EM algorithm stops when all estimates greater than

10^{-7} have had changes less than 10^{-3} . Lastly, multireads with over 200 alignments are discarded from the calculation [17].

3.5: IsoEM

IsoEM uses a novel EM algorithm in addition to base quality scores, strand information, and read pairing information to calculate both gene and isoform expression levels, what they refer to as “frequencies”. It assumes a known set of isoforms and also models insert size distribution. The authors point out the pervasive problem of multireads and favor what is sometimes referred to as the “rescue” method of [24] which fractionally allocates multireads, similar to Cufflinks [36] over multiread discarding [25].

In order to increase estimation accuracy, insert sizes are used, similar to Cufflinks to figure out the most likely originating exon. This is also combined with base quality scores to probabilistically align reads during their E step. IsoEM supports short reads the best [25].

The steps of using IsoEM begin with read alignment via Bowtie. Then radix sort is used to match isoforms to reads via the following algorithm: all reads are collected, their coordinates are sorted, and then a line sweep technique matches the read to the isoform. The authors state this step helps speed up the program because it allows all reads to be processed at once, which is a major bottleneck for the program. For multireads, a “weight” is computed for assigning that read to a particular isoform. This computation takes into account probability and base quality, as well as fragment length. If single end, paired end, or read direction information is available, this can also be used to more accurately calculate the weight in order to determine the best isoform and read match [25].

For the initialization of their EM algorithm, expression levels are initialized randomly. IsoEM also features two key optimizations. First is that the reads and isoforms are divided into

“compatibility components” which consist of all isoforms connected to all of their possible reads. The authors state that the frequencies, or expression levels, of different components do not interfere with each other, thus they can be analyzed separately. Therefore the EM algorithm of IsoEM is performed on each component, rather than all reads and isoforms as a whole. Second, the components are split into “equivalence classes” which eliminates copies of multiple equivalent reads. Reads are said to be equivalent if they belong to the same isoform and their weights are proportional. The major benefit of this optimization is that it reduces the amount of reads held in memory. Removal is achieved through a union-find algorithm based on trees that is in log time and uses hashing. A slight drawback of this approach is that since all reads are maintained in memory to perform removal, if there is too large a number of reads, collisions are more likely to occur which result in poor performance [25].

For their experiment, IsoEM was run on single end, simulated data and compared to ERANGE, Cufflinks, and the original version of RSEM. IsoEM outperformed all competitors, as also supported by the findings of [17]. The authors also mention that at this time, the latest version of RSEM was released and they found it to be comparable to IsoEM, again as supported by [17]. When run on real data, IsoEM has slightly lower estimation accuracy than for simulated data, but still outperforms other programs. The authors also found that using strand information for paired end reads resulted in no significant gain when estimating genome wide isoform levels [25].

Similar to GWIE and ChromIE, IsoEM is Java based and requires the most amount of memory. IsoEM requires no more than 16 GB of memory, but more memory does result in faster computation time. Also, as the number of reads increases, run time also increases [25].

3.6: Read Assignment via EM (RAEM)

In [8] RAEM was developed to calculate isoform expression levels from RNA-Seq data in order to show the power of using an isoform level approach over previously popular gene based approaches for transcriptome quantification. It was one of the first papers to promote this approach for microRNA target prediction. Through analysis of alternatively spliced isoforms, exon-exon junctions, isoform level down regulation, and seed enrichment, microRNA targeting regions were identified as well as novel isoforms [8].

Experiments were conducted using both simulated data from Flux Simulator [10] and real data. For the experiment, short reads were aligned to the reference genome and the results were saved in SAM format. The SAM file was then processed by SAMMate [41] to produce gene RPKM values. After alignment, isoforms were quantified and then used in combination with differential splicing and seed enrichment to predict targets. Aligning junction reads via TopHat also allowed for novel isoform prediction and quantification. RAEM was able to find novel isoforms and predict more targets than gene based approaches. With isoform down regulation it was also able to identify non-dominant targets that are overlooked by gene based approaches [8].

RAEM consists of an iterative EM-like algorithm that is applied to quantify the transcriptome at the isoform level. Details of the E and M steps of the algorithm are discussed in sections 2.6 and 2.6.1. The algorithm works on a matrix of short reads paired with the isoform that they can be aligned to, similar to the matrices of GWIE and ChromIE. It incorporates only unique reads and runs the EM algorithm for every gene [8].

GWIE and ChromIE were developed to be more global versions of this gene level EM based approach and to include the estimation of multireads along with unique reads. Whereas [8] was concerned with microRNA target prediction, GWIE and ChromIE were designed for

isoform expression quantification alone. They were not used with additional techniques such as seed enrichment or for target prediction, but the results produced by the programs could be used in this manner.

Chapter 4: Software

4.1: Introduction

This chapter covers the design and implementation of the two programs, Genome Wide Isoform Estimation (GWIE), specifically its incremental version (iGWIE) and Chromosomal Isoform Estimation (ChromIE), designed for this thesis. Since both programs share similar input files, file processing classes, data structures, and supporting classes, such as those that represent biological entities (such as genes, mRNAs, exons), this will be covered once but is valid for both programs.

The main variation between the two programs is the way the EM algorithm is run on the isoform matrices of each program. This differentiation between the set up and use of the matrices will be covered in more detail in section 4.4.

4.2: Supporting Classes

These are simple objects that contain information necessary for MILE to form its data structures and perform the EM algorithm. Classes Gene, mRNA, and Exon contain the information relevant to the biological object they represent as well as minor functionality, such as set methods. These classes are represented as UML object diagrams in Figure 3 below section 4.2.3. Classes EmInfo and ExonReadLocationFilter are also used by the MILE class and are shown in Figure 4 below section 4.2.5.

4.2.1: Gene

Genes are represented by class Gene and include descriptive information such as its chromosome, name, ID from the GFF file, strand, starting and ending locations, exon length, RPKM value, mRNAs, and Exons. Genes support the functionality to add an mRNA to its list of mRNAs or an Exon to its list of Exons. Class Gene also supports setting its RPKM value.

4.2.2: mRNA

mRNAs are represented by class mRNA. mRNA contains ID, chromosome, strand, parent gene and its location, length, number of exons, expression value, and a list of its Exons. Class mRNA supports the functionality to set its length, expression value, exon proportion and count, as well as adding an Exon to its list.

4.2.3: Exon

Exons are represented by class Exon. This class includes ID, parent gene and mRNA, location start and end, expression value, proportion, chromosome, and minimum and maximum value. The only functionality of class Exon is to set its parent mRNA.

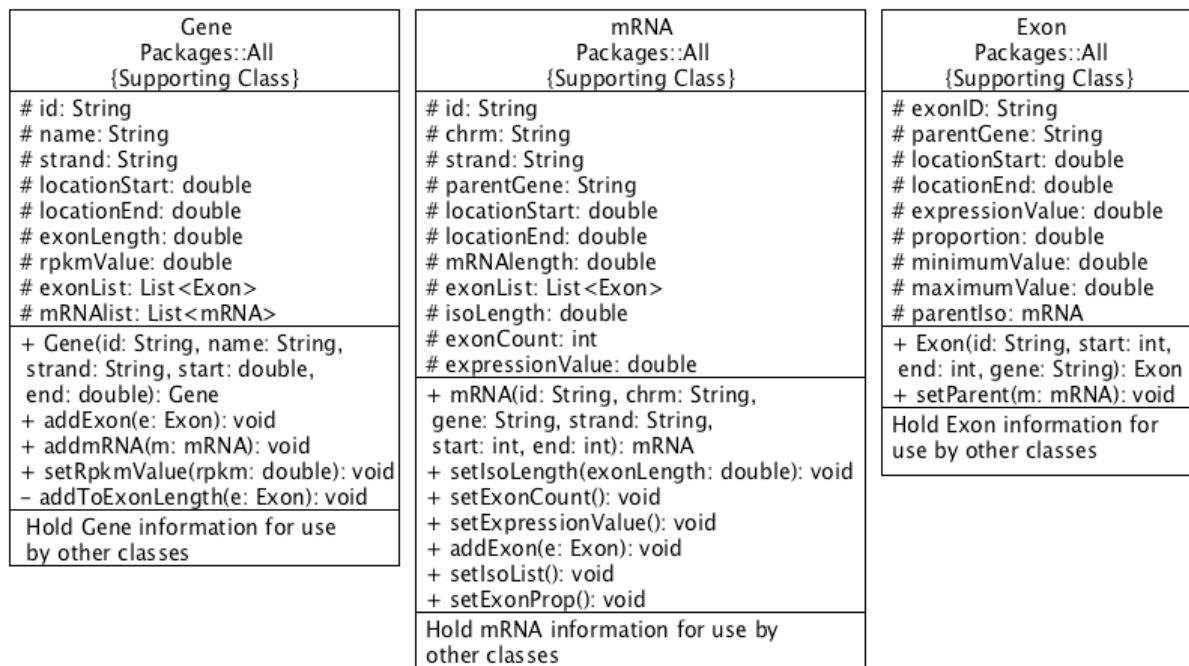


Figure 3. UML Diagrams of Classes Gene, mRNA, and Exon. (Left) UML diagram depicting class Gene. (Center) UML diagram depicting class mRNA. (Right) UML diagram depicting class Exon. The first block contains the class name and indicates that this class is a member of all packages (GWIE, iGWIE, and ChromIE). The second block contains class attributes. The third block contains class methods. The final block contains a brief description of the responsibilities of the class. These UML diagrams were created using UMLet, a free software package [37].

4.2.4: EmInfo

This class contains information obtained from parsing of the SAM, or reads, file. It holds a read ID and lists which have the Genes and mRNAs that the read maps to. MILE packages the read ID and its Exon into this class before adding this object to the appropriate isoform matrix. Once the read ID has been added to the matrix, whenever the read ID is encountered again, its lists are extended with the new Gene and mRNA information.

4.2.5: ExonReadLocationFilter

This class contains an array of List<Exon>. The size, or number of slots, and read length, or increment, to use can be modified within this class. The increment represents the exon length range that each slot will cover. For example, if the increment is 1000, this means that slot zero will range from 0 – 999, slot one from 1000 – 1999, slot two from 2000 – 2999, and so on. This means that if an exon has a length from 450 to 580, then it would be placed into slot zero. As another example, if an exon were to have a length from 1950 to 2100 this exon would be placed into both slots one and two.

The GffProcessor creates exons and uses the addExon(Exon e) method of this class to allow the filter to place the given exon into the correct slots. In the event of exon overlap across slots, the method getMappedExons(int startPosition, int endPosition) will return a list of Exons, excluding duplicates, which map to the read currently being processed in the ReadsFileProcessor. The ReadsFileProcessor calls this method to set the expression value of the returned Exons and their parent mRNAs and then send the read and its exon to MILE to be added to the appropriate isoform matrix.

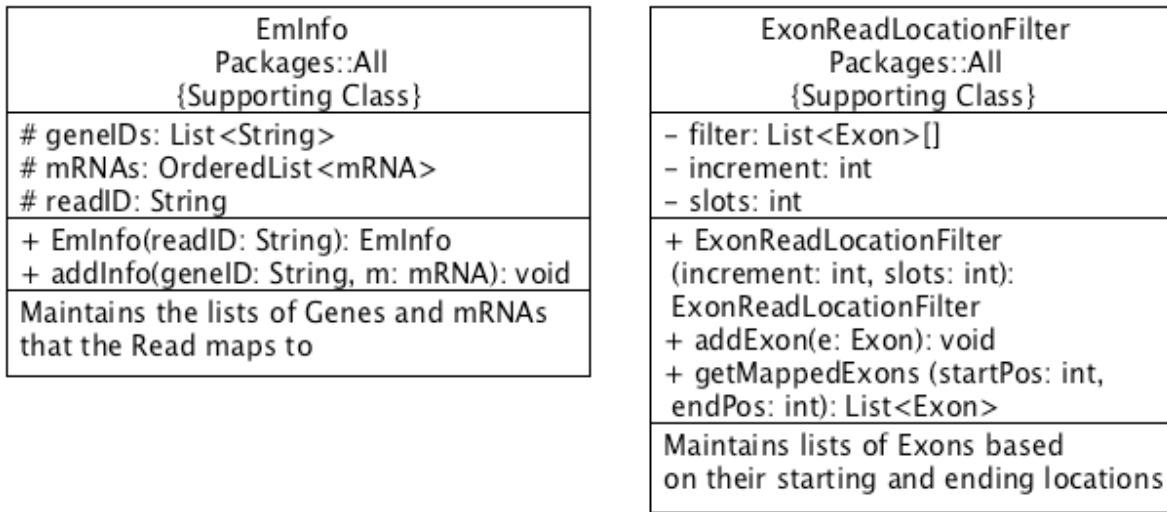


Figure 4. UML Diagrams of Classes EmInfo and ExonReadLocationFilter. (Left) UML diagram depicting class EmInfo. (Right) UML diagram depicting class ExonReadLocationFilter. The first block contains the class name and indicates that this class is a member of all packages (GWIE, iGWIE, and ChromIE). The second block contains class attributes. The third block contains class methods. The final block contains a brief description of the responsibilities of the class. These UML diagrams were created using UMLet, a free software package [37].

4.3: Input Files and File Processing Classes

Both programs require three input files to perform the estimation. Each input file has a Java class responsible for reading the file information and using it to set up the data structures used by the class responsible for conducting the isoform estimation, MultireadIsoformLevelEstimator (MILE).

First is the GFF file describing the chromosomes, genes, mRNAs, and exons as well as their relationships to each other and locations. The GFF file is read and divided into data structures by the GffProcessor class. Second is the SAM file consisting of all of the reads and their locations. This file is processed by the ReadsFileProcessor class. Third is the RPKM file, which consists of each gene and its RPKM value and is read by the RpkmFileReader class. These three classes are depicted in Figure 5 below section 4.3.3.

4.3.1: GFF File and GffProcessor

The GFF or annotation file consists of gene, mRNA, and exon information that describe the genetic organization of the species. For this thesis, the GFF file Homo Sapiens GRCh3757_Protein Coding was used. This file was obtained from: <ftp://ftp.ensembl.org/pub/release-62/gtf/> [9]. The basic format of the GFF file is shown below:

```
chr11 protein_coding gene 127926 139099 . - . ID=ENSG00000230724;Name=AC069287.3
chr11 protein_coding mRNA 130207 131297 . - . ID=ENST00000427071;Name=AC069287.3;Parent=ENSG00000230724
chr11 protein_coding exon 131120 131297 . - . ID=ENST00000427071.1;Name=AC069287.3;Parent=ENST00000427071
```

Each line of the GFF file contains the starting and ending location of a gene, mRNA, or exon, as well as its ID, name, and chromosome. For mRNAs, the parent is the gene; for exons, the parent is the mRNA.

The GffProcessor class reads in the GFF file and for each line breaks it apart to initialize the appropriate object and assign it to the correct data structure. For lines containing a gene, a new Gene is created with the information taken from the line including ID, name, starting location, and ending location, and is then added to the collection of Genes to be used in MultireadIsoformLevelEstimator. For lines containing an mRNA, its information is used to initialize a new mRNA object. Then the mRNA is added to its parent Gene. For lines containing an exon, a new Exon instance is created with the information taken from the line. This Exon is added both to its parent Gene's and parent mRNA's respective list of Exons as well as the ExonReadLocationFilter. When an Exon is added to its parent Gene, that Gene's total exon length is updated with the length of the added Exon. Similarly, when added to its parent mRNA, the mRNA's total isoform length is updated. These lengths are maintained because they are required for isoform proportion calculation later in the EM algorithm.

4.3.2: SAM File and ReadsFileProcessor

The SAM file consists of reads and the exon location that they can be mapped to. This location represents the start of the read, which can vary in length. For this thesis, simulated reads were generated using Flux Simulator, a free program that simulates RNA-Seq data produced by an Illumina Genome Analyzer [10]. These reads were 50 base pairs (bp) in length and were aligned to the reference genome using TopHat to produce a SAM file. The basic format of the SAM file is shown in an example line below:

```
12:9570287-9600824C:ENST00000290818:6257:3932:2597:2792:2743:2792    16    chr1    13005    0    50M
*          0          0          AGGCTCTGGTGGAGAACCTGTGCATGAAGGCCGTCAACCAGTCCATAGGC
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII    NM:i:1 NH:i:10 CC:Z:chr12    CP:i:92559.
```

The SAM file is tab delimited and consists of millions of lines, where each line represents a read and its alignment information. The necessary information obtained from this line consists of the read ID and starting position of the mapped location. In the above line the first string is broken down to get the read ID (ENST00000290818) and starting position (13005).

The ReadsFileProcessor reads in each line from the input file and breaks it apart to get the read ID and starting position of the read as described above. Since reads are fifty bases long in the simulated reads used for this thesis, for each read, the exons that were mapped to that starting position plus fifty bases are retrieved from the ExonReadLocationFilter and traversed for a match. When an exon is located with a matching starting and ending position, its expression value is both incremented and set for its parent mRNA. The corresponding read of that exon is then added to the isoform matrix of GWIE or the appropriate chromosome's isoform matrix of ChromIE.

4.3.3: RPKM File and RpkmFileReader

The RPKM file consists of gene names and their respective RPKM values. An example of the RPKM file format is:

```
MARCH1      0
MARCH2      4.514773398
MARCH3      20.25911846
MARCH4      0
MARCH5      3.493217677
MARCH6      17.88849396
```

The RpkmFileReader reads in each line from the RPKM file and assigns the RPKM value given in the file to its Gene. The genes and example RPKM values used for this thesis were obtained using SAMMate [41].

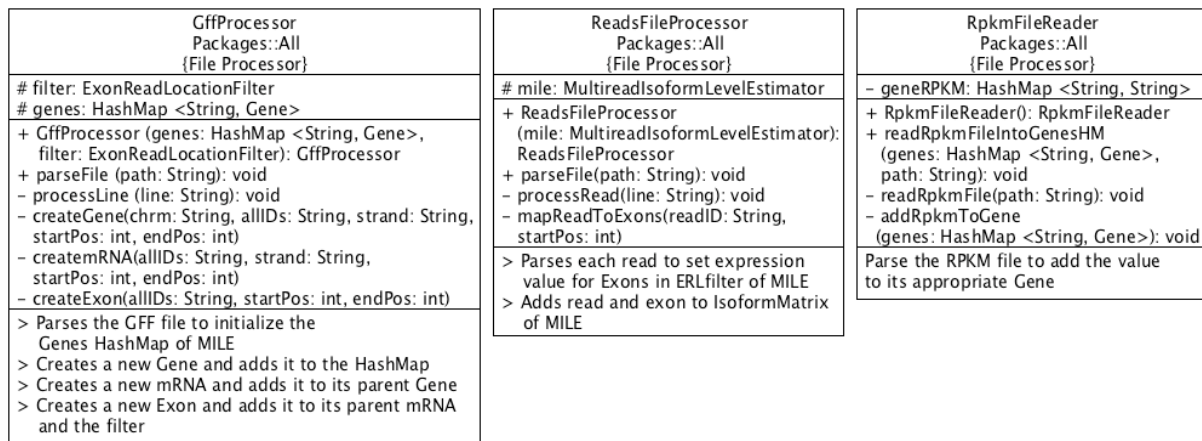


Figure 5. UML Diagrams of File Processing Classes. (Left) UML diagram depicting class GffProcessor. (Center) UML diagram depicting class ReadsFileProcessor. (Right) UML diagram depicting class RpkmFileReader. The first block contains the class name and indicates that this class is a member of all packages (GWIE, iGWIE, and ChromIE). The second block contains class attributes. The third block contains class methods. The final block contains a brief description of the responsibilities of the class. These UML diagrams were created using UMLet, a free software package [37].

4.4: Main Class: MultireadIsoformLevelEstimator (MILE)

The class MultireadIsoformLevelEstimator is responsible for calling the file processing classes to initialize its data structures, which are then used during the conduction of the EM algorithm. This class is depicted below in Figure 6, which shows the similarities and differences between the implementation of MILE for GWIE versus ChromIE.

MultireadIsoformLevelEstimator Package::GWIE and iGWIE	MultireadIsoformLevelEstimator Package::ChromIE
<pre># genes: HashMap<String, Gene> # expressedIsos: OrderedList<mRNA> # filter: ExonReadLocationFilter # isoformMatrix: HashMap<String, EmlInfo> - oldProb: ArrayList<Double> - newProb: ArrayList<Double> - isoProp: ArrayList<Double> - columnCalc: ArrayList<Double> - rowCalc: ArrayList<Double> - rpkmFile: String - gffFile: String - readsFile: String</pre>	<pre># genes: HashMap<String, Gene> # expressedIsos: OrderedList<mRNA> # filter: ExonReadLocationFilter # isoMatrices: HashMap<String, EmlInfo>[] - oldProb: ArrayList<Double> - newProb: ArrayList<Double> - isoProp: ArrayList<Double> - columnCalc: ArrayList<Double> - rowCalc: ArrayList<Double> - rpkmFile: String - gffFile: String - readsFile: String</pre>
<pre>+ MultireadIsoformLevelEstimator (rpkmFile: String, gffFile: String, readsFile: String): MultireadIsoformLevelEstimator + execute(): void - conductInitialization(): void - conductEstimation(): void - initializeArrays(isoCount: double): void - clearCalcArrays(): void - eStep(currentReadIsos: OrderedList<mRNA>): double - mStep(rowTotal: double): void - calcDiff(): double - calcIsoProp(): void - printResults(): void - totallIsos(): void + addToMatrix(readID: String, exon: Exon): void</pre>	<pre>+ MultireadIsoformLevelEstimator (rpkmFile: String, gffFile: String, readsFile: String): MultireadIsoformLevelEstimator + execute(): void - conductInitialization(): void - totallIsos(currentMatrix: HashMap<String, EmlInfo>): void - conductEstimation (isoMatrix: HashMap<String, EmlInfo>, chromNum: int): void - initializeArrays(isoCount: double): void - clearCalcArrays(): void - eStep(currentReadIsos: OrderedList<mRNA>): double - mStep(rowTotal: double): void - calcDiff(): double - calcIsoProp(): void - printResults(chromNum: int): void + addToMatrix(readID: String, e: Exon): void - getChromosome(chrom: String): int - addTolIsoMatrix(chromNum: int, readID: String, e: Exon): void - chromIsValid(chrom: String): boolean</pre>
<pre>> Use supporting classes to initialize data structures > Run EM algorithm on Genome Wide Isoform Matrix > Print result files: Once at the end (GWIE) After every EM iteration (iGWIE)</pre>	<pre>> Use supporting classes to initialize data structures > Run EM algorithm on each Chromosome's Isoform Matrix > Print results per chromosome</pre>

Figure 6. UML Diagrams of MILE. (Left) UML diagram depicting MILE as used by GWIE and iGWIE. (Right) UML diagram depicting MILE as used by ChromIE. The first block contains the class name and indicates which package the class belongs to. The second block contains class attributes. The third block contains class methods. The final block contains a brief description of the responsibilities of the class. These UML diagrams were created using UMLet, a free software package [37].

4.4.1: Step 1: Data Structure Initialization

The first step within the program is to initialize a HashMap of Genes via the GffProcessor. These Genes are arranged with lists of their mRNAs, which have lists of their Exons. Next, the RPKMFileReader assigns the RPKM value to each Gene. Finally, the ReadsFileProcessor assigns reads to their Exons and then an isoform matrix. This is where the two programs begin to differ. For GWIE, MILE maintains one isoform matrix of all isoforms and all reads. For ChromIE, this class maintains one isoform matrix for each chromosome, which holds all of the reads for that chromosome.

4.4.2: Step 2: Isoform Estimation using the EM Algorithm

This step is run the same for both programs with slight differences with what it is run on and the output that is produced. For GWIE, the isoform matrix consists of all reads, which is a little over fifteen million. The EM algorithm is run repeatedly on this one isoform matrix until a cutoff value is reached. This program does not produce output until the cutoff value is reached.

Its second version, iGWIE, is the same in the sense that it uses only one isoform matrix of all reads. The only difference is that incremental output is produced from each iteration of the EM algorithm. This allows the user to watch the convergence of the E and M steps and possibly terminate the program before the cutoff is reached, if the difference value has become significantly low enough for their research purposes.

For ChromIE, instead of one isoform matrix containing all of the reads, there is one isoform matrix for each chromosome. The EM algorithm is then run on each isoform matrix until the same cutoff value is reached. The final result for each chromosome is printed once the cutoff is reached.

As described in section 2.6, the following variables typical to calculating an EM algorithm are used in the software:

x – *Set of observed variables*. For the code, this represents the number of reads to each isoform.

z – *Set of unknown variables*. For the code, this is the proportion of each isoform. Since it is unknown, every value is initialized to be one divided by the total number of expressed mRNAs.

θ – *Parameters we want to estimate*. For the code, this is the expression level of each isoform.

$\theta(t)$ – *Estimate of parameters at iteration t* . For the code, this value decreases over iterations.

$q(z|x, \theta)$ – *Arbitrary averaging distribution*. For the code, all of the reads per isoform are simply added up and divided by the total number of reads, so it is just a simple average.

The remainder of this section describes the application of the EM-like algorithm as taken by GWIE, iGWIE, and ChromIE. All three programs use the same code for calculating the E and M steps. However, the data that is used in this calculation varies between GWIE and iGWIE when compared to ChromIE.

The GWIE approach is global, in the sense that it looks at all reads to all expressed isoforms of all genes. The ChromIE approach is at the chromosomal level, meaning it handles all of the reads of each chromosome separately. Expressed isoforms are those that have at least one read mapped to them; other isoforms are discarded from the calculations. Arrays are maintained to hold the values of the previous step's probability, the new probability, the isoform proportions, and intermediate column and row calculations.

The `conductEstimation()` method of GWIE and the `conductEstimation(HashMap <String, EmInfo> isoformMatrix, int chromosomeNumber)` method of ChromIE begin by initializing the previous probabilities to be one divided by the total number of expressed mRNAs (Z). The E-like step is as follows: for every read, if the read can be mapped to the isoform then its previous

probability is multiplied by its expression value (one divided by the length of the isoform) and this calculation is both added to a running sum and placed into an array for later use. If the read does not map to that isoform, then a zero is added into the calculation.

Next is the M-like step where the values calculated above are normalized. Every calculation is divided by the sum of all calculations. This normalized value is then saved. It is later divided by the total number of reads and used to replace the value of the previous probability. Then the difference is calculated as the previous probability minus the newer probability. These steps are repeated until the difference converges to a chosen cutoff. The cutoff value used is an arbitrarily small, positive number (0.00001).

Once the difference reaches the cutoff, the isoform proportions are calculated as the previous probability of the isoform divided by its length. The code also sums all of the isoform proportions and saves the running total. Finally, for each isoform the following values are printed to the output file: name, parent gene, proportion of reads that go to the isoform (older probability), isoform proportion (the isoform proportion found previously divided by the running total), gene length and RPKM, and isoform length and RPKM.

Thus this algorithm represents a simple EM-like algorithm. It uses a basic average instead of optional averages that can be used to construct EM variants. Instead of using log functions, the code continues to iterate and reduce the proportion/probability down until the lower values are minimized so low that it leaves the isoform expression value that is the strongest for that read as the maximum value. Figure 7 depicts a simple example to clarify the E and M steps.

1	All Reads	All Isoforms	
		Isoform 1	Isoform 2
	Read 1		
	Read 2		
	Read 3		
	Read 4		
	Read 5		

2	All Reads	All Isoforms	
		Isoform 1	Isoform 2
	Read 1	$1/L_{iso1}$	$1/L_{iso2}$
	Read 2	0	1/20
	Read 3	1/10	0
	Read 4	0	1/20
	Read 5	1/10	1/20

3	All Reads	All Isoforms	
		Isoform 1	Isoform 2
	Read 1	$1/L_{iso1} * P_{init}$	$1/L_{iso2} * P_{init}$
	Read 2	0	$1/20 * 1/2 = 1/40$
	Read 3	$1/10 * 1/2 = 1/20$	0
	Read 4	0	1/40
	Read 5	1/20	1/40

4	All Reads	All Isoforms	
		Isoform 1	Isoform 2
	Read 1	$1/L_{iso1} * 0.5$	$1/L_{iso2} * 0.5$
		$1/L_{iso1} * 0.5 + 1/L_{iso2} * 0.5$	$1/L_{iso1} * 0.5 + 1/L_{iso2} * 0.5$
	Read 2	0	$1/L_{iso2} * 0.5$
			$0 + 1/L_{iso2} * 0.5$
	Read 3	$1/L_{iso1} * 0.5$	0
		$1/L_{iso1} * 0.5 + 0$	
	Read 4	0	$(1/40)/(1/40) = 1$
	Read 5	$0.05/0.05+0.025 = 0.667$	$0.025/0.05 + 0.025 = 0.333$

5	All Reads	All Isoforms	
		Isoform 1	Isoform 2
	Read 1	Normalized Value = 0.667	0.333
	Read 2	0	1
	Read 3	1	0
	Read 4	0	1
	Read 5	0.667	0.333
	Column Total/5:	$2.334/5 = 0.4668$	$2.666/5 = 0.5332$

Figure 7. E and M Steps of GWIE and ChromIE. The above figures depict a simple example illustrating the isoform matrix set up for GWIE and ChromIE and how the E and M steps are applied. If this was an isoform matrix in GWIE, the left most column labeled “All Reads” would contain all possible reads from the given SAM file; if it were ChromIE these would be all the reads that map to exons within a single chromosome. For GWIE, the rightmost column labeled “All Isoforms” would contain all possible mRNAs in the given GFF file; for ChromIE, this would be all mRNAs within a single chromosome. (1) shows the initial matrix set up of reads to possible isoforms. (2) shows the initialization phase. If a read maps to the exons of an isoform, it is given the initial value of $\frac{1}{L_{iso\#}}$, where $L_{iso\#}$ is the length of that given isoform. If the read does not map to that isoform, the initial value is zero. In the example, the length of Isoform 1 is 10 and the length of Isoform 2 is 20. (3) shows the first round of calculating probabilities. Wherever a read maps to an isoform, the initial value assigned in (2) is multiplied by the initial probability, P_{init} . For GWIE and ChromIE, the initial probability is one divided by the total number of expressed mRNAs (globally and for the chromosome respectively). In the above example, there are only two isoforms, so $P_{init} = \frac{1}{2}$. (4) shows the normalization of the values. The value calculated in (3) is divided by the sum of all of the values calculated in (3) for all isoforms that match to that read. For instance, Read 1 can map to Isoform 1 or Isoform 2, so it must be divided by the sum of their probabilities from (3). However, Read 2 only maps to Isoform 2, so it must be divided by its own value from (3), since the value from Isoform 1 is zero. (5) shows the normalized value results from the calculations done in (4). These values will be used to replace P_{init} in (3) and then steps (3) and (4) will be repeated until convergence.

4.5: Critique and Comparison of Programs

Both GWIE and ChromIE have their pros and cons. Despite the negative aspects of the programs, they still accomplish the goal of isoform level expression estimation.

4.5.1: GWIE: Pros and Cons

GWIE and iGWIE allows for all reads and all isoforms to be considered during estimation together. This achieves the global, genome wide goal of isoform estimation. Based on intermediate results, it is hypothesized that this approach will result in highly accurate expression levels.

Due to the large amount of reads to be analyzed, GWIE requires a large amount of memory, which some users may not have at their disposal. The use of all reads together also leads to the problem that once the program is started it must run to completion; it cannot be stopped at some random point and then restarted with the previous calculations.

An even greater drawback is the amount of time that it takes GWIE to run the EM algorithm until the cutoff is reached. For users with a short time frame, this program would not be usable. In a trial run, it took GWIE over five months to reach the halfway mark towards the cutoff. Due to power outages and the inability to start the program from any point, final results were not acquired. This led to the development of iGWIE, so that at least incremental results could be tracked and analyzed.

Both GWIE and iGWIE share the same underlying problem. The difference value used to iterate down towards the cutoff starts at approximately one to several decimal places. For example, for GWIE the beginning difference value was 1.2008002045216042, while for ChromIE running on chromosome X it was 1.1686005892854419 and for chromosome one it was 1.131445021781292. Due to the millions of reads that must be calculated in the EM

algorithm with every single iteration, this difference value decreases on average by roughly 0.2 with each iteration. Hence, it takes the programs several months to achieve complete results.

4.5.2: ChromIE: Pros and Cons

ChromIE was developed to decrease run time while still resulting in estimation results with greater coverage than a gene based approach. Though it cannot achieve a global, genome wide accuracy comparable to GWIE, ChromIE is comparably accurate to gene-based approaches. This is because all the reads for all genes of a chromosome can be considered together but there is no true way to estimate which approach is superior since ChromIE considers more values than gene based approaches. Thus it has a broader range than a gene-by-gene estimation but is still not as accurate as GWIE after many iterations.

Another advantage of ChromIE is that if the program were to unexpectedly be stopped, it could be restarted and with minor alterations can ignore chromosomes that have already been analyzed. Thus, the program does not have to run from start to finish uninterrupted in order to produce results.

Like GWIE, however, ChromIE requires a lot of memory and even requires a little more than GWIE due to having multiple isoform matrices instead of just one. Even though its run time is far shorter than GWIE, it still takes approximately one week per chromosome to run, which might be too long for most users. Despite this time frame, it is still quicker than GWIE and capable of producing results as accurate as both GWIE and a gene level approach.

An interesting aspect of ChromIE is that the difference value used to iterate down towards the cutoff value starts at a similar level as with GWIE. For example, for chromosome X, the initial difference value is 1.1686005892854419. Similarly, for each iteration, the difference value decreases by about 0.2 each time. Since ChromIE considers far fewer reads and isoforms,

which are represented by much smaller isoform matrices than that of GWIE's singular, large matrix, ChromIE is able to iterate down at a rate much faster than GWIE.

Chapter 5: Results

Due to the large memory requirement of GWIE and the hundreds of iterations required while analyzing over 90,000 isoforms and approximately fifteen million reads, it takes GWIE months to achieve results. Unfortunately, as the result of several power outages, the halfway results of the original GWIE run were lost. In response to this iGWIE was run to gather intermediate incremental results, i.e. estimates produced at every iteration. Again, due to power outages, the highest level attained was that of the seventh iteration. iGWIE was stopped to run ChromIE, in the hopes of acquiring more results for this thesis. The following sections describe the accuracy of the approaches as best as can be determined with the incomplete results gathered thus far.

To test the accuracy of GWIE and ChromIE, approximately fifteen million single-end short reads were created using Flux Simulator [10]. These reads were then aligned to the reference genome using TopHat with the human index built by Bowtie. This produced a SAM file indicating multireads and exon locations. This SAM file was then used with an annotation or GFF file to build the data structures in MILE. Within MILE the EM-like algorithm was run to determine isoform expression levels.

Since the programs were run using simulated data, the actual values were available for analysis. In addition, these results were compared to the results gathered from running [8] on the same sample data. However, since [8] runs the EM algorithm on a gene level and with only unique reads that go to one gene only (multireads are discarded), it does not fully encompass the estimates of all possible reads.

To determine the differences between the actual values, values computed from iGWIE and ChromIE, and the values found by RAEM, a simple analyzer program (MILEanalyzer) was

written. This program computed the differences between the real values for reads and copy proportion and the estimates computed by iGWIE, ChromIE, and RAEM.

5.1: iGWIE Results

Figure 8 below displays the average difference values between the real values and programs using the seventh iteration output of iGWIE. Figures 9 and 10 show the predicted proportion values from ChromIE compared against the expected real values.

Real Copy vs. iGWIE Estimated Copy	0.14232803215316633
Real Read vs. iGWIE Estimated Read	0.14232676369188355
Real Read vs. RAEM Estimated Read	0.07359557747788822
Real Copy vs. RAEM Estimated Copy	0.07394645100352958
RAEM Estimated Read vs. iGWIE Estimated Read	0.1383764910279287
RAEM Estimated Copy vs. iGWIE Estimated Copy	0.13837679161917676

Figure 8. iGWIE Intermediate Results. Average difference between real values and estimated values. These are based on the difference values found after only seven iterations of the EM algorithm. Read is the reads proportion; copy is the isoform proportion.

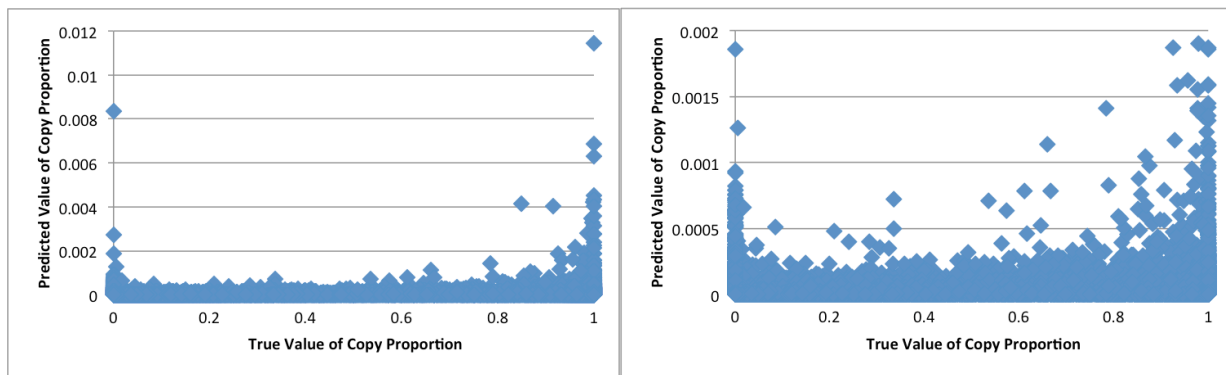


Figure 9. iGWIE Predicted Copy Proportions vs. Real Copy Proportions. These plots depict the predicted value for copy proportion compared against the true value for each isoform. As shown, most values are close to the true value with a few outliers concentrated around the boundaries. The left plot shows the results for all isoforms; the right plot shows a closer view of the left plot, excluding outliers beyond 0.002.

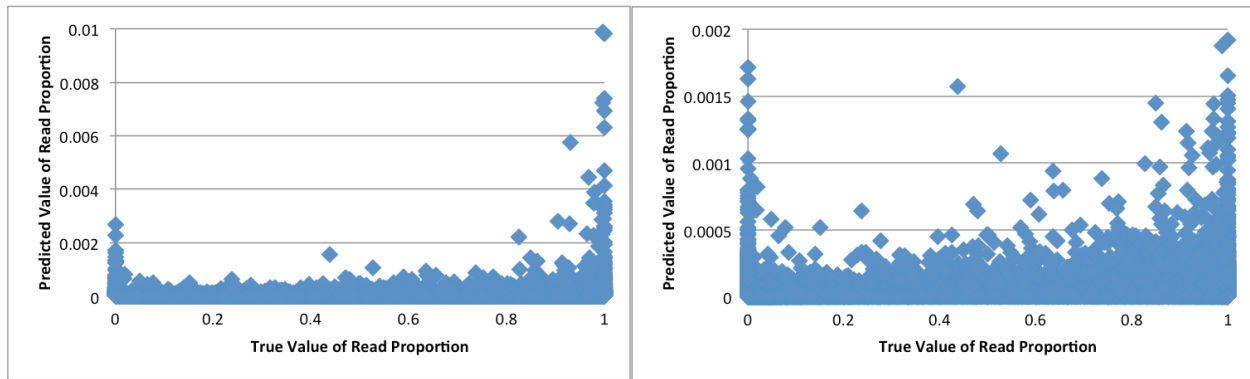


Figure 10. iGWIE Predicted Read Proportions vs. Real Read Proportions. These charts depict the predicted value for read proportion compared against the true value for each isoform. As can be seen above, the predicted and true values are extremely close together, with several outliers. The left plot shows all results; the right plot is a closer view of the left plot, excluding outliers beyond 0.002.

From these results, it can be seen that RAEM is on average more accurate than iGWIE and iGWIE is closer to RAEM estimates than the real values. However, it must be noted that RAEM considers only unique reads and isoforms of a gene are estimated separately, whereas iGWIE considers all reads and all isoforms at once. Also these results are from the seventh iteration of iGWIE. Before losing power, GWIE had iterated much more than seven times. It will take at least several hundred iterations before GWIE or iGWIE gets close to the difference cutoff.

However, from the above results and with this in mind, it is highly likely that if left to iterate just a few more times and not even all the way down to the cutoff, the results of GWIE or iGWIE will surpass the accuracy of RAEM. The closer the iterations move towards the difference cutoff, the more accurate the estimates become. This means that the user may choose to end iGWIE before it reaches the cutoff, perhaps only after a few weeks of iterations, and the results will be much closer to the actual values than RAEM. However, a couple of weeks may still be too long for results that will just be more precise than similarly close results achievable in a few hours via RAEM.

5.2: ChromIE Results

For Chromosome X, the first chromosome to finish analysis, the average difference values between the real values and programs are listed below in Figure 11. Figures 12 and 13 show the accuracy of the predicted proportions against the expected true values.

Real Copy vs. ChromIE Estimated Copy	0.13877052152299002
Real Read vs. ChromIE Estimated Read	0.13873630745205742
Real Read vs. RAEM Estimated Read	0.08363285571106194
Real Copy vs. RAEM Estimated Copy	0.0827108677624375
RAEM Estimated Read vs. ChromIE Estimated Read	0.13078468757456937
RAEM Estimated Copy vs. ChromIE Estimated Copy	0.13079910173739379

Figure 11. ChromIE Analysis of Chromosome X. Average difference between real values and estimated values. Read is the reads proportion; copy is the isoform proportion.

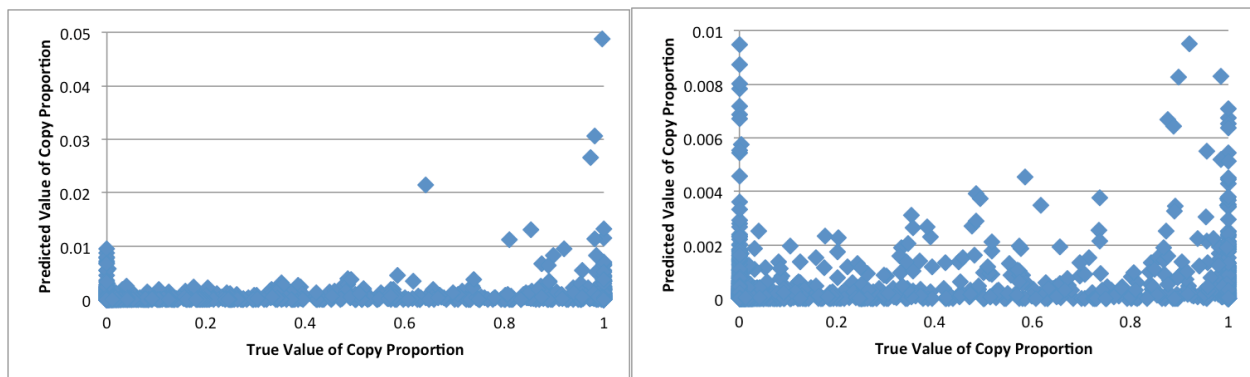


Figure 12. ChromIE Predicted Copy Proportions vs. Real Copy Proportions. These charts depict the predicted value for copy proportion compared against the true value for each isoform. As shown, most values are close to the true value with few outliers. These outliers are concentrated more on the higher bound than both boundaries as in the iGWIE results of Figure 5. The left plot shows all results; the right plot is a closer view of the left plot, excluding outliers beyond 0.01.

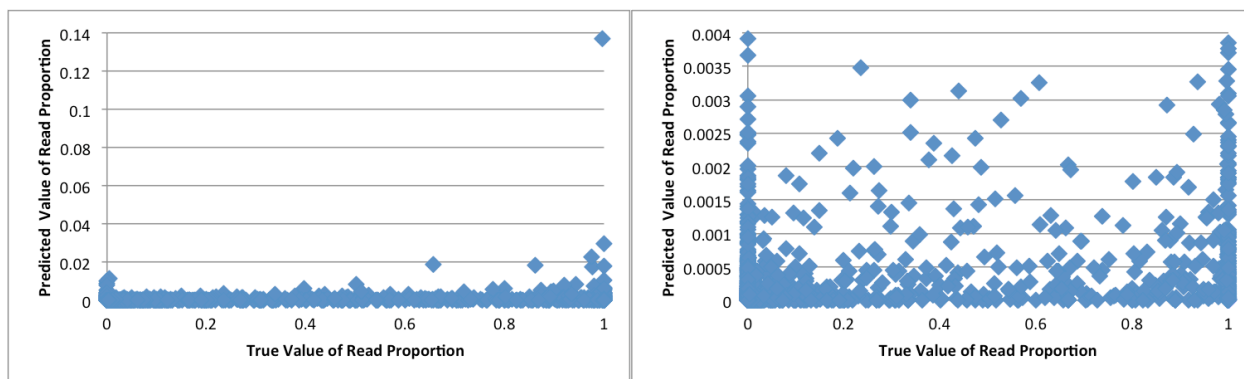


Figure 13. ChromIE Predicted Read Proportions vs. Real Read Proportions. This chart depicts the predicted value for read proportion compared against the true value for each isoform. As can be seen above, the predicted and true values are closer together, with fewer outliers than that of iGWIE as seen in Figure 6. The left plot shows all results; the right plot is a closer view of the left plot, excluding outliers beyond 0.004.

From Figure 11, it can be seen that ChromIE is capable of achieving average estimates that are close to both the real values and the estimates produced by RAEM. These values are also slightly more accurate than those produced by iGWIE but the results are still close. This is probably attributable to the ability of ChromIE to analyze all isoforms of a chromosome together as well as multireads and thus more reads are considered at a time than with RAEM where only reads belonging to a gene are estimated together. For ChromIE, this means that similar genes within the chromosome will be analyzed together, perhaps creating something similar to the compatibility components of IsoEM [25]. Despite the high accuracy of ChromIE, it is worth noting that all other programs discussed in this thesis complete in hours or days, whereas ChromIE took approximately one week to produce the results shown above for chromosome X.

Using the Integrative Genomics Viewer (IGV) [27], Figure 14 was created to display the isoform proportion estimates of both GWIE and ChromIE with respect to the read coverage of the genome. This example is of the gene SHOX (ENSG00000185960) located in chromosome X. It was chosen to display a moderate number of isoforms as well as exon skipping in the isoforms.

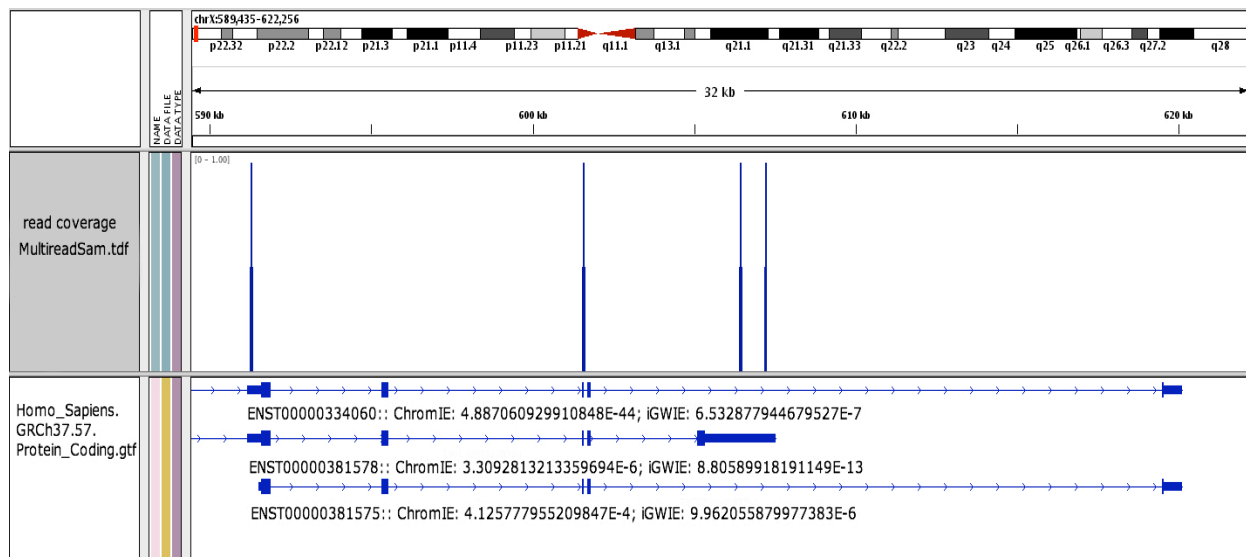


Figure 14. Isoform Proportion Estimates with Read Coverage. In the above figure, three isoforms of the SHOX gene of chromosome X are shown. The values under each isoform indicate the isoform proportion as found by ChromIE and iGWIE on the seventh iteration. From this it can be seen that ChromIE favors the shorter (bottom) isoform copy over the upper isoforms, but the proportion estimates for the middle and bottom isoform are still close. iGWIE favors the two copies (top and bottom isoforms) over the middle isoform that has greater read coverage. This suggests that ChromIE is more capable of finding different isoforms, such as the middle and bottom isoform but not the top as well since it is a copy of the bottom isoform, while iGWIE is more likely to find the copies together and place less emphasis on smaller isoforms, even if they might have more read coverage.

Chapter 6: Conclusion and Future Work

The programs designed for this thesis, GWIE, iGWIE, and ChromIE, have benefits to offer the field of transcriptome quantification, however they also suffer from the major drawbacks of the length of time it takes the programs to run and the amount of memory required to run them. Due to the sequential nature of the programs (data structures must be initialized before use and the E step must be run before the M step since the M step depends on the calculations performed during the E step) concurrency capabilities for this program are limited. With the availability of lighter, faster programs, GWIE and ChromIE may not be of interest to many researchers. If time is not a concern however, these programs can offer a high level of precision over other currently available programs.

Future work to these programs would be to add more implementation to the rudimentary novel transcript finding the programs are currently capable of at the moment. The current versions will simply alert the user that a novel isoform has been found. However, those isoforms are discarded and not currently included in the estimation.

Additional work could be done to use the ExonReadLocationFilter to find alternative splicing junctions. Currently, the programs just use the filter to map exons to reads. Since the filter is designed to allow read overlap based on exon length, additional operations could be added to provide the user with information when overlaps are found that do not originate from actual read to exon overlap.

These programs were never designed for release since it was believed that the memory requirement would be too much. Some finishing touches would be necessary if the programs were to be made available for public use. Though well documented, there are still some areas that would require clarification and older code would need to be deleted. A simple GUI could

also be created to allow users to drag and drop input files, specify the maximum amount of memory to allow the programs to use, and specify which program to run. Instead of printing the results to output files, they could be piped straight into the analyzers and visual representations of the results could be presented to the user.

Most importantly, however, would be to possibly switch the programs from Java to a more mathematical calculation based language, such as R. This would most likely help with both the memory requirement as well as the speed.

References

- [1] “A Basic Introduction to the Science Underlying NCBI Resources: What is a Genome?” *NCBI: A Science Primer*. March 2004. 4 March 2012.
<http://www.ncbi.nlm.nih.gov/About/primer/genetics_genome.html>.
- [2] “A Schematic Representation of Alternative Splicing”. Figure. Scitable by Nature Education. 2011. 20 March 2012. <<http://www.nature.com/scitable/content/a-schematic-representation-of-alternative-splicing-95777>>.
- [3] Alberts, B., A. Johnson, J. Lewis, et al. “Chromosomal DNA and its Packaging in the Chromatin Fiber.” *Molecular Biology of the Cell*. 2002.
- [4] Bilmes, Jeff A. “A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models.” April 1998.
- [5] Borman, Sean. “The Expectation Maximization Algorithm: A Short Tutorial.” Jan. 2009.
- [6] Crick, Francis. “Central Dogma of Molecular Biology.” *Nature*. Aug. 1970: 561 – 563.
- [7] Dellaert, Frank. “The Expectation Maximization Algorithm.” Feb. 2002.
- [8] Deng, Nan, Adriane Puetter, Kun Zhang, Kristen Johnson, Zhiyu Zhao, Christopher Taylor, Erik K. Flemington and Dongxiao Zhu. “Isoform-Level microRNA-155 Target Prediction Using RNA-Seq.” *Nucleic Acids Research*. Sept. 2010 – Jan. 2011.
- [9] Flicek, P., M.R. Arnode, D. Barrell, et al. “Ensembl 2011.” *Nucleic Acids Research*. Nov. 2010. <<ftp://ftp.ensembl.org/pub/release-62/gtf/>>.
- [10] “Flux Simulator.” The Flux Project. 2012 Flux Simulator version 1.0 RC5.
<<http://flux.sammeth.net>>.
- [11] Gupta, Maya R. and Yihua Chen. “Theory and Use of the EM Algorithm.” *Foundations and Trends in Signal Processing*. 2011: 223-296.
- [12] Hamdan, Hani. “Mixture Model Clustering of Binned Uncertain Data: The Classification Approach”. *IEEE Xplore*. 2006: 1645 – 1650.
- [13] “Illumina Sequencing Technology”. Illumina. 2010.
- [14] Jank, Wolfgang. “Stochastic Variants of EM: Monte Carlo, Quasi-Monte Carlo and More”. *University of Maryland*.
- [15] Jiang, Hui and Wing Hung Wong. “Statistical Inferences for Isoform Expression in RNA-Seq.” *Bioinformatics*. Oct. 2008 – Feb. 2009: 1026 – 1032.
- [16] Kim, Hyunsoo, Yingtao Bi, Sharmistha Pal, Ravi Gupta and Ramana V. Davuluri. “IsoformEx: Isoform Level Gene Expression Estimation Using Weighted Non-Negative Least Squares from mRNA-Seq Data.” *Bioinformatics*. July 2011.
- [17] Li, Bo and Colin N. Dewey. “RSEM: Accurate Transcript Quantification from RNA-Seq Data With or Without a Reference Genome.” *BMC Bioinformatics*. 2011.
- [18] Li, Bo, Victor Ruotti, Ron M. Stewart, James A. Thomson and Colin N. Dewey. “RNA-Seq Gene Expression Estimation with Read Mapping Uncertainty.” *Bioinformatics*. Sept. – Dec. 2009: 493-500.
- [19] Li, Heng, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, Richard Durbin, and 1000 Genome Project Data Processing Subgroup. “The Sequence Alignment/Map Format and SAMtools”. *Bioinformatics*. 2009: 2078 – 2079.
- [20] Liu, Chuanhai, Donald B. Rubin, and Ying Nian Wu. “Parameter Expansion to Accelerate EM: The PX-EM Algorithm”. *Biometrika*. 1998: 755 – 770.

- [21] Mardis, Elaine R. "Next-Generation DNA Sequencing Methods." *Annual Review of Genomics and Human Genetics*. June 2008: 387-402.
- [22] Meng, Xiao-Li and Donald B. Rubin. "Maximum Likelihood Estimation Via the ECM Algorithm: A General Framework". *Biometrika*. June 1993: 267 – 278.
- [23] Metzker, Michael L. "Sequencing Technologies – The Next Generation." *Nature Reviews: Genetics*. Dec. 2009 – Jan 2010: 31 – 46.
- [24] Mortazavi, Ali, Brian A. Williams, Kenneth McCue, Lorian Schaeffer, and Barbara Wold. "Mapping and Quantifying Mammalian Transcriptomes by RNA-Seq." *Nature Methods*. May - July 2008: 621 – 628.
- [25] Nicolae, Marius, Serghei Mangul, Ion I. Mandoiu and Alex Zelikovsky. "Estimation of Alternative Splicing Isoform Frequencies from RNA-Seq Data." *Algorithms for Molecular Biology*. Oct. 2010 – Apr. 2011.
- [26] Pan, Qun, Ofer Shai, Leo J. Lee, Brendan J. Frey, and Benjamin J. Blencowe. "Deep Surveying of Alternative Splicing Complexity in the Human Transcriptome by High-Throughput Sequencing." *Nature Genetics*. July – Nov. 2008: 1413 – 1415.
- [27] Robinson, James T., Helga Thorvaldsdóttir, Wendy Winckler, Mitchell Guttman, Eric S. Lander, Gad Getz, and Jill P. Mesirov. "Integrative Genomics Viewer." *Nature Biotechnology*. Jan. 2011.
- [28] Roche, Alexis. "EM Algorithm and Variants: an Informal Tutorial."
- [29] Sari, F. and M.E. Celebi. "A New Accelerated EM Based Learning of the Image Parameters and Restoration". *Neural Networks*. July 2004: 2513 – 2518.
- [30] Schuster, Stephan C. "Next-Generation Sequencing Transforms Today's Biology." *Nature Methods*. Dec. 2007: 16 – 18.
- [31] Si, Yaqing, Peng Liu, Pinghua Li and Thomas Brutnell. "Model-Based Clustering for RNA-Seq Data." Sept. 2011.
- [32] Singh, Ajit. "The EM Algorithm." Nov. 2005.
- [33] Tang, Fuchou, Catalin Barbacioru, Yangzhou Wang, Ellen Nordman, Clarence Lee, Nanlan Xu, Xiaohui Wang, John Bodeau, Brian B. Tuch, Asim Siddiqui, Kaiqin Lao, and M. Azim Surani. "mRNA-Seq Whole-Transcriptome Analysis of a Single Cell." *Nature Methods*. Dec. 2008 – May 2009: 377 – 384.
- [34] Trapnell, Cole, Daehwan Kim, Geo Pertea, Harold Pimentel, Ryan Kelley, Lior Pachter, and Steven Salzberg. "TopHat." 2008 – 2012. Johns Hopkins University, University of California, Berkeley, and Harvard University. 9 March 2012. <<http://tophat.cbcb.umd.edu/>> and <<http://tophat.cbcb.umd.edu/manual.html>>.
- [35] Trapnell, Cole and Ben Langmead. "Bowtie." 2010 – 2012. Johns Hopkins University, University of California, Berkeley, and Harvard University. 9 March 2012. <<http://bowtie-bio.sourceforge.net/index.shtml>> and <<http://bowtie-bio.sourceforge.net/manual.shtml>>.
- [36] Trapnell, Cole, Brian A. Williams, Geo Pertea, Ali Mortazavi, Gordon Kwan, Marijke J. van Baren, Steven L. Salzberg, Barbara J. Wold, and Lior Pachter. "Transcript Assembly and Quantification by RNA-Seq Reveals Unannotated Transcripts and Isoform Switching During Cell Differentiation." *Nature Biotechnology*. Feb. 2010 – May 2010: 511 – 518.
- [37] "UMLet: Free UML Tool for Fast UML Diagrams." Version 11.4. M. Auer, J. Poelz, A. Fuernweger, L. Meyer, T. Tschurtschenthaler. 21 March 2012. <<http://www.umllet.com/>>.
- [38] Wang, Xi, Zhengpeng Wu and Xuegong Zhang. "Isoform Abundance Inference Provides a More Accurate Estimation of Gene Expression Levels in RNA-Seq." *Journal of Bioinformatics and Computational Biology*. July - Sept. 2010: 177 – 192.

- [39] Wang, Zhong, Mark Gerstein and Michael Snyder. “RNA-Seq: A Revolutionary Tool for Transcriptomics.” *Nature Reviews Genetics*. Jan. 2009: 57 – 63.
- [40] Xu, Guorong. “Computational Pipeline for Human Transcriptome Quantification Using RNA-Seq Data.” *University of New Orleans Theses and Dissertations*. 2011: Paper 343.
- [41] Xu, Guorong, Nan Deng, Zhiyu Zhao, Thair Judeh, Erik Flemington, and Dongxiao Zhu. “SAMMate: a GUI Tool for Processing Short Read Alignments in SAM/BAM Format.” *Source Code for Biology and Medicine*. 2011.
- [42] Yao, Zizhen, Zasha Weinberg and Walter L. Ruzzo. “CMfinder – A Covariance Model Based RNA Motif Finding Algorithm.” *Bioinformatics*. June – Dec. 2005: 445 – 452.
- [43] Zhao, Zhiyu, Tin Nguyen, Nan Deng, Kristen Johnson, and Dongxiao Zhu. “SPATA: A Seeding and Patching Algorithm for De-Novo Transcriptome Assembly.” *IEEE International Conference on Bioinformatics and Biomedicine*. BIBM 2011.
- [44] Zhu, Xiaojin. “The EM Algorithm.” 2007.

Vita

Kristen Johnson was born in Metairie, Louisiana. She received her Bachelor of Science in Psychology from the University of New Orleans in 2007. She returned to the University of New Orleans in the Spring of 2009 to pursue a Master's degree in Computer Science. She became a member of Dr. Dongxiao Zhu's group in Fall 2010.