

UNIVERSITY OF ZAGREB  
FACULTY OF ELECTRICAL ENGINEERING AND  
COMPUTING

MASTER'S THESIS NO. 1127

**Splice isoform identification from  
transcript graphs**

Dario Pavlović

Zagreb, June 2015

Zagreb, 6 March 2015

## MASTER THESIS ASSIGNMENT No. 1127

Student: **Dario Pavlović (0036455992)**  
Study: Computing  
Profile: Computer Science

Title: **Splice isoform identification from transcript graphs**

### Description:

One common problem in transcriptome assembly of eukaryotes is the correct identification of splice isoforms. Splice isoforms are different transcripts originating from the same gene, each of which is made of a subset of gene building blocks-exons. Assembly methods usually rely on graph processing methods, and large shared subsequences among different isoforms contribute significantly to the computational complexity of transcript assembly. The main goal of this project is to successfully identify all isoforms present in an overlap graph of a particular gene. The goals are to explore state-of-the-art graph analysis methods when approaching the problem and use the information on sequence coverage distribution to infer the number of different transcripts and exact sequences and lengths of their constituting exons. Synthetic datasets are to be constructed to test the quality of the proposed solutions. The solution will additionally be tested on RNA-Seq data of well known organisms, such as *Saccharomyces cerevisiae*, to estimate the quality on a real-world dataset. The solution should be appropriated for parallel architectures and implemented in C++. The code is to be documented using comments and should follow the Google C++ Style Guide when possible. The complete application should be hosted on Github under an OSI approved licence.

Issue date: 13 March 2015  
Submission date: 30 June 2015


Mentor:



---

Associate Professor Mile Šikić, PhD

Committee Chair:



---

Full Professor Siniša Srblijić, PhD

Committee Secretary:



---

Assistant Professor Tomislav Hrkać, PhD

Zagreb, 6. ožujka 2015.

## DIPLOMSKI ZADATAK br. 1127

Pristupnik: **Dario Pavlović (0036455992)**  
Studij: Računarstvo  
Profil: Računarska znanost

Zadatak: **Identifikacija RNA izoformi iz grafa transkripata**

### Opis zadatka:

Jedan od problema pri sastavljanju transkriptoma eukariota je uspješna identifikacija izoformi. Izoforme su različiti transkripti koji nastaju od istoga gena, a svaka je sastavljena od podskupa eksona. Metode sastavljanja se uobičajeno temelje na radu s grafovima, a veliki zajednički podnizovi između različitih izoformi značajno doprinose računalnoj kompleksnosti sastavljanja transkriptoma. Glavni cilj ovoga projekta je uspješna identifikacija prisutnih izoformi u grafu preklapanja pojedinoga genoma. Potrebno je istražiti mogućnosti primjene suvremenih metoda za analizu grafova u pristupu problemu te korištenje informacija poput razdiobe pokrivenosti u cilju određivanja broja različitih transkripata, te točnih sljedova i duljina eksona koji ih tvore. Potrebno je proizvesti nekoliko testnih skupova podataka različite kompleksnosti i provesti iscrpno testiranje. Rješenje će biti potrebno dodatno testirati na RNA sekvenciranim podacima dobro poznatih organizama kao što je *Saccharomyces cerevisiae* u cilju procjene kvalitete na stvarnim podacima. Rješenje mora biti prilagođeno paralelnoj arhitekturi i napisano u jeziku C++. Programski kod je potrebno komentirati i pri pisanju pratiti stil opisan u Googleovom C++ vodiču. Kompletnu aplikaciju postaviti na Github pod jednom od OSI odobrenih licenci.

Zadatak uručen pristupniku: 13. ožujka 2015.

Rok za predaju rada: 30. lipnja 2015.

Mentor:



---

Izv. prof. dr. sc. Mile Šikić

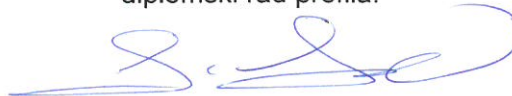
Djelovođa:



---

Doc. dr. sc. Tomislav Hrkać

Predsjednik odbora za  
diplomski rad profila:



---

Prof. dr. sc. Siniša Srbljić

*I would like to thank professor Mile Šikić for his outstanding support and patience during the last three years.*

*Thank you to my family and all those who helped me get to this point.*

*Also, thank you to Jadranka Rezić for proofreading the thesis and providing valuable feedback.*

# TABLE OF CONTENTS

<b>1. Introduction</b>	<b>1</b>
<b>2. Overview</b>	<b>3</b>
2.1. Definitions . . . . .	3
2.2. RNA-Seq and related technologies . . . . .	4
2.2.1. DNA Microarrays . . . . .	4
2.2.2. RNA-Seq . . . . .	5
2.2.3. A comparison of microarrays and RNA-Seq . . . . .	6
2.3. RNA analysis pipelines . . . . .	8
2.3.1. Reference based RNA-Seq analysis . . . . .	8
2.3.2. <i>De novo</i> RNA assemblers . . . . .	9
2.3.3. A comparison of assemblers . . . . .	11
<b>3. Materials and methods</b>	<b>13</b>
3.1. Overview . . . . .	13
3.2. Problem definition . . . . .	13
3.2.1. Maximum likelihood estimation . . . . .	14
3.2.2. Transcript abundance estimation . . . . .	14
3.3. Transcript abundance measures . . . . .	15
3.4. A simple count model . . . . .	16
3.5. Expectation Maximization algorithm . . . . .	17
3.5.1. EM – a simple example . . . . .	18
3.5.2. EM – a description . . . . .	18
3.5.3. EM and mixture densities . . . . .	21
3.6. A robust statistical model . . . . .	23
<b>4. Implementation</b>	<b>30</b>
4.1. General overview . . . . .	30

4.2. External dependencies . . . . .	31
4.2.1. Bowtie2 . . . . .	31
4.2.2. Seqan . . . . .	31
4.3. Code structure and description . . . . .	31
<b>5. Results</b>	<b>34</b>
5.1. Testing . . . . .	34
5.2. Discussion . . . . .	38
<b>6. Conclusion</b>	<b>40</b>
<b>References</b>	<b>41</b>

# 1. Introduction

Transcriptome analysis is one of the most important areas in biological investigation of living organisms. Which transcripts and in what quantities are present in a sample can tell about its state, what genes are expressed and why, how that will affect the state of the organism and its functioning and how are the processes related to transcript levels connected to various diseases and their development.

This work focuses on the transcript isoform abundance estimation problem. Estimating isoform abundances can be of use for better understanding of gene expression levels and transcriptome structure in a sample. Also, comparing transcript or gene expressions levels between various cell types and tissues or in different timepoints of the same tissue can help us better understand fundamental biological processes and their effects on the overall functioning of the living organisms [1].

So-called microarrays have been used a lot for solving this and many other problems in the transcriptome analysis. They are relatively cheap and have existed for almost two decades now, ever since their inception in the early 1990s. They are based on the principle of hybridization between two complementary DNA strands, one called a probe and the other called target sequence. If a fluorescently dyed target sequence strongly binds to a probe, it generates a signal of presence which is then quantified and normalized for further downstream analyses.

High-throughput RNA-Seq [2] is becoming evermore popular for transcriptome analysis. RNA-Seq is an application of next-generation sequencing technologies in transcriptome analysis. Next-generation DNA and RNA sequencing (NGS) are becoming tools of choice for various types of bioinformatics and functional genomics analyses. These sequencing technologies have enabled scientists to produce extremely large amounts of data which are analysed with a plethora of new tools, mathematical models and computational methods. This has greatly enhanced our understanding of the underlying biological structures and principles, as well as significantly improved the sheer quantity and quality of assembled genomes and genomic data of different species. For example, when sequencing DNA using NGS, it is possible to, depen-

ding on the experiment setup, price and desired properties of data, generate numbers of reads ranging from hundreds of millions up to a billion. Such a huge quantity of information is often-times necessary for high-precision and high-sensitivity analyses. DNA and RNA assemblers, for example, often rely on these capabilities for successful reconstruction of genomes and transcriptomes.

The chapters of this work are described as follows:

2. Overview: In this chapter, I will present some important definitions, give a general overview of the RNA study field, including a brief description of microarrays as well as how they compare to RNA-Seq and what are the pros of RNA-Seq over them and other technologies. Also, various ways of tackling abundance estimation problem will be addressed here.

3. Materials and methods: Here, the idea and the statistical model along with statistical algorithms used in this work will be described in detail.

4. Implementation: A general overview of the current code structure and future functionalities will be presented in this chapter along with a brief documentation for each module.

5. Results: The results and how the work in this thesis compares to the results of other abundance estimation software will be analysed in this chapter.

6. Conclusion: A brief conclusion and the look into the future of this work will be provided in this chapter.



## 2. Overview

### 2.1. Definitions

Definitions of some important concepts will be covered here for easier understanding of the text following this section.

A **transcript** is a mature and functional RNA molecule transcribed from DNA [3].

A **transcriptome** is the complete set of RNA and their associated quantity in a cell at any given time [4, 5]. The transcriptome includes messenger RNA (mRNA), ribosomal RNA (rRNA), transfer RNA (tRNA) and short RNA such as microRNA (miRNA). The transcriptome itself constantly changes over time and usually one is analysing its snapshot at a certain timepoint or comparing transcriptome levels in-between different timepoints(snapshots).

An **exon** is a part of genomic sequence that contains coding information for aminoacids and consequently, protein. In other words, it is a part of the genome expressed in the protein [6]. Exons are present in mature mRNA.

An **intron** is a non-coding part of DNA sequence. Introns are removed from mature mRNA before it is translated into protein [6].

A **gene** is, according to Gerstein et al.: "A union of genomic sequences encoding a coherent set of potentially overlapping functional products" [7]. A detailed and extensive discussion on why this definition was proposed is available in the paper. It can also be defined as a basic unit of heredity [6].

**Splicing** is a regulatory stage in mRNA creation in which introns are removed from pre-mRNA and exons are kept and ligated [8].

**Alternative splicing** means inclusion of different exons of a gene in the transcripts it produces [8]. It enables a gene to code for multiple mRNA and protein structures by eliminating and keeping different exon combinations from pre-mRNA. For example, human genome is estimated to have around 20,000 protein-coding genes, but approximately 100,000 different proteins are created from these [8]. Detailed information on alternative splicing can be found in [8, 9] and other resources.

An **isoform** in the context of this work is a synonym for transcript. Also, it can be defined as different forms of protein produced from the same gene by alternative splicing [6].

**Dynamic range** is the ratio between the largest and smallest values of RNA levels that can be accurately quantified [10].

## 2.2. RNA-Seq and related technologies

### 2.2.1. DNA Microarrays

DNA microarrays [2, 11] have been the most popular technology for larger scale analysis of transcriptomes [12] in the past two decades. Microarray technology is based on the hybridization or binding of complementary DNA sequences.

There are 3 widely popular types of microarrays [2]:

- Spotted arrays [13]
- In-situ, Synthesized arrays [14, 15]
- Self assembled arrays [16, 17, 18, 19]

All of these methods have their own characteristics and chemical properties. For example in the first technology type, an array is made up of probes that are aligned precisely in order across a 2-D or 3-D plate made up of glass or silicon. The probes are fixed on the plates by using a covalent or a non-covalent chemical bond. The probes are essentially short or somewhat longer sequences of known genes and/or exons which are used to detect their reversed complementary sequences, cDNA or cRNA, created from the sample's mRNA. The sample cDNA is fluorescently labelled for detection. This technology enables good sensitivity and dynamic range while being relatively cheap [2].

A great characteristic of microarrays compared to older technologies and methods is that there can be tens of thousands of different probes on a single microarray chip, which enables detection and analysis of thousands of different RNA sequences and their presence in a sample, all at the same time. This has been a key feature of microarrays since it has simplified very expensive and relatively low throughput techniques used prior to their advent [4, 20].

Detailed descriptions of microarrays, as well as above-mentioned microarray technologies, are outside of the scope of this work and are available in the papers referenced herein and in other works, as well as in the various web resources.

### 2.2.2. RNA-Seq

RNA-Seq, also called Whole Transcriptome Shotgun Sequencing (WTSS) or short-read massively parallel RNA sequencing [21, 4, 22] uses next-generation deep sequencing to analyse RNA at a certain timepoint in a cell or tissue. In contrast to traditional Sanger-based sequencing, as well as microarrays, RNA-Seq provides enormous amounts of data in a single run and is also highly scalable [22, 23].

The process of mRNA sequencing and downstream analysis is usually as follows [5, 21]. The mRNA is isolated from other RNA in the sample and converted to its corresponding cDNA using reverse transcription. One of the reasons for converting to cDNA is that RNA is not very stable and degrades quickly. The cDNA is then broken into fragments, typically between 200–500 bp in length which marks the creation of a fragment library [4]. Sequencing adaptors are then attached to these fragments and a short piece of sequence is read from either one or both ends of the fragments, obtaining single-end or paired-end reads, respectively. Reads are typically 25-450 base pairs long [22]. Small RNA molecules can be sequenced directly without RNA or cDNA fragmentation. Due to technological limitations, larger molecules must be fragmented as described. Also, it is possible to fragment RNA directly without first converting to cDNA.

Using RNA-Seq technology, reads are determined directly from complementary DNA sequences (or RNA fragments) in a massively parallel fashion. This makes RNA-Seq a digital and quantitative measure of bases present in the sample. The number of reads estimated this way can range from hundreds of thousands, up to a billion. The higher the depth of sequencing, the higher the coverage of transcriptome and easier it is to detect low abundant transcripts. Also, *de novo* assemblers greatly rely on high coverage of the transcriptome and therefore need larger sequencing depth [24] to be effective. If a genome is more complex, or the organism has high alternative splicing levels or other transcriptomic variability across its genome or genes, the greater sequencing depth will be needed. Of course, with larger sequencing depth, comes a greater cost of the experiment. Therefore, it is necessary to try estimating needed coverage in advance, which can be very complicated in transcriptome analysis and has been tackled in various ways in the past [4, 25, 26, 27].

When the reads have been obtained, further analyses can take place. Their type as well as the necessary steps differ depending on the goal of the experiment. RNA-Seq can be used to analyse and estimate overall RNA present in a sample or to quantify only one type of RNA, for example, coding mRNA, or to examine short RNA such as

miRNA. Metagenomic and metatranscriptomic studies are another important application of NGS and RNA-Seq [22]. It can also help in identifying exon-intron boundaries, alternative splicing events, SNPs and in theory provide to a base resolution of transcript and gene boundaries in a genome [4, 22].

Typically, when estimating transcript isoform abundance and/or gene expression levels, the next step involves either mapping of the reads to a reference (annotated) genome of the sample being analysed, or a genome of a related organism, if such exist. Many different tools and methods have been published through the years that rely on this idea [26, 28, 29, 30], with well known being Cufflinks [3] and Scripture [31]. After mapping of the reads, transcript reconstruction and statistical investigation can be done. Another possibility, used when a reference is not available, or is poorly understood and annotated, which is quite often the case, is *de novo* transcriptome assembly. Tools like Trinity [32], Oases [33], SOAPdenovo-Trans [34] and Trans-Abyss [35] are used to reconstruct transcripts from the reads. Here, the analysis continues with the reads being usually mapped back to the transcripts and software packages such as RSEM [36, 37] or eXpress [38] are used for downstream investigation of the data.

### **2.2.3. A comparison of microarrays and RNA-Seq**

An expected question arises when talking about next-generation sequencing applications. How well does RNA-Seq perform compared to older technologies? There were a number of studies that compared RNA-Seq to microarrays, comparing both their similarities and differences [12, 20, 26, 39, 40, 41, 42].

Zhao et al. have focused more on the priorly observed differences between RNA-Seq and microarrays, trying to investigate them in more detail [12]. They used Affymetrix GeneChip HT HG-U133+ PM arrays and compared them to Illumina HiSeq™ 2000 platform on Human CCR6+ CD4 memory T cell. They did observe high correlations between gene expression results from both profiles, but also some significant discrepancies. What they found is that RNA-Seq expression profiles had better results, especially with regards to low abundance transcripts and the detection of changes of these transcripts throughout the course of the experiment. Since microarrays have problems with background hybridization noise, they are unable to precisely quantify low abundant isoforms. Also, they discuss many other disadvantages characteristic of microarray technology, specifically inaccurate annotations for certain probes, problems in detecting splice variants and a limited number of transcripts that can be probed [12].

In a well-known paper, Marioni et al. have tried to assess technical reproducibility

of RNA-Seq experimental data [20] using Illumina platform. Their results confirmed that RNA-Seq is very reproducible, with statistical variation that showed only around 0.5% of genes were outliers from the Poisson model they set up. They also found that RNA-Seq is much more capable of correctly identifying gene expression levels and their changes across samples compared to array technology, with most of the RNA-Seq results being confirmed by qPCR.

Mortazavi et al. haven't focused much on the comparisons of RNA-Seq and arrays in their paper [26], but have provided a very interesting result that shows the potential of RNA-Seq. Namely, they identified that RNA-Seq provides a dynamic range that spans five orders of magnitude. Combined with the amount of data that can be produced by RNA-Seq, it is a powerful realization of the possibilities of the technology. They also found a huge number of new potential transcripts and gene parts, something that was not possible using standard arrays, but can obviously be provided by RNA-Seq.

The other papers have as well identified and confirmed advantages of RNA-Seq over microarrays, most important being non reliance on known annotations and sequences, ability to produce enormous amounts of data, very large dynamic range, high technical reproducibility and the ability to quantify low or high-abundant transcripts and expressed genes.

RNA-Seq is not without its disadvantages, namely price of the experiment, read length, errors and biases typical of next-generation sequencing, especially with regards to fragment distribution. Fragmenting introduces specific biases into the data that should be taken care of in the later stages [4, 5, 43]. Short reads, typical of NGS, are another major problem since they often-times do not cover long enough portions of transcriptomes (or genomes) to distinguish between repetitive regions. When alternative splicing is present, a read coming from a certain exon can originate from any of the isoforms that include that particular exon. Read origin uncertainty is one of the bigger challenges of transcript abundance estimation. Also, sensitivity to read errors usually needs to be met by trying to filter out low-quality reads in some way. In theory, it would be best if all the molecules could be read directly without fragmentation and completely from start to end, generating one read per molecule [24]. This would greatly simplify downstream pipelines. However, since this is currently not possible, computational challenges posed by huge number of very short reads along with experimental biases are non-trivial and are still being actively tackled by scientists using various methods and models. Overall, it is predicted that RNA-Seq will be, if it already is not, a technology of choice for transcriptome analysis and that many of

these obstacles will be cleared over time with RNA-Seq mostly replacing microarrays [4, 22].

## 2.3. RNA analysis pipelines

### 2.3.1. Reference based RNA-Seq analysis

When a reference genome is available, it can be used to help in abundance estimation and gene expression level analysis. Quite a few approaches over the years have been published that relied on mapping of the reads to reference genomes (with or without annotations) [20, 26, 28, 29, 30, 44]. These approaches use various statistical models and tests on read mappings to quantify RNA and gene expression in sequenced samples. Cufflinks [3] and scripture [31] are two well known, robust genome-guided assemblers and downstream analysis pipelines [45].

In short, scripture searches for significant paths in a base connectivity graph. The graph is built from spliced alignments of reads. It scores the paths by using coverage from the reads. Then, identifying these paths, helps it construct a transcript graph which, combined with paired-end information helps reconstruct transcripts.

Cufflinks uses a two-step approach to assembly and analysis. It first tries to reconstruct possible present isoforms using read mappings to a reference genome. For read mapping, it uses TopHat [46], a software designed by the same authors that is specifically suited for mapping of RNA-Seq reads to reference genomes. TopHat tries to align the reads having splice junctions in mind, both existing and possible novel ones. Cufflinks then proceeds to reconstruct transcripts from TopHat mappings. It divides these mappings into sets of non-overlapping loci, assembling each locus independently. To assemble transcripts, it finds a maximum matching in a bipartite graph, trying to obtain a minimal set of transcripts that is in conjunction with reads and their mapping. When it reconstructs possible isoforms, it proceeds to apply a rigorous statistical model to obtain transcript abundance measure. To estimate abundances of each isoform, Cufflinks maximizes the following likelihood function with respect to relative abundances:

$$L(\rho|R) = \prod_{r \in R} \sum_{t \in T} \frac{\rho_t \bar{l}(t)}{\sum_{u \in T} \rho_u \bar{l}(u)} \left( \frac{F(I_t(r))}{l(t) - I_t(r) + 1} \right) \quad (2.1)$$

In this equation,  $\rho$  corresponds to non-negative abundances,  $F$  is the probability distribution of fragment lengths,  $I_t(r)$  is the implied length of a fragment given read

mapping  $r$ ,  $l(t)$  is the length of the isoform  $t$  and  $\bar{l}(t)$  is the effective length of  $t$ . The effective length is defined as:

$$\bar{l}(t) = \sum_{i=1}^{l(t)} F(i)(l(t) - i + 1) \quad (2.2)$$

and can be interpreted as the mean number of positions at which a fragment can start in  $t$  [37]. In general, maximizing some kind of a likelihood function is a very common approach in abundance estimation [3, 30, 36, 37, 29].

An interesting approach to isoform identification and quantification was proposed by Bernard et al. [28]. Therein, they propose the idea of isoform detection and abundance estimation at once, as a solution to a path selection over a graph and call their software FlipFlop. The graph in their method is a set of nodes which represent bins. Bins are defined as ordered set of exons. Edges connect any two bins if the bins can be associated with two reads starting at successive positions in a candidate isoform. They then reduce this path selection to a flow decomposition problem that yields a solution in a relatively fast and efficient manner. Their results show that it performs comparably to Cufflinks with regards to sensitivity, surpassing it on some tests. Since it does solve a harder computational problem, it is slower, but according to the authors, it is a significant improvement over similar and computationally very intensive approaches [28].

### 2.3.2. *De novo* RNA assemblers

When a reference genome or transcriptome is not available, or when it is still being finished and/or is poorly annotated, the only alternative for detailed transcriptome analysis is *de novo* assembly of transcripts from the available reads. Even when the genome is annotated, *de novo* reconstruction can still be used to help in improving the annotation, identifying new transcripts and SNPs. *De novo* assembly presents some of the hardest bioinformatics challenges.

*De novo* assembly of DNA sequences has been extensively studied and has become popular over the years with the improvements in both technology and software used for the assembly. DNA assemblers usually assume some properties of the data they work with, such as uniformity of coverage or usage of sequencing depth to adjust the parameters and to resolve various ambiguities in the assembly process. Assumptions such as these mightily simplify the reconstruction process. However, these assumptions don't work with transcriptome assembly [24]. For example, if sequencing depth

was taken into account, probably only highly expressed transcripts would be reconstructed in the end, which is fundamentally wrong since there can be many isoforms with low abundances that should be detected and assembled. Also, alternative splicing additionally complicates the assembly by having same exonic sequences present in multiple isoforms, making it very difficult to correctly resolve ambiguities [24].

As was mentioned, a number of *de novo* RNA assemblers has been published in the recent years [32, 33, 34, 35]. Many are actually based on genome assemblers with additional post-processing steps or repeated runs in order to accommodate them to RNA-Seq data.

Trans-ABYSS [35], for example, is based on the well known ABySS genome assembler that incorporates k-mer de Bruijn graph approach. Trans-AbySS works by running the genome assembler a few times with different k-mer values. It then proceeds to merge all of these assemblies into transcripts. Trans-Abyss is highly modular because it is structured as a collection of scripts and its various parts can be used with other tools. It also has the ability to do additional analyses, namely prediction of polyadenylation sites, computation of gene-level expression and identification of gene fusion. It is also resource friendly according to some analyses [45].

Oases [33] is, similarly like Trans-ABYSS, based on a genome assembler of its own, namely Velvet. Oases tries to combine multiple k-mers approach of Trans-ABYSS with a more specific topological analysis of the produced contigs. Specifically, it runs Velvet without its later stages that make assumptions valid only for DNA assembly. In each run, it builds a de Bruijn graph, removes the errors, and creates scaffolds that are later grouped and transfrags are extracted from these groups. After all the runs are finished, the resulting transfrags are merged into the final assembly. Its big strength is the error removal step which helps it perform very well [45].

Another genome assembler based transcriptome reconstruction software is known under the name of SOAPDenovo-Trans [34]. It is a modified SOAPDenovo genome assembler that uses advantages of SOAPDenovo together with additional modules designed for RNA-Seq data. It runs in two phases, contig assembly and transcript assembly. Contig assembly is essentially very similar to the same phase in genome reconstruction, with an added error correction module. The transcript assembly phase runs in four substeps, namely scaffold construction, graph simplification, graph traversal, and gap filling. These four stages produce final isoforms.

Trinity [32] is a popular standalone transcriptome assembler divided into 3 independent modules, *Inchworm*, *Chrysalis* and *Butterfly*. Trinity is not built on top of any genome assembler, but is rather specialized for transcriptome assembly. Inchworm is



the first step in Trinity assembly. It is designed as a greedy and fast heuristic that tries to assemble contigs, retrieving one transcript representative from a set of those that share same k-mers. It is a six-step algorithm that uses most frequent k-mers as starting points for contig extension. Chrysalis clusters these contigs into connected components in a recursive fashion. These connected components are likely to contain contigs that are alternative splice forms or closely-related paralogs. The final step is Butterfly. It tries to resolve different spliced isoforms and paralogous transcripts as well as ambiguities. As described, Trinity is a modular software and its modules can be replaced with others, as mentioned in [45].

When the assembly is done and transcripts are produced, tools such as RSEM and eXpress are used to estimated abundances. Similarly to reference-based approaches, these tools also define statistical models which involve maximizations of certain likelihood functions.

### **2.3.3. A comparison of assemblers**

BingXin et al. have tried to compare the performance of various genome-guided and *de novo* RNA assemblers [45]. In their extensive study, they used three distinct datasets to try and evaluate the performance of five assemblers: Cufflinks, Scripture, trans-ABYSS, Trinity and Oases. What their results suggest is that reference-based assemblers are usually superior to the other category. This makes sense, since they can make use of reference genomes, transcriptomes and annotations which can be a significant help in the analysis. However, it turns out, according also to their results, that, as the coverage of RNA-Seq data grows, *de novo* assemblers can perform comparably well, and even surpass reference based ones in their performance. It is important to note that their tests show that all of the assemblers perform better on some and slightly worse on some other data. Overall, however, they've found out that Trinity performs slightly better than other *de novo* assemblers, while Cufflinks remains superior to all of them, especially when paired with reference annotations. What they suggest is that it may be best to merge results from different assemblers (all of them potentially) or even use hybrid reference-*de novo* approaches that should yield much better results than any particular assembler itself.

In another study, Celaj et al. tested how well *de novo* assemblers, namely Trinity and Oases, perform in a metatranscriptomic environment, where RNA from multiple species is present [47]. They compared them to Metavelvet [48] and IDBA-MT [49], a dedicated metatranscriptome assembler. What they found is that Trinity outperformed

the other assemblers with regards to the number of mappable reads, although it has not necessarily reconstructed most accurate sequences.

Clarke et al. [50] have compared the performance of five assemblers, namely Mira [51], Velvet, Trinity, ABySS and Oases. Their results have indicated that Trinity and Velvet have provided best overall performance on the datasets they have used, but were also having inconsistent results across samples. Mira, being an overlap based assembler, had good performance with regards to short reads and contigs, but failed to produce longer contigs. This, according to them, indicates it is not really suitable for analysis of complex eukaryotic transcriptomes.

The common denominator in these studies is that all assemblers have an Achilles heel, meaning that on some datasets they don't perform well. In general, some are better on some data, and the others on other. This means that a need for a new, improved and more robust assembler still exists, as mentioned in [50].

## 3. Materials and methods

### 3.1. Overview

In this and the following chapters, I will present and explain **isomorph**, a software for transcript abundance estimation from RNA-Seq data. `isomorph` estimates abundances by mapping the reads to isoforms reconstructed by a *de novo* assembler. `isomorph` has at its core a statistical model built very similarly to the robust model presented by Li et al. [36, 37] whom have published software called RSEM. Since RSEM has shown that it performs well, and is very popular, even being included in the Trinity pipeline manual and the Trinity package, I have implemented a method similar to RSEM, with slight modifications. `isomorph` also supports a very fast mode of estimation where it just counts the number of reads mapped to a certain isoform and outputs the percentages of mapped reads per isoform. This mode can be used to give one an overall picture of relative abundances in a very fast way, without complex analysis if there is no need for it. `isomorph` can also count the number of bases that are equal between reads and transcripts they mapped to and use this information as an indicator of relative abundances. `isomorph` supports both single and paired end reads, with the model being slightly different between the two. In this work, I have used Trinity for the task of transcript assembly.

### 3.2. Problem definition

Before moving on, a complete problem definition shall be given. There are two important aspects that need to be introduced – maximum likelihood estimation (MLE) in general and transcript abundance estimation with the application of MLE in it. I will use the same notation as the RSEM paper [36, 37] for abundance measures with slight modifications in some parts.

### 3.2.1. Maximum likelihood estimation

Let us assume that we have a probability density function  $p(x|\Theta)$ . This density is a mixture of probability distributions, governed by a set of parameters  $\Theta$ . Now, if we have a data set of  $N$  independent, identically distributed observations drawn from  $p$ , we would like to use those data to estimate the unknown distribution parameters  $\Theta$ . Formally, given a set  $X = \{x_1, x_2, \dots, x_N\}$  of observed data points, and under the assumption that this data was generate by  $p$ , we have the following:

$$p(X|\Theta) = \prod_{n=1}^N p(x_n|\Theta) = \mathcal{L}(\Theta|X) \quad (3.1)$$

This defines the probability of observing the data under the parameters  $\Theta$ , of distribution  $p$ . In other words, if we assume that the observed data is fixed, (3.1) represents the likelihood function of the parameters of the distribution  $p$  given the data. The goal is to estimate the parameters of the distribution with the help of observed that and this is done by finding  $\hat{\Theta}$  that maximizes  $\mathcal{L}(\Theta|X)$ . Formally, we are looking for  $\hat{\Theta}$ , such that:

$$\hat{\Theta} = \underset{\Theta}{\operatorname{argmax}} \mathcal{L}(\Theta|X) \quad (3.2)$$

In practice, logarithm of the likelihood,  $\log \mathcal{L}(\Theta|X)$  is usually optimized for the parameters, instead of (3.1) since it is mostly much easier to tackle and solve [52]. Sometimes, it is very easy to solve for (3.2). For example, if  $p$  is just a single normal distribution with parameters  $\Theta = (\mu, \sigma^2)$ , the derivative of (3.2) can be set to zero and a straightforward calculation will lead to a closed solution for both  $\mu$  and  $\sigma$  [52]. Unfortunately, in many cases it is not so easy because the closed form solution does not exist or is very hard to compute. Here, iterative methods are usually applied that do not guarantee global optimality of the estimate, but converge to a local optimum of the likelihood.

### 3.2.2. Transcript abundance estimation

RNA sequencing provides  $N$  short reads, where  $N$  typically ranges from a million up to hundreds of millions. I assume that reads are of fixed length,  $L$ . There are two interesting measures of relative abundance that can be estimated from the data [36, 37],  $\nu_i$  – *fraction of nucleotides* and  $\tau_i$  – *fraction of transcripts* made up by any particular isoform.

$$\nu_i = \frac{\tau_i l_i}{\sum_i \tau_i l_i} \quad (3.3)$$

$$\tau_i = \frac{\nu_i}{l_i} \left( \sum_i \frac{\nu_i}{l_i} \right)^{-1} \quad (3.4)$$

This definition is based on the assumption that the number of reads coming from any particular isoform is proportional to  $\nu_i$  [36]. The task at hand is to reconstruct  $M$  mRNA isoforms present in the sample using a *de novo* approach, and then estimate their relative abundances by using the reads and their relationship to transcripts. Since the isoform of origin for each read is not necessarily known, this can be a hard problem and I have tackled it by estimating maximum likelihood estimate of  $\tau$  and  $\nu$  using the EM algorithm.

It is also possible to calculate  $\tau$  and  $\nu$  by substituting the length of a transcript in 3.3 and 3.4 for its effective length,  $\bar{l}_i$ , especially when taking read fragment size distribution into account inside the model [37].

### 3.3. Transcript abundance measures

During the years, there have been three commonly used abundance measures, RPKM, FPKM and TPM. RPKM is defined as the number of reads per kilobase per million mapped reads. Formally, as defined in [53], for a gene region  $g$ :

$$RPKM_g = \frac{r_g \times 10^9}{l_g \times R} \quad (3.5)$$

where  $r_g$  is the number of reads that map to  $g$ ,  $l_g$  is the length of  $g$  and  $R$  is the total number of reads. FPKM is similar, but expressed in terms of fragments per kilobase per million mapped fragments. TPM, or transcripts per million, on the other hand, is defined as [53]:

$$TPM_g = \frac{r_g \times r_l \times 10^6}{l_g \times T} \quad (3.6)$$

Here,  $r_l$  is the length of a read and  $T$  is the total number of transcripts in the sample.

In terms of what aforementioned  $\tau$  and  $\nu$  represent as measures, it is easiest to multiply them by  $10^6$  in order to get transcripts per million (TPM) and nucleotides per million (NPM) measures, respectively [36]. Wagner et al. have shown that TPM is in general the best measure of isoform abundance since it is independent of technology and eliminates statistical biases present in RPKM measure [53]. They found that

RPKM is inconsistent across samples and can inflate statistical significance values. Li et al. have compared the estimation of  $\tau$  and  $\nu$  to RPKM estimation [36]. According to them, RPKM is an approximation of  $10^9 \times v_i/l_i$  and in the situation where mean expressed length of transcripts is 1kb, 1TPM is equivalent to 1RPKM. They note that RPKM is highly reliant on mean expressed transcript length, while TPM is much more comparable across samples with different mean lengths. For these reasons, `isomorph` reports relative abundances in terms of TPM and NPM values.

### 3.4. A simple count model

When thinking about relative abundances of isoforms, the simplest way of trying to assess them is to map the reads back to reconstructed isoforms and count the number of reads that has mapped to each isoform. In theory, if the isoform of origin for each read were known, this would be very close to the correct maximum likelihood estimator for isoform abundances [30]. If we assume that the isoform of origin is known for each read, then according to Nicolae et al. [30], the maximum likelihood estimator of  $\tau_i$ , where  $\tau_i$  is the frequency or fraction of isoform  $i$ , would be:

$$\tau_i = \frac{c_i}{\sum_{j=1}^M c_j} \quad (3.7)$$

where:

$$c_i = \frac{n_i}{\sum_{k \leq l_i} F(k)(l_i - k + 1)}. \quad (3.8)$$

Here,  $n_i$  denotes the number of reads coming from isoform  $i$ ,  $k$  iterates over all possible fragment lengths coming from  $i$ , and the denominator is equal to the effective length of transcript  $i$ , which was mentioned in the description of Cufflinks.

There are two significant challenges with this approach. Firstly, the exact origin of a read is not known. In general, multiple alignments for a significant number of reads will be reported, which means they can originate from any of the isoform they mapped to and the positions they mapped to in those isoforms. What to do exactly with these kinds of reads is a question that had multiple answers during the years, with some methods completely ignoring them [20] and others trying to "rescue" them in some way [26], by allocating them to different genes or isoforms proportionally to the number of uniquely mapping reads covering them. Currently, `isomorph` handles both uniquely

mapping and multi-mapping reads. For a multi-mapping read, only the best alignment according to the score reported is taken into account and the corresponding isoform is considered to be the origin of that particular read. The second major challenge are unmapped reads. Sequencing errors, bad read quality or assembler simply not being able to reconstruct all the transcripts present in the sample (very lowly expressed ones for example), can leave a significant portion of the reads unmapped. There are two ways `isomorph` is able to cope with this. It can simply ignore such reads (assuming they are irrelevant) or it can use the idea from [36]. Therein, the authors introduce a notion of a so-called "noise" isoform that they use as a sentinel for unmapped reads. If this second approach is selected, `isomorph` will assign all the unmapped reads to the noise isoform, labeled with index  $i = 0$ , with  $l_0$  being defined as 1. Taking noise isoform into account should produce better estimates for other isoforms.

Another interesting possibility is to map the reads to the transcripts and then check how many bases are exact matches between a read and an isoform. Or it is possible to count how many bases overall are assigned to a certain transcript. The percentage of such bases can be thought of as an approximation to the fraction of nucleotides,  $\nu$ , of the transcriptome that a particular transcript occupies. However, I assume that one should be careful to then run a mapping tool with a less stringent mode of operation, allowing for more errors and lesser quality alignments. Also, this measure should intuitively be more correct as the read coverage of the transcriptome grows and in ideal case approaches infinity, similarly as the fraction of reads should approach  $\nu$  when  $N \rightarrow \infty$  [36].

### 3.5. Expectation Maximization algorithm

Before I present the robust statistical model I used for more sensitive and accurate analysis than simple read counting, here I will describe details of a famous algorithm, used a lot in isoform abundance estimation, starting from Cufflinks all the way to RSEM. It is known as **Expectation-Maximization** or simply EM algorithm [52, 54, 55, 56]. In short, it is often used to find maximum likelihood estimate (MLE) of the parameters of a mixture of probability distributions given some observed data, where there is incomplete or unobserved data, or when a model can be simplified by introducing so-called *latent* variables that model hidden or unknown data.

### 3.5.1. EM – a simple example

Let us firstly consider a simple, often used example [56, 57]. Imagine two physically identical coins being tossed, with each coin having a probability of heads  $p_1$  and  $p_2$ , respectively,  $p_i \in [0, 1], \forall i$ , and we are trying to estimate what these probabilities are (the coins are not necessarily fair and  $p_1$  is not necessarily equal to  $p_2$ ). Let us assume we chose a random coin to toss, and did it for  $R$  rounds. In each round, we tossed it  $X$  times and remembered the results. Now, there are two possibilities. It is either known which coin was tossed in each round, or is not (for example, we forgot which coin we tossed in what round). In the first scenario, it is relatively easy to estimate the probabilities since we know for each coin how many times we got heads and we can simply say that  $p_i = H_i/N_i$ , where  $H_i$  is the number of times coin  $i$  produced heads, while  $N_i$  is the number of times the coin was tossed overall and  $N_1 + N_2 = X \times R$ .

In the second scenario, the solution is not so trivial due to the fact that it is not known which heads or tails came from which coin since we do not know the choices for each round. Here, we can introduce  $Z$ , a hidden or unobserved random variable telling us which coin was used in the  $j$  –  $th$  round. If we knew values of  $Z$  we would be able to compute estimates of  $p_1$  and  $p_2$ . We can proceed in the following manner: Assume  $p_1$  and  $p_2$  are random. Now, having these values, we can guess the most probable distribution of  $Z$ , namely what is the most probable assignment of coins to the rounds and using this guess calculate  $p_1$  and  $p_2$  again. This iterative process could be iterated until we reach a point where  $p_1$  and  $p_2$  do not change any more. This is the basic idea of EM algorithm. The algorithm itself uses more advanced statistics than described in this subsection, but the principle is the same.

### 3.5.2. EM – a description

Expectation-Maximization algorithm [54] is an iterative method of estimating maximum likelihood parameters of a statistical model when there is unobserved or hidden data.

The basic idea behind the EM algorithm is to use a two step iterative process that converges to an optimum of the likelihood function. Starting from a random set of parameters (a point in the likelihood space) it reaches a local optimum by adjusting the parameters in each iteration. The steps used in every iteration are called the E-step and the M-step, respectively.

In general, let us assume we have a set of  $N$  observed data points denoted by  $X = \{x_1, x_2, \dots, x_N\}$ . These points are generated by some probability distribution.



Here, we either know there also exists unobserved data, or introduce unobserved latent variables, marked  $Z$ . This defines our complete data set  $Y = (X, Z)$ , while  $X$  is called *incomplete* data. Now we assume a joint density function:

$$p(Y|\Theta) = p(X, Z|\Theta) = p(X|\Theta)p(Z|X, \Theta) = \mathcal{L}(\Theta|X, Z) \quad (3.9)$$

and call it **complete-data likelihood** while the original likelihood function  $\mathcal{L}(\Theta|X)$  is called **incomplete-data likelihood**. It is important to note that  $\mathcal{L}(\Theta|X, Y)$  is itself a random variable [52] following some distribution. Also, note that the probability  $p(X|\Theta)$  can always be obtained by marginalization of the joint probability:

$$p(X|\Theta) = \sum_z p(X, z|\Theta) \quad (3.10)$$

As was mentioned above, we usually optimize the log of the likelihood. Now, it should be noted how applying the log to (3.9) and (3.10) in general produces significantly different expressions:

$$\log \mathcal{L}(\Theta|X, Z) = \log p(X, Z|\Theta) \quad (3.11)$$

$$\log \mathcal{L}(\Theta|X) = \log \sum_z p(X, z|\Theta) \quad (3.12)$$

In the second case, the sum is an argument to *log*, which is usually very difficult to solve. Therefore, we work with the complete-data log likelihood. Since the values of latent variables are not known, it is not possible to work with complete likelihood directly. The idea of the EM algorithm is to instead do calculations with a lower bound of the data likelihood from (3.12). This is possible by using a result known as *Jensen's inequality*.

**Theorem 3.5.1 (Jensen's inequality)** *Let  $f$  be a convex function defined on an interval  $D$ . If  $x_1, x_2, \dots, x_N \in D$ , and  $\alpha_1, \alpha_2, \dots, \alpha_N \geq 0$ ,  $\sum_{i=1}^N \alpha_i = 1$ , then*

$$f\left(\sum_{i=1}^N \alpha_i x_i\right) \leq \sum_{i=1}^N \alpha_i f(x_i) \quad (3.13)$$

The proof is available in [58]. If a function is concave, then the inequality is reversed [59]. Now, let's use this result to lower-bound (3.12). To do it, we can multiply the

likelihood by 1, but in such a way as to introduce a new distribution  $q(z)$  that will play the role of  $\alpha_i$  in the end [60]. We should also remember that  $\log$  is a concave function.

$$\begin{aligned}
\log \mathcal{L}(\Theta|X) &= \log \sum_z p(X, z|\Theta) \\
&= \log \sum_z p(X, z|\Theta) \frac{q(z)}{q(z)} \\
&\geq \sum_z q(z) \log \left( p(X, z|\Theta) \frac{1}{q(z)} \right) \\
&= \sum_z q(z) \log(p(X, z|\Theta)) - \sum_z q(z) \log q(z)
\end{aligned} \tag{3.14}$$

The EM algorithm tries to maximize the above expressed lower bound. Knowing that the second term, also called the entropy of  $q$ , does not depend on  $\Theta$ , we can ignore it in the optimization step. The only question now is, what will  $q(z)$  be? Usually, the posterior probability  $p(z|x, \Theta)$  is used here as it satisfies both criteria given in 3.5.1 while being possible to be calculated. The lower bound optimized by EM is usually denoted by  $Q(\Theta|\Theta^t)$  and expressed as [60]:

$$Q(\Theta|\Theta^t) = E_{Z|X, \Theta^t} [\log \mathcal{L}(\Theta|X, Z)] \tag{3.15}$$

where  $\Theta^t$  are current parameter estimates and the expect-value is calculated with regards to  $Z$ . Knowing (3.14) and having what was said about  $q(z)$  in mind, (3.15) can be written as:

$$\begin{aligned}
E_{Z|X, \Theta^t} [\log \mathcal{L}(\Theta|X, Z)] &= E_{Z|X, \Theta^t} [\log p(X, Z|\Theta)] \\
&= \sum_z p(z|X, \Theta^t) \log p(X, z|\Theta)
\end{aligned} \tag{3.16}$$

Of course, in the case of a continuous distribution, (3.16) can be written as an integral [52]. The first step of the EM algorithm consists of calculating the above written expect value and is called the E-step. When the expected value is calculated, the M-step proceeds to maximize the result of the E-step with regards to  $\Theta$ . Namely, it tries to find  $\Theta^{t+1}$  so as to maximize (3.15). Formally, the M-step finds:

$$\Theta^{t+1} = \underset{\Theta}{\operatorname{argmax}} Q(\Theta|\Theta^t) \tag{3.17}$$

Usually, this is much easier to solve than (3.2). There is also a modified variant of EM [52], that tries to find any combination of parameters  $\Theta^{t+1}$  such that  $Q(\Theta|\Theta^{t+1}) > Q(\Theta|\Theta^t)$ .

### 3.5.3. EM and mixture densities

Since in *isomorph* (and RSEM) the EM algorithm is applied to finding parameters of a mixture model [36], let us look at how EM can be applied for finding maximum likelihood estimates of a mixture of densities. I have mainly followed the EM mixture-density derivation procedure explained and presented in [52].

We start by assuming the following mixture of  $M$  probability distributions:

$$p(x|\Theta) = \sum_{i=1}^M \pi_i p_i(x|\theta_i) \quad (3.18)$$

such that  $\sum_{i=1}^M \pi_i = 1$ . Every distribution  $p_i$  is governed by its own set of parameters  $\theta_i$ . As for  $\pi_i, i = 1 \dots M$ , they can be thought of as the mixing coefficients of the mixture. If now we apply the log to the presented likelihood, we get:

$$\log \mathcal{L}(\Theta|X) = \log \prod_{n=1}^N \sum_{i=1}^M \pi_i p_i(x|\theta_i) = \sum_{n=1}^N \log \sum_{i=1}^M \pi_i p_i(x|\theta_i) \quad (3.19)$$

which, as introduced in (3.12) has a sum under the logarithm and is very hard to solve and optimize. If, as explained, we now assume that  $X$  is incomplete data and introduce a set of random variables  $Z = \{z_n\}_{n=1}^N$  that tell us from which component density each observation was generated, we get the following:

$$\begin{aligned} \log \mathcal{L}(\Theta|X, Z) &= \log (p(X, Z|\Theta)) \\ &= \log \prod_{n=1}^N p(x_n, z_n|\Theta) \\ &= \sum_{n=1}^N \log p(x_n, z_n|\Theta) \\ &= \sum_{n=1}^N \log (p(x_n|z_n)p(z_n)) \\ &= \sum_{n=1}^N \log (p_{z_n}(x_n|\theta_n)\pi_n) \end{aligned} \quad (3.20)$$

Having simplified the log-likelihood, it is now possible to proceed to calculating  $Q(\Theta|\Theta^t)$ . There is one other quantity needed for that, namely  $p(z_n|x_n, \Theta^t)$ . It can be calculated by the help of Bayes' rule:

$$p(z_n|x_n, \Theta^t) = \frac{\pi_{z_n}^t p_{z_n}(x_n|\theta_{z_n}^t)}{p(x_n|\Theta^t)} = \frac{\pi_{z_n}^t p_{z_n}(x_n|\theta_{z_n}^t)}{\sum_{i=1}^M \pi_i^t p_i(x_n|\theta_i^t)} \equiv h_i^n \quad (3.21)$$

This quantity is known as **responsibility**. Now we can write  $Q(\Theta|\Theta^t)$ .

$$\begin{aligned} Q(\Theta|\Theta^t) &= \sum_z p(z|X, \Theta^t) \log p(X, z|\Theta) \\ &= \dots \\ &= \sum_{i=1}^M \sum_{n=1}^N \log(\pi_i p_i(x_n|\theta_i)) h_i^n \\ &= \sum_{i=1}^M \sum_{n=1}^N \log(\pi_i) h_i^n + \sum_{i=1}^M \sum_{n=1}^N \log(p_i(x_n|\theta_i)) h_i^n \end{aligned} \quad (3.22)$$

A detailed explanation of why the final form of  $Q(\Theta|\Theta^t)$  looks like it does along with a more detailed derivation procedure is available in [52]. It is basically just playing a bit with some a little scary-looking sums that reduce to the final form after observing certain properties of the likelihood function. This defines the E-step of the algorithm. The M-step now maximizes  $Q$  with respect to  $\pi_i$  and  $\theta_i$  and it can do it independently of each other since there is a sum in between and they do not depend on one another. Finding the estimate for  $\theta_i$  will depend on the exact probability distributions assumed. The most often used distributions are Gaussian(normal) distributions. Detailed description for Gaussian distributions can be found in [52]. Since in the model implemented here, I will primarily be concerned with finding equivalent of  $\pi_i$  or prior probabilities of a distribution, let us illustrate the M-step for finding these values. Let us find the derivative of  $Q$  with regards to  $\pi_i$  and set it to 0. Since it has to be that  $\sum_{i=1}^M \pi_i = 1$ , we need to use the Lagrange multiplier to obtain the correct expression. Thus, we obtain:

$$\frac{d}{d\pi_i} \left( \sum_{i=1}^M \sum_{n=1}^N \log(\pi_i) h_i^n + \lambda \left( \sum_i \pi_i - 1 \right) \right) = 0 \quad (3.23)$$

After taking the derivative, the following is obtained:

$$\frac{1}{\pi_i} \sum_{n=1}^N h_i^n + \lambda = 0 \quad (3.24)$$

Summing both sides over  $i$  and multiplying by  $\pi_i$ , we get  $\lambda = -N$  and subsequently by substituting for  $\lambda$  in (3.24):

$$\pi_i^{t+1} = \frac{\sum_{n=1}^N h_i^n}{N} \quad (3.25)$$

This, along with expressions for  $\theta_i$  completes the M-step of the algorithm. The E-step and M-step can now be applied interchangeably in order to converge to an estimate of the mixture parameters, starting from an initial parameter guess.

### 3.6. A robust statistical model

For the purposes of this work, I have used a statistical model originally proposed by Li et. al [36] and Li and Dewey [37] which they use for a software called RSEM. I have implemented their original model with the added fragment length distribution probability from the second paper. The model description is as follows.

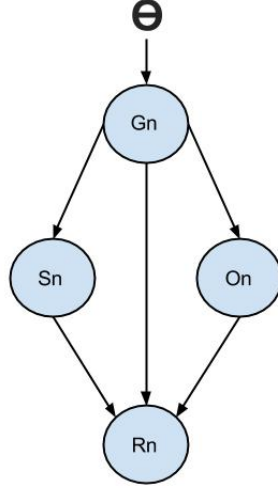
We are given a set of  $N$  reads,  $R = \{r_1, r_2, \dots, r_N\}$  of fixed length  $L$ . We assume that a *de novo* assembler has reconstructed all  $M$  transcripts present in the sample. The goal is to find relative abundances of these transcripts. The absolute abundances can also be found by carefully examining the sample size or by using spike-in measurements [26] but this work is not focused on that aspect. What we are trying to estimate, is a set of parameters  $\Theta = \{\theta_1, \theta_2, \dots, \theta_M\}$  which correspond to the prior probabilities of selecting a transcript a read comes from. The probability of observing the set  $R$  now is:

$$p(R|\Theta) = \prod_{n=1}^N \sum_{i=1}^M \theta_i p(r_n | G_n=i) \quad (3.26)$$

where  $G_n$  refers to the isoform  $n$ -th read originated from. This is a mixture probability function, as was presented in (3.18) and we can use the EM algorithm to estimate its unknown parameters. In general, this model is identifiable and the EM should converge to a local optimum of the likelihood function, although in some cases this may not hold [36]. These cases are usually plateaus of likelihood function with regards to data [36]. Now, trying to directly optimize (3.26) would be very hard, so now we assume that  $R$  is actually incomplete data. The model proposed in [36] introduces three latent random variables that "complete" the data. They are the transcript of origin for  $n$ -th read,  $G_n$ , the read start position in the transcript  $s_n$  as well as the read orientation  $o_n$ . This model is illustrated in the figure 3.1. The figure presents the Bayesian network showing how these variables are related. The complete data likelihood can now be written in the following way:

$$p(R, G, S, O|\Theta) = \prod_{n=1}^N p(g_n|\Theta)p(s_n|g_n)p(o_n|g_n)p(r_n|g_n, s_n, o_n) \quad (3.27)$$

Here, the inner sum from (3.26) is not present any more which makes this expression solvable for optimal values of the parameters  $\Theta$ . The probability  $p(s_n=j|g_n=i)$  models the so-called read start position distribution. This distribution can be modelled in variety of ways [36]. I have chosen for it to be uniform, which assumes that the read can be sequenced starting at any point in the transcript with equal probability. Therefore,  $p(s_n=j|g_n=i)=(l_i - L + 1)^{-1}$ . If the transcripts are assumed to have poly(A) tails,  $p(s_n=j|g_n=i)=l_i^{-1}$ , where  $l_i$  denotes the length of the transcript  $i$  [36]. Next, the latent variable that models the orientation can have only two values, 0 indicating that the read sequence is in the same orientation as its associated isoform, and 1 otherwise. Therefore, for strand-specific protocols,  $p(o_n=0|g_n=i)=1$  and in non-strand-specific case, we simply model it as  $p(o_n=0|g_n=i)=p(o_n=1|g_n=i)=0.5$ .



**Figure 3.1:** Bayesian network of the used model

Lastly, the final probability  $p(r_n=\rho|g_n=i, s_n=j, o_n=k)$ , meaning the probability of a read coming from a specific isoform  $i$ , position  $j$  and orientation  $k$  can be calculated in different ways. Here, since we are assuming the usage of Illumina sequencing data, this probability should model the behavior of Illumina sequencers and the associated properties. One of these properties is that in general, sequencers make more base-call errors towards the end of the read, so the probability of having a mismatch in the beginning should be smaller than at the end of a read. Li et. al have tackled this problem by defining a position-dependent function (matrix) [36] such that:

$$p(r_n=\rho|g_n=i, s_n=j, o_n=k) = \begin{cases} \prod_{l=1}^L w_l(\rho_l, \gamma_{j+l-1}^i), & k = 0, \\ \prod_{l=1}^L w_l(\rho_l, \bar{\gamma}_{j+l-1}^i), & k = 1 \end{cases} \quad (3.28)$$

Here,  $w_l$  is the afore-mentioned function (matrix) that, for each position  $l$ , models the probability of observing the character  $a$  in the read, while observing  $b$  at the position  $j + l - 1$  in the transcript.  $\rho_l$  is the character at position  $l$  inside the read and  $\gamma_{j+l-1}^i, \bar{\gamma}_{j+l-1}^i$  are characters at position  $j + l - 1$  of transcript  $i$ , with  $\bar{\gamma}$  denoting the reverse complement of a character at the corresponding position. In other words, if the read is assumed to start at position  $j$  of a transcript, then, comparing the transcript and the read position by position starting at  $j$  and 1, respectively, we obtain the probability  $p(r_n=\rho|g_n=i, s_n=j, o_n=k)$  by using  $w_l$ . In this work, I have used a simplified model, instead of a complex  $w_l$ . Simply, if there is a mismatch in the first third of read length, the corresponding probability is 0.2, in the second third it is 0.4 and in the last third it is 0.6. Another approach I used is to simply have  $w_l(\rho_l, \gamma_{j+l-1}^i)=0.5$  for any position where there is a mismatch and 1 otherwise. Formally, in the first approach:

$$w_l(\rho_l, \gamma_{j+l-1}^i) = \begin{cases} 1, & \rho_l = \gamma_{j+l-1}^i, \\ 0.3, & \rho_l \neq \gamma_{j+l-1}^i \text{ and } l \leq L/3, \\ 0.6, & \rho_l \neq \gamma_{j+l-1}^i \text{ and } L/3 < l \leq 2L/3, \\ 0.9, & \rho_l \neq \gamma_{j+l-1}^i \text{ and } 2L/3 < l, \end{cases} \quad (3.29)$$

and in the second approximation,  $w_l$  is defined as:

$$w_l(\rho_l, \gamma_{j+l-1}^i) = \begin{cases} 1, & \rho_l = \gamma_{j+l-1}^i, \\ 0.5, & \rho_l \neq \gamma_{j+l-1}^i \end{cases} \quad (3.30)$$

and identically for  $\bar{\gamma}_{j+l-1}^i$ . Both of these approaches are very rudimentary approximations of sequencing experiments, but are fast to calculate and do not involve any additional complex calculations.

Now, let us derive the expressions for the E and M steps of the EM algorithm, having the complete-data likelihood as a starting point. These derivations are also available in [36] supplementary material. First, let's introduce an indicator random variable  $z_{nik}$  [36] such that:

$$z_{nik} = \begin{cases} 1, & (g_n, s_n, o_n)=(i, j, k), \\ 0, & \text{otherwise} \end{cases} \quad (3.31)$$

Every read will have a set of these variables assigned and they replace the original notation. For example,  $p(r_n=\rho|g_n=i, s_n=j, o_n=k)$  now becomes  $p(r_n=\rho|z_{nik}=1)$ . It is clear that  $\sum_{i,j,k} z_{nik}=1$  or in other words, the read can come from only one isoform, start position and orientation. The problem here is, we do not know what these exactly are so we are trying to estimate the expected value of the joint probability distribution  $p(r, z|\Theta)$  with regards to  $z$ . For the E-step, we need to find:

$$\begin{aligned} Q(\Theta|\Theta^t) &= E_{Z|R, \Theta^t} [\log \mathcal{L}(\Theta|R, Z)] \\ &= E_{Z|R, \Theta^t} [\log p(R, Z|\Theta)] \end{aligned} \quad (3.32)$$

Before continuing, it should be noted that we can write the complete-data likelihood function as follows:

$$\begin{aligned} p(R, Z|\Theta) &= \prod_{n=1}^N \prod_{i=1}^M \prod_{j=1}^{l_i} \prod_{k=0}^1 (p(r_n, z_{nik}|\Theta))^{z_{nik}} \\ &= \prod_{n=1}^N \prod_{i=1}^M \prod_{j=1}^{l_i} \prod_{k=0}^1 \left( \frac{\theta_i}{l_i} p(r_n|z_{nik}=1) \right)^{z_{nik}} \end{aligned} \quad (3.33)$$

This is possible because we assume only one  $z_{ijk}=1$  for each read. Now, applying log to (3.33) gives [36]:

$$\log p(R, Z|\Theta) = \sum_{n,i,j,k} z_{nik} \log \left( \frac{\theta_i}{l_i} p(r_n|z_{nik}=1) \right) \quad (3.34)$$

Equation (3.32) now becomes:

$$\begin{aligned} Q(\Theta|\Theta^t) &= E_{Z|R, \Theta^t} [\log p(R, Z|\Theta)] \\ &= \sum_{n,i,j,k} E_{Z|R, \Theta^t} [z_{nik}] \log \left( \frac{\theta_i}{l_i} p(r_n|z_{nik}=1) \right) \end{aligned} \quad (3.35)$$

This is because the log is not dependent on  $z_{nik}$  since  $p(r_n|z_{nik}=1)$  can be calculated as in (3.28). It is well known that for indicator, or Bernoulli, random variables, and  $z_{nik}$  are just that,  $E_{Z|R, \Theta^t} [z_{nik}]$  can be expressed by the help of Bayes' rule as follows:

$$\begin{aligned} E_{Z|R, \Theta^t} [z_{nik}] &= p(z_{nik} = 1|R, \Theta^t) \\ &= \frac{(\theta_i^t/l_i) p(r_n|z_{nik}=1)}{\sum_{q,r,w} (\theta_q^t/l_q) p(r_n|z_{nqrw}=1)} \end{aligned} \quad (3.36)$$



The E-step is now completed. The only thing remaining is to substitute (3.36) into (3.35). In the M-step, we need to find the first order derivative of  $Q(\Theta|\Theta^t)$  with respect to  $\theta_i, \forall i$  and set them to 0 to solve for  $\theta_i^{t+1}$ . Of course, it is important not to forget that  $\sum_i \theta_i = 1$  is a condition on  $\Theta$ . Therefore, we take Lagrange multiplier  $\lambda$  into account. Starting with (3.35), we obtain:

$$\begin{aligned} & \frac{d}{d\theta_i} \left( Q(\Theta|\Theta^t) + \lambda \left( \sum_{m=1}^M \theta_m - 1 \right) \right) = \\ & = \frac{d}{d\theta_i} \left( \sum_{n,i,j,k} E_{Z|R,\Theta^t} [z_{nik}] \log \left( \frac{\theta_i}{l_i} p(r_n | z_{nik}=1) \right) + \lambda \left( \sum_{m=1}^M \theta_m - 1 \right) \right) \quad (3.37) \\ & = \frac{\sum_{n,j,k} E_{Z|R,\Theta^t} [z_{nik}]}{\theta_i} + \lambda = 0 \end{aligned}$$

To solve for  $\lambda$ , we can multiply the last expression by  $\theta_i$ , sum it from 1 to  $M$  and set it to 0. This is possible because all sum members are obviously 0, and therefore the whole sum must also be equal to 0.

$$\sum_{i=1}^M \left( \sum_{n,j,k} E_{Z|R,\Theta^t} [z_{nik}] + \lambda \theta_i \right) = \sum_{n,i,j,k} E_{Z|R,\Theta^t} [z_{nik}] + \lambda \sum_{i=1}^M \theta_i = N + \lambda = 0 \quad (3.38)$$

Here,  $\sum_{n,i,j,k} E_{Z|R,\Theta^t} [z_{nik}]$  equals  $N$  because for each read, the inner sum over  $(i, j, k)$  equals the sum of probabilities which have to sum to 1. Now we have solved for  $\lambda$  and can go back to (3.37) to find  $\theta_i^{t+1}$ . Namely, we obtain:

$$\frac{\sum_{n,j,k} E_{Z|R,\Theta^t} [z_{nik}]}{\theta_i} - N = 0 \quad (3.39)$$

and from here easily solve for  $\theta_i^{t+1}$ . Finally, the M-step can be written as:

$$\theta_i^{t+1} = \frac{\sum_{n,j,k} E_{Z|R,\Theta^t} [z_{nik}]}{N} \quad (3.40)$$

This result is quite intuitive. The reason is, the parameters for the next iteration are calculated as an estimate of the expected number of reads that belong to isoform  $i$  in the current iteration, divided by the total number of reads. If we knew the origin of each read, this would simply be reduced to the number of reads coming from that isoform divided by the total number of reads. With this, the derivation of EM algorithm for our problem has been completed. However, this result is very tedious to

calculate, especially when number of reads and transcripts is large (which is usually the case). The sum over all reads and over all isoforms can be very time consuming to obtain in each iteration. For that reason, the idea is to reduce the "search space" [36] by first mapping the reads to transcripts and allowing only mappings with at most  $x$  mismatches to be considered valid. These mappings will then define which transcripts are joined to what reads, and in what positions and orientations. Only these pairs of transcripts and positions will be used to calculate the parameters in the M-step, instead of every possible transcript. More specifically, let us define  $\pi_n^x$  as a set of isoforms and associated positions read  $n$  maps to in such a way that there are at most  $x$  mismatches [36]. Then, we adjust (3.36) to be the following:

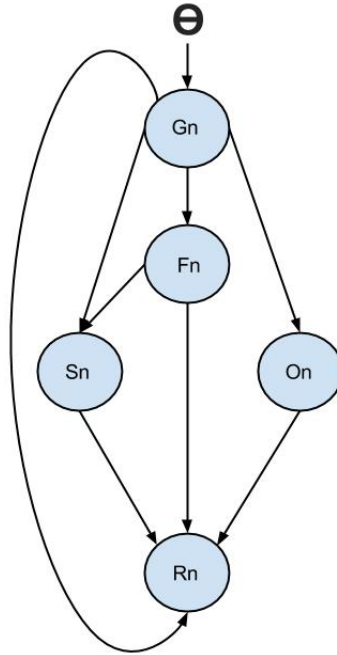
$$E_{Z|R,\Theta^t}[z_{nik}] = \begin{cases} \frac{(\theta_i^t/l_i)p(r_n|z_{nik}=1)}{\sum_{q,r,w \in \pi_n^x} (\theta_q^t/l_q)p(r_n|z_{nqrw}=1)}, & (i, j, k) \in \pi_n^x, \\ 0, & otherwise \end{cases} \quad (3.41)$$

Usually, this is much faster to solve than the original expected value because reads can map to only a very limited number of transcripts in this way. We adjust the expression for  $\theta^{t+1}$  (3.40) accordingly. *isomorph* does the E and M step for a specific number of iterations, which is 2000 by default and outputs the results. If it is assumed that reads are uniformly sequenced from the transcriptome [36],  $\theta_i$  correspond to  $\nu_i$  described in (3.3) and  $\tau_i$  can be calculated by using (3.4).

Using read mappings actually provides an approximation to original expressions derived here and in [36], albeit a good one. In the count model, I mentioned the noise isoform which is used in [36]. *isomorph* does not use this idea when doing the EM algorithm (but does with the count model) and for that reason, I have not included it in the EM derivation. Also, in the Supplementary Material of [36], it was proposed to skip the reads with a large number of mappings in the calculations since they do not bring along a lot of new information and are computationally heavy at the same time. *isomorph* also implements that idea, but not in the same way. It rather ignores such reads completely, while RSEM tries to compensate for them later on by using additional calculations (see Supplementary Material of [36]). For read mapping, I have used *bowtie2* [61], with parameters set exactly as in RSEM.

This basic model can be further expanded, as proposed in [37] and shown in figure 3.2. In that paper, the authors have added fragment length distribution, read qualities, and read lengths as latent variables. Since they have shown that modeling read qualities

does not significantly improve the results for Illumina data and since I was working with Illumina reads of fixed length, right now `isomorph` only supports fragment length distribution added to this basic model and ignores read qualities as well as read length distribution. Fragment length is modeled only when analyzing paired-end reads. For single-end, it is also ignored.



**Figure 3.2:** Bayesian network of the model extended with fragment length variable

To estimate fragment length, I have assumed that it is a normal distribution, similarly as in [3], whose parameters,  $\mu$  and  $\sigma$  `isomorph` estimates from valid read mappings using TLEN field of the reported SAM alignment file. Then, when calculating probabilities  $p(R, Z|\Theta)$ , fragment length is taken into account by additionally multiplying with  $p(f = x|G = i) = I(x) / \sum_{y=1}^{l_i} I(y)$  where  $x$  is the implied fragment length of the mapping and  $I$  is the fragment length probability distribution. The expression is normalized over all possible fragment lengths coming from isoform  $i$  [37]. Finally,  $\tau$  is modified to be calculated by replacing the length in (3.4) with the effective length of transcript  $i$  [37].

# 4. Implementation

## 4.1. General overview

Implementing the model should be relatively straightforward after deriving the expressions for E and M steps. I have implemented it in the most optimal way I was able to. A couple of things can be noticed from the description of the model. Firstly, the probabilities  $p(r_n|z_{nik})$  can be pre calculated once and used later on since they do not depend on  $\theta_i^t$ . Calculating them in every iteration would be very time consuming. Normalisation factors, namely  $\sum_{y=1}^{l_i} I(y)$  should also be calculated once because they are always the same for a certain isoform. Further, estimating fragment length distribution parameters does not have to be done from all possible alignments, but rather only first (or random) million (or even less) since that should represent a relatively good sample. If the number of reads, and consequently number of their alignments is huge (>100 mil), this can be a significant time saver.

I have also tried to enable easy plugging in of possibly different and new estimators that behave according to other or new models, by applying strategy design pattern. What this means is that adding a new estimator should be easy by inheriting `Estimator` abstract base class and implementing `estimate_abundances` method. Also, adding new mapping tools instead of bowtie2 should be painless because the running of the alignment is encapsulated inside a parametrized method in the utility source file and only the method and its parameters should be adapted accordingly. To reconstruct transcripts from the reads, any assembler can be used since `isomorph` depends only on the transcripts produced and this step must be run before running `isomorph`. I have used Trinity for that task in this work.

The source code is written in C++ and documented according to Doxygen conventions for easy automatic generation of the documentation provided by doxygen tool. The code was written according to Google C++ Style guide with possible minor deviations when I thought it would result in a more readable code. The code also includes

Makefiles written according to seqan [62] tutorial and convention, and Python scripts used for testing. `isomorph` methods, data and functions are all wrapped inside an `isomorph` namespace to not clutter the environment of programs that could possibly use it and to enable it to become a full-blown standalone library in the future. A detailed README file, along with doxygen docs and source code for download is available at <https://github.com/darxsys/isomorph>.

## 4.2. External dependencies

### 4.2.1. Bowtie2

For aligning the reads to reconstructed transcripts, I have used bowtie2 [61] in the same fashion as it is used in RSEM [37]. The essential idea is to let bowtie produce all good enough alignments and later use the EM statistical model to quantify them and not let bowtie decide what is good and what is not [36, 37]. `isomorph` runs bowtie with two sets of parameters, depending on if it is working with single-end or paired-end reads. To enable greater flexibility, tools such as BWA [63] and others will be supported in future releases.

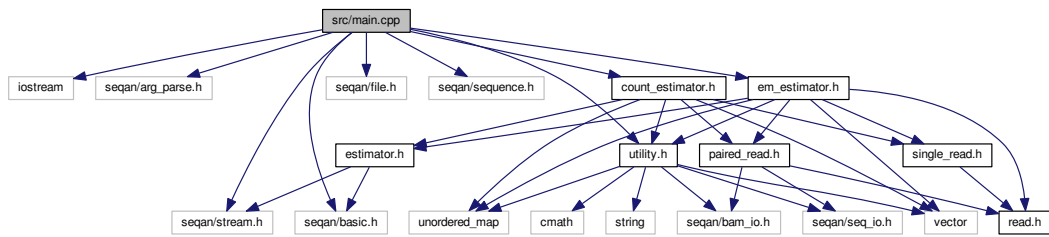
### 4.2.2. Seqan

For easier handling of input and output data, especially parsing of SAM [64] alignment files, as well as parsing command line arguments in a clean fashion, I have used seqan [62]. Seqan is an open source C++ library providing many different APIs, methods and data structures for handling of biological data. `isomorph` currently uses seqan modules for reading FASTA, FASTQ and SAM files, as well as its argument parser library. `isomorph` adds an extra layer of abstraction by encapsulating seqan's data structures and used functions into its own for easier downstream usage and later modification.

## 4.3. Code structure and description

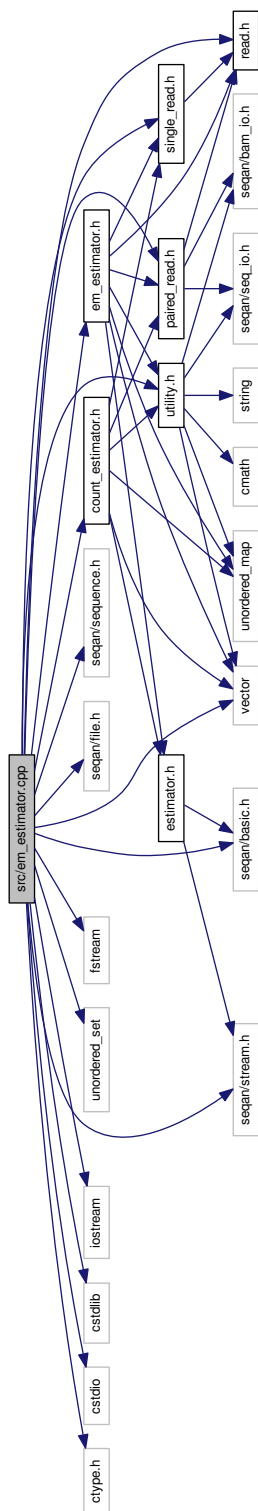
The code itself is divided into modules for doing different tasks. A general overview of the dependencies between files and outside libraries can be seen in figure 4.1. The modules currently included into `isomorph` are as follows:

- `main.cpp` – The entry point. Includes functions for argument parsing and various checks.



**Figure 4.1:** General overview of file dependencies in `isomorph`

- `utility.h/utility.cpp` – Encapsulates often used functions and classes that can be used by other modules, including running of bowtie alignment, reading input files, calculating normal distribution probabilities and reverse-complementing characters and sequences.
- `count_estimator.h/count_estimator.cpp` – This module defines a simple count estimator explained in section 3.4. It defines `CountEstimator` class that inherits `Estimator` class and implements the public virtual method `estimate_abundances`. It also implements other, utility methods needed for data processing, but these are purposefully hidden from the class clients.
- `em_estimator.h/em_estimator.cpp` – `isomorph`'s core module. It implements the advanced statistical model explained under Methods chapter. Similarly to `count_estimator`, it inherits `Estimator` class and hides everything from clients except `estimate_abundances` method. Detailed documentation of other, auxiliary methods and data structures inside this module is available at <https://github.com/darxsys/isomorph>. Figure 4.2 shows the dependencies of this module.
- `read.h/paired_read.h/single_read.h` – These three headers declare and encapsulate information related specifically to the reads. This includes the set of read alignments, read sequence and quality strings and enables easier manipulation with the reads and their associated mappings as well as future extensibility. These modules could be significantly changed in future releases.



**Figure 4.2:** The dependencies for the core em\_estimator module

# 5. Results

## 5.1. Testing

To test performance of `isomorph`, specifically core EM algorithm, I have used a method similar to that in [28]. Since I am not aware of any real benchmark RNA-Seq datasets, I have used two RNA-Seq read simulators – one packaged with RSEM and the other called Polyester [65] to simulate reads coming from reconstructed or annotated transcripts.

RSEM simulator is very customizable and enables users to manually set expression levels of every particular isoform as well as providing many other simulation options. It uses RSEM’s built-in model along with the results of RSEM run on the data to produce a new dataset of reads that correspond to custom expression values. To use this simulator, I have downloaded three different read datasets from the European Nucleotide Archive <http://www.ebi.ac.uk/ena>, two using single-end and one using paired-end sequencing protocol. All of these reads come from often-used yeast (*Saccharomyces cerevisiae*) organism. The single-end sets can be found under run accession IDs ERR458493 and SRR960622. I will refer to these sets as Y1 and Y2, respectively. For paired-end data, I have downloaded reads with run accession ID SRR059167 and will refer to it as Y3. I ran Trinity for each of these read sets in order to produce reference transcripts. As mentioned, to use RSEM simulator, first RSEM needs to be run on the data, so I did just that. After that, I ran the simulator to generate 10 million reads for each set using default parameter values, but setting the noise isoform fraction to 0 taking care that the simulator does not produce noise reads coming from it since `isomorph` does not take those into account when running the EM algorithm currently. Simulated read length for Y1 was 51bp, for Y2 it was 36bp and Y3 had a mate-length of 51bp. In the case of Y1 set, Trinity reconstructed 2897 transcripts, for Y2 that number was 8385 and for Y3 1960 and this were used as reference transcripts for `isomorph` testing.

Polyester [65] is a read simulator targeted to enable the analysis of differential ex-



pression at both gene and transcript level. It can simulate experiments with biological replicates and fold change between experiments has to be specified by the user. I have used it to generate sets of reads from human chromosome 22 annotated transcripts, which are packaged together with polyester. This set is comprised of 918 transcripts. Since I was not testing for differential expression, I used it to only generate one replicate per run. Polyester uses a negative-binomial distribution to draw reads from reference transcripts and can be customized with many different parameters. In my use case, I just asked it to provide number of reads coming from each transcript to be a function of the length of that particular transcript. This is possible by providing a flag `meanmodel=TRUE` to it. Using it, I created two single-end and one paired-end simulated dataset, all of them by simulating from the same reference transcript set. I will refer to these sets as CT1, CT2 and CT3 respectively. CT1 and CT2 single-end sets had 273979 and 273014 reads generated. For CT3, 275857 reads were obtained. Length of the reads was set to 60bp for all three tests. In case of CT3 set, each mate of a pair had a length of 60bp. Polyester does not report expected TPM or RPKM values for each transcript after simulating the reads (at least I was not able to find a way to do so), but it does provide, for each read, the exact transcript of origin. Using this information along with the assumption that reads are sequenced in proportion to  $\nu_i$  (3.3)[36], the expected TPM for an isoform can be calculated as:

$$\tau_i = \frac{N_i}{l_i} \sum_{k=1}^M \left( \frac{N_k}{l_k} \right) * 10^6 \quad (5.1)$$

where  $\tau_i$  represents TPM value of isoform  $i$ ,  $M$  is the number of transcripts and  $l_k$  is the length of transcript  $k$ . After simulating the reads, I used this formula to find expected values of TPM for each transcript in the sample. Then, I ran `isomorph` for all three datasets and analysed the results. To do the analysis of all of these results, I measured how good `isomorph`'s relative abundance estimates are compared to the expected ones. To do this, I used the following expression:

$$\delta_i = \frac{|TPM_{iso,i} - TPM_{sim,i}|}{TPM_{sim,i}} \quad (5.2)$$

This expression is the percent change between an expected (provided by simulator) and obtained TPM value. I found the mean, median, and standard deviation for  $\delta$  values in every test sample. For comparison, I also ran RSEM and IsoEM [30] and reported their results. Table 5.1 shows summary of all results. The table also shows the fraction of TPM values reported by `isomorph` that were no more than five percent different than the expected ones and can therefore be regarded as very accurate. This column is

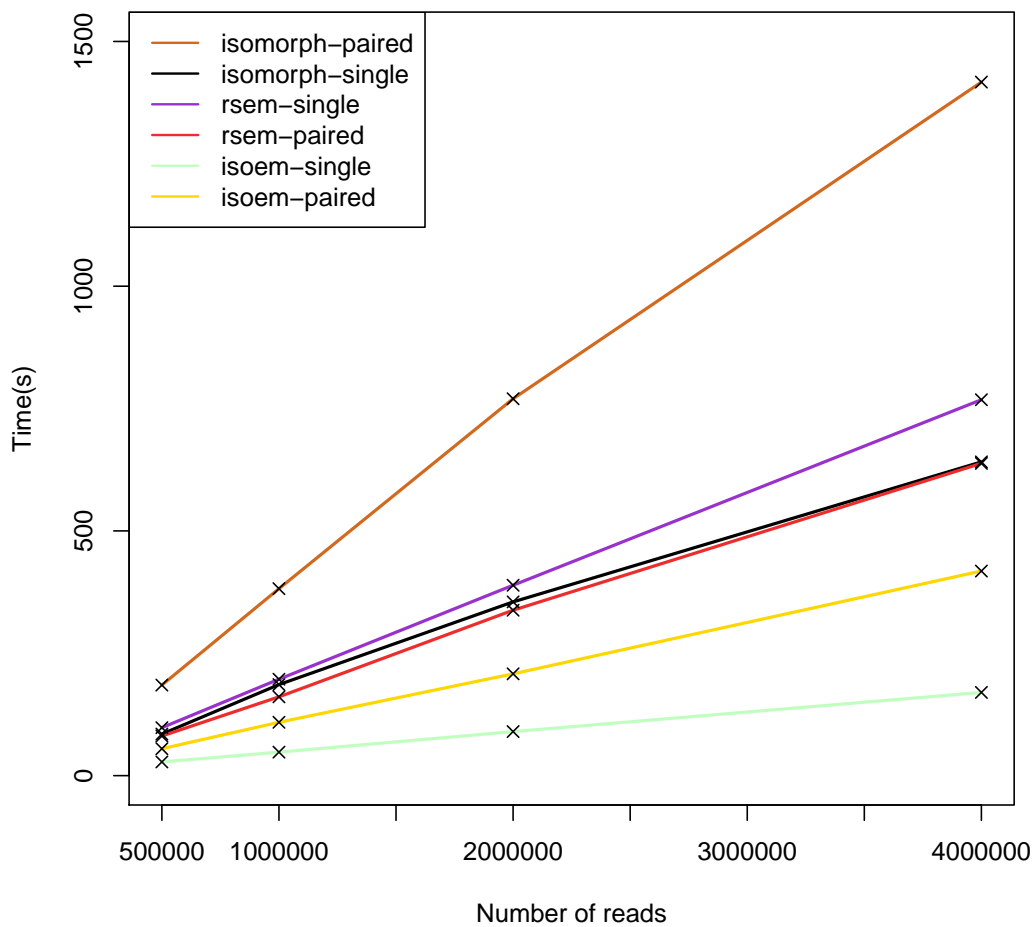
marked by  $P_{eq}$ . IsoEM kept throwing exceptions when I ran Y3 set using it, so I could not report any results there. I ran RSEM with all default parameters. To test IsoEM, I ran it according to the simple example provided in the README bundled with it, without changing any default parameters. `isomorph` was also run in the same fashion, using default parameters.

**Table 5.1:** Results of running `isomorph`, IsoEM and RSEM on the same simulated data. Shown are, mean, median and standard deviation of values of  $\delta$  for each sample, as well as the percentage of  $\delta$  values that were less than 5 percent. I was not able to run IsoEM on Y3 set due to a runtime error it kept throwing.

Software	Dataset	Mean $_{\delta}$	Median $_{\delta}$	Stdev $_{\delta}$	$P_{eq}$
isomorph	Yeast-Y1-Single	0.067	0.060	0.070	0.423
	Yeast-Y2-Single	0.050	0.036	0.444	0.672
	Yeast-Y3-Paired	0.278	0.225	0.269	0.096
	CT1	0.170	0.070	0.243	0.466
	CT2	0.189	0.070	0.261	0.460
	CT3	0.147	0.062	0.202	0.471
IsoEM	Yeast-Y1-Single	0.113	0.060	1.895	0.424
	Yeast-Y2-Single	0.124	0.050	4.751	0.497
	Yeast-Y3-paired	-	-	-	-
	CT1	0.190	0.068	0.269	0.449
	CT2	0.197	0.070	0.260	0.449
	CT3	0.989	0.936	0.421	0.0
RSEM	Yeast-Y1-Single	0.030	0.021	0.045	0.831
	Yeast-Y2-Single	0.057	0.038	0.064	0.59
	Yeast-Y3-paired	0.074	0.018	0.213	0.822
	CT1	0.170	0.070	0.24	0.004
	CT2	0.702	0.670	0.563	0.004
	CT3	0.839	0.748	1.222	0.0

To test and compare `isomorph` runtime, I took Y2 single-end dataset along with the paired-end Y3 data and created sets of 500 thousand, 1 million, 2 million, and 4 million reads for each of the two using RSEM simulator. Then, for each set, I ran `isomorph`, RSEM and IsoEm, and measured overall execution time. For IsoEM, I have included the times for converting to genomic coordinates, since that module is an important part of the software itself. All results also include times for running bowtie.

The results, reported in seconds per run, can be seen in figure 5.1. Testing was done on a machine with 16 Intel(R) Xeon(R) E5-2640 CPUs clocked at 2GHz each and having HyperThreading support. Also, it has 256GB SSD drive and 396GB of RAM. `isomorph` currently supports only single threaded execution so one core was used during the testing.



**Figure 5.1:** `isomorph`, RSEM and IsoEm runtime depending on the number of reads.

The plot in figure 5.1 implies that the runtime is roughly linear in the number of reads, although one should be cautious to also consider number of possible read mappings as a factor due to the fact that if many reads have a lot of multiple mappings, this could slow down `isomorph`. Also, it takes around two times less time to process single-end than paired-end reads. This makes sense, since for paired-end reads, `isomorph` does insert size estimation and calculates fragment length probabilities in every iteration of the EM algorithm. Also, processing alignments for paired reads is

slightly more complicated and time consuming.

I did not test `isomorph`'s other module – namely the one that implements the simple count model, since its results should only be used as an indicator of abundances and because that model is extremely simplified and should not be used for real-life abundance measures right now. In the future, when technology possibly enables knowing all read origins, it could be of much greater use. I would also mention that `isomorph` requires no auxiliary indexing files or preprocessing steps in the current version. RSEM, for example, uses `rsem-prepare-reference` as a preprocessing step that produces index files for reconstructed transcripts. These index files are then used in the main algorithm.

## 5.2. Discussion

The results from table 5.1 show that the performance of `isomorph` is on average better on single than on paired-end reads. For single reads, up to 67% of transcripts were correctly quantified. When using the RSEM simulator, paired-read results were very bad, while with polyester, they were comparable to single-end results. This indicates a possible "outlier" data set, something that `isomorph` cannot handle well right now. It is observable that the results using polyester are much more comparable across runs. The reason for this is probably the fact that all three polyester runs were done on the same transcript set, while RSEM simulator was run on three different sets of isoforms, coming from different Yeast samples and experiments. Also the best results according to the median value of  $\delta$  are for CT1 and CT2 datasets, where at least fifty percent of transcripts are quantified with less than seven percent error. Consequently, it is obvious that for some data, `isomorph` performs much better than for the other and it needs to be further investigated if this is due to a too specific/too stringent probabilistic model or some other implementation details. Also, the table implies that its performance is comparable to IsoEM and RSEM. A curious thing is that both IsoEM and RSEM performed extremely poorly on CT3 set where paired-end reads were sampled using polyester. It is possible that their models are too advanced for that simple dataset and that `isomorph`'s simplicity is better suited for it, or a low number of reads contributes to them not being able to correctly infer what is correct in the data. I have noticed that bowtie did provide a very large number of mappings for this dataset, and it possibly influenced them and skewed their results.

Figure 5.1 shows a good characteristic of `isomorph` having linear time complexity with regards to the number of reads. However, noticeable is the fact that it is

far slower than both RSEM and IsoEM. Its performance for paired-end data indicates that it can and should be significantly sped up. IsoEM does look like the fastest, but this is due to the fact that it starts multiple threads and I did not find an option to force it to run on only one thread (while both RSEM and `isomorph` ran on one thread).

In the future, it will be important to significantly improve the paired-end model to produce much better estimates, as well as single and paired-end time performance. Possible options to add are gene expression analysis and statistical testing of gene expression difference across samples, calculating confidence intervals or doing Gibbs sampling [36, 37]. Adding read length distribution modelling should be considered if reads from other sequencing platforms are to be used with `isomorph`. Support for multiple mappers and different input formats (GTF annotation) will be added.

## 6. Conclusion

In this work I have focused on RNA-Seq, next-generation sequencing applied to analysis of RNA. RNA-Seq has been more and more used as a de facto go to method of RNA investigation in the past years. The goal of the thesis was to implement RNA transcript abundance estimation for a sample using RNA-Seq reads. Estimating abundances is important for multiple applications, including the analysis of gene expression and relation to different states of a cell or an organism. The transcript reconstruction and abundance estimation problem usually can be approached in two ways, doing it with read mappings to a reference genome or assembling possible isoforms *de novo* and proceeding with further downstream analysis. Both approaches have their advantages and problems and I have compared and illustrated the use-cases for both.

*De novo* assembly and analysis was the approach used as a fundamental idea in this thesis. When reference genomes or transcriptomes are not available, or are poorly understood, this is the only way of having as clear as possible look into the sample being analysed. `isomorph`, a software presented here, is an implementation of a robust probabilistic model that uses reads mapped to *de novo* assembled transcripts to maximize a likelihood function with regards to relative abundances. The tests have shown that, overall, `isomorph` performs relatively well, but does have a significant challenge processing paired-end reads, which needs to be looked further into. It also shows linear time complexity with regards to the number of reads.

The future of `isomorph` will include advancing the model, and probably significantly deviating from the one described herein, with the goal of improving its results and performance, especially for paired-end data. Additional options for better and more flexible RNA analysis will be added.

## REFERENCES

- [1] O. Morozova and M. A. Marra, "Applications of next-generation sequencing technologies in functional genomics," *Genomics*, 2008.
- [2] R. Bumgarner, "DNA microarrays: Types, Applications and their future," *Current Protocols in Molecular Biology*, 2013. [Online]. Available: <http://onlinelibrary.wiley.com/doi/10.1002/0471142727.mb2201s101/abstract>
- [3] C. Trapnell, B. A. Williams, G. Pertea, A. Mortazavi, G. Kwan, M. J. van Baren, S. L. Salzberg, B. J. Wold, and L. Pachter, "Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation," *Nature Biotechnology*, 2010.
- [4] Z. Wang, M. Gerstein, and M. Snyder, "RNA-Seq: a revolutionary tool for transcriptomics," *Nature Reviews Genetics*, 2009.
- [5] U. Nagalakshmi, K. Waern, and M. Snyder, "RNA-Seq: A Method for Comprehensive Transcriptome Analysis," *Current Protocols in Molecular Biology*, 2010.
- [6] National Institutes of Health. (Accessed June 15, 2015) Genetics Home Reference. [Online]. Available: <http://ghr.nlm.nih.gov/>
- [7] M. B. Gerstein, C. Bruce, J. S. Rozowsky, D. Zheng, J. Du, J. O. Korb, O. Emanuelsson, Z. D. Zhang, S. Weissman, and M. Snyder, "What is a gene, post-ENCODE? History and updated definition," *Genome Research*, 2007.
- [8] H. Keren, G. Lev-Maor, and G. Ast, "Alternative splicing and evolution: diversification, exon definition and function," *Nature Reviews Genetics*, 2010.
- [9] D. L. Black, "Mechanisms of alternative pre-messenger RNA splicing," *Annual Review of Biochemistry*, 2003.

- [10] K. Gourolain, A. Soulier, B. Pellegrin, M. Bouvier-Alias, C. Hézode, F. Darthuy, J. Rémiré, J.-M. Pawlotsky, , and the DITTO Group, “Dynamic Range of Hepatitis C Virus RNA Quantification with the Cobas Ampliprep-Cobas Amplicor HCV Monitor v2.0 Assay,” *Journal of Clinical Microbiology*, 2005.
- [11] D. Shalon, S. J. Smith, and P. O. Brown, “A DNA Microarray System for Analyzing Complex DNA Samples Using Two-color Fluorescent Probe Hybridization,” *Genome Research*, 1996.
- [12] S. Zhao, W.-P. Fung-Leung, A. Bittner, K. Ngo, and X. Liu, “Comparison of RNA-Seq and Microarray in Transcriptome Profiling of Activated T Cells,” *PLOS One*, 2014. [Online]. Available: <http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0078644>
- [13] J. Derisi, L. Penland, P. Brown, M. Bittner, P. Meltzer, M. Ray, Y. Chen, Y. Sun, and J. Trent, “Use of a cDNA microarray to analyse gene expression patterns in human cancer,” *Nature Genetics*, 1996.
- [14] S. Fodor, J. Read, M. Pirrung, L. Stryer, A. Lu, and D. Solas, “Light-directed, spatially addressable parallel chemical synthesis,” *Science*, 1991.
- [15] A. Pease, D. Solas, E. Sullivan, M. Cronin, C. Holmes, and S. Fodor, “Light generated oligonucleotide arrays for rapid DNA sequence analysis,” *Proceedings of the National Academy of Sciences, USA*, 1994.
- [16] J. Ferguson, F. Steemers, and D. Walt, “High-density fiber-optic DNA random microsphere array,” *Analytical chemistry*, 2000.
- [17] K. Michael, L. Taylor, S. Schultz, and D. Walt, “Randomly ordered addressable high-density optical sensor arrays,” *Analytical chemistry*, 1998.
- [18] F. Steemers, J. Ferguson, and D. Walt, “Screening unlabeled DNA targets with randomly ordered fiberoptic gene arrays,” *Nature biotechnology*, 2000.
- [19] D. Walt, “Techview: molecular biology. Bead-based fiber-optic arrays,” *Science*, 2000.
- [20] J. C. Marioni, C. E. Mason, S. M. Mane, M. Stephens, and Y. Gilad, “RNA-Seq: An assessment of technical reproducibility and comparison with gene expression arrays,” *Genome Research*, 2008.



- [21] R. D. Morin, M. Bainbridge, A. Fejes, M. Hirst, M. Krzywinski, T. J. Pugh, H. McDonald, R. Varhol, S. J. Jones, and M. A. Marra, “Profiling the HeLa S3 transcriptome using randomly primed cDNA and massively parallel short-read sequencing,” *BioTechniques*, 2008.
- [22] K.-O. Mutz, A. Heilkenbrinker, M. Lönne, J.-G. Walter, and F. Stahl, “Transcriptome analysis using next-generation sequencing,” *Current opinion in Biotechnology*, 2013.
- [23] Illumina, “An Introduction to Next-Generation Sequencing Technology,” Illumina Inc., Tech. Rep., 2015.
- [24] J. A. Martin and Z. Wang, “Next-generation transcriptome assembly,” *Nature Reviews*, 2011.
- [25] M. A. Busby, C. Stewart, C. A. Miller, K. R. Grzeda, and G. T. Marth, “Scotty: a web tool for designing RNA-Seq experiments to measure differential gene expression,” *Bioinformatics*, 2013.
- [26] A. Mortazavi, B. A. Williams, K. McCue, L. Schaeffer, and B. Wold, “Mapping and quantifying mammalian transcriptomes by RNA-Seq,” *Nature Methods*, 2008.
- [27] W. R. Francis, L. M. Christianson, R. Kiko, M. L. Powers, N. C. Shaner, and S. H. D. Haddock, “A comparison across non-model animals suggests an optimal sequencing depth for de novo transcriptome assembly,” *BMC Genomics*, 2013.
- [28] E. Bernard, L. Jacob, J. Mairal, and J.-P. Vert, “Efficient RNA isoform identification and quantification from RNA-Seq data with network flows,” *Bioinformatics*, 2014.
- [29] H. Jiang and W. H. Wong, “Statistical inferences for isoform expression in RNA-Seq,” *Bioinformatics*, 2009.
- [30] M. Nicolae, S. Mangul, I. I. Mandoiu, and A. Zelikovsky, “Estimation of alternative splicing isoform frequencies from RNA-Seq data,” *Algorithms for Molecular Biology*, 2011.
- [31] M. Guttman, M. Garber, J. Z. Levin, J. Donaghey, J. Robinson, X. Adiconis, L. Fan, M. J. Koziol, A. Gnirke, C. Nusbaum, J. L. Rinn, E. S. Lander, and A. Regev, “Ab initio reconstruction of cell type-specific transcriptomes in mouse

- reveals the conserved multi-exonic structure of lincRNAs,” *Nature Biotechnology*, 2010.
- [32] M. G. Grabherr, B. J. Haas, M. Yassour, J. Z. Levin, D. A. Thompson, I. Amit, X. Adiconis, L. Fan, R. Raychowdhury, Q. Zeng, Z. Chen, E. Mauceli, N. Hacohen, A. Gnirke, N. Rhind, F. di Palma, B. W. Birren, C. Nusbaum, K. Lindblad-Toh, N. Friedman, and A. Regev, “Full-length transcriptome assembly from RNA-Seq data without a reference genome,” *Nature Biotechnology*, 2011.
- [33] M. H. Schulz, D. R. Zerbino, M. Vingron, and E. Birney, “Oases: robust de novo RNA-seq assembly across the dynamic range of expression levels,” *Bioinformatics*, 2012.
- [34] Y. Xie, G. Wu, J. Tang, R. Luo, J. Patterson, S. Liu, W. Huang, G. He, S. Gu, S. Li, X. Zhou, T.-W. Lam, Y. Li, X. Xu, G. K.-S. Wong, and J. Wang, “SOAPdenovo-Trans: de novo transcriptome assembly with short RNA-Seq reads,” *Bioinformatics*, 2014.
- [35] G. Robertson, J. Schein, R. Chiu, R. Corbett, M. Field, S. D. Jackman, K. Mungall, S. Lee, H. M. Okada, J. Q. Qian, M. Griffith, A. Raymond, N. Thiessen, T. Cezard, Y. S. Butterfield, R. Newsome, S. K. Chan, R. She, R. Varhol, B. Kamoh, A.-L. Prabhu, A. Tam, Y. Zhao, R. A. Moore, M. Hirst, M. A. Marra, S. J. M. Jones, P. A. Hoodless, and I. Birol, “De novo assembly and analysis of RNA-seq data,” *Nature Methods*, 2010.
- [36] B. Li, V. Ruotti, R. M. Stewart, J. A. Thomson, and C. N. Dewey, “RNA-Seq gene expression estimation with read mapping uncertainty,” *Bioinformatics*, 2010.
- [37] B. Li and C. N. Dewey, “RSEM: accurate transcript quantification from RNA-Seq data with or without a reference genome,” *BMC Bioinformatics*, 2011.
- [38] A. Roberts and L. Pachter, “Streaming fragment assignment for real-time analysis of sequencing experiments,” *Nature Methods*, 2013.
- [39] D. Bottomly, N. A. R. Walter, J. E. Hunter, P. Darakjian, S. Kawane, K. J. Buck, R. P. Searles, M. Mooney, S. K. McWeeney, and R. Hitzemann, “Evaluating Gene Expression in C57BL/6J and DBA/2J Mouse Striatum Using RNA-Seq and Microarrays,” *PLOS One*, 2011. [Online]. Available: <http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0017820>

- [40] W. Zhang, J. Ferguson, S. M. Ng, K. Hui, G. Goh, A. Lin, E. Esplugues, R. A. Flavell, C. Abraham, H. ZHao, and J. H. Cho, “Effector CD4+ T Cell Expression Signatures and Immune-Mediated Disease Associated Genes,” *PLOS One*, 2012. [Online]. Available: <http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0038510>
- [41] A. Sirbu, G. Kerr, M. Crane, and H. J. Ruskin, “RNA-Seq vs Dual- and Single-Channel Microarray Data: Sensitivity Analysis for Differential Expression and Clustering,” *PLOS One*, 2012. [Online]. Available: <http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0050986>
- [42] X. Fu, N. Fu, S. Guo, Z. Yan, Y. Xu, H. Hu, C. Menzel, W. Chen, Y. Li, R. Zeng, and P. Khaitovich, “Estimating accuracy of RNA-Seq and microarrays with proteomics,” *BMC Genomics*, 2009.
- [43] A. Roberts, C. Trapnell, J. Donaghey, J. L. Rinn, and L. Pachter, “Improving RNA-Seq expression estimates by correcting for fragment bias,” *Genome Biology*, 2011.
- [44] C. Trapnell, D. G. Hendrickson, M. Sauvageau, L. Goff, J. L. Rinn, and L. Pachter, “Differential analysis of gene regulation at transcript resolution with RNA-seq,” *Nature Biotechnology*, 2013.
- [45] B. X. Lu, Z. B. Zeng, and T. L. Shi, “Comparative study of de novo assembly and genome-guided assembly strategies for transcriptome reconstruction based on RNA-Seq,” *Science China Life Sciences*, 2013.
- [46] C. Trapnell, L. Pachter, and S. L. Salzberg, “TopHat: discovering splice junctions with RNA-Seq,” *Bioinformatics*, 2009.
- [47] A. Celaj, J. Markle, J. Danska, and J. Parkinson, “Comparison of assembly algorithms for improving rate of metatranscriptomic functional annotation,” *Microbiome*, 2014.
- [48] T. Namiki, T. Hachiya, H. Tanaka, and Y. Sakakibara, “MetaVelvet: an extension of Velvet assembler to de novo metagenome assembly from short sequence reads,” *Nucleic Acids Research*, 2012.
- [49] H. C. M. Leung, S.-M. Yiu, J. Parkinson, and F. Y. L. Chin, “IDBA-MT: De Novo Assembler for Metatranscriptomic Data Generated from Next-Generation Sequencing Technology,” *Journal of Computational Biology*, 2013.

- [50] K. Clarke, Y. Yang, R. Marsh, L. Xie, and K. K. Zhang, “Comparative analysis of *de novo* transcriptome assembly,” *Science China Life Sciences*, 2013.
- [51] B. Chevreux, T. Pfisterer, B. Drescher, A. J. Driesel, W. E. G. Müller, T. Wetter, and S. Suhai, “Using the miraEST Assembler for Reliable and Automated mRNA Transcript Assembly and SNP Detection in Sequenced ESTs,” *Genome Research*, 2004.
- [52] J. A. Bilmes, “A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models,” University of California Berkeley, Tech. Rep., 1998.
- [53] G. P. Wagner, K. Kin, and V. J. Lynch, “Measurement of mRNA abundance using RNA-seq data: RPKM measure is inconsistent among samples,” *Theory in Biosciences*, 2012.
- [54] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum-likelihood from incomplete data via the em algorithm,” *Journal of the Royal Statistical Society. Series B (Methodological)*, 1977.
- [55] R. A. Redner and H. F. Walker, “Mixture Densities, Maximum Likelihood and the Em Algorithm,” *SIAM Review*, 1984.
- [56] C. B. Do and S. Batzoglou, “What is the expectation maximization algorithm?” *Nature Biotechnology*, 2008.
- [57] S. Zafeiriou. (Accessed Jun 22, 2015) Tutorial on expectation maximization (example). [Online]. Available: [http://ibug.doc.ic.ac.uk/media/uploads/documents/expectation\\_maximization-1.pdf](http://ibug.doc.ic.ac.uk/media/uploads/documents/expectation_maximization-1.pdf)
- [58] S. Borman, “The expectation maximization algorithm: A short tutorial,” online resource from Carnegie Mellon University, accessed June 22, 2015. [Online]. Available: [http://www.cs.cmu.edu/~dgovinda/pdf/recog/EM\\_algorithm-1.pdf](http://www.cs.cmu.edu/~dgovinda/pdf/recog/EM_algorithm-1.pdf)
- [59] “Basic probabilistic inequalities,” accessed June 22, 2015. [Online]. Available: <http://www.statlect.com/subon/inequal.htm>
- [60] “Expectation-maximization,” accessed June 22, 2015. [Online]. Available: <http://www.cs.utah.edu/~piyush/teaching/EM.pdf>
- [61] B. Langmead and S. L. Salzberg, “Fast gapped-read alignment with Bowtie 2,” *Nature Methods*, 2012.

- [62] A. Döring, D. Weese, T. Rausch, and K. Reinert, “SeqAn An efficient, generic C++ library for sequence analysis,” *BMC Bioinformatics*, vol. 9, no. 11, 2008.
- [63] H. Li and R. Durbin, “Fast and accurate short read alignment with Burrows–Wheeler transform,” *Bioinformatics*, 2009.
- [64] H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, R. Durbin, and 1000 Genome Project Data Processing Subgroup, “The sequence alignment/map format and samtools,” *Bioinformatics*, 2009.
- [65] A. C. Frazee, A. E. Jaffe, B. Langmead, and J. T. Leek, “Polyester : simulating rna-seq datasets with differential transcript expression,” *Bioinformatics*, 2015.

## Splice isoform identification from transcript graphs

### Abstract

Providing a measure of abundances of different RNA transcripts in a sample can be very important in many different biological or medical studies. Next-generation sequencing is becoming a technology of choice for providing fundamental data for such analyses. In this work, I have presented an RNA abundance estimation model that uses next-generation RNA-Seq reads to estimate optimal relative transcript abundance values. `isomorph`, a software implementation of this method is presented and explained. `isomorph` tries to maximize a likelihood function with regards to isoform abundance levels to provide an optimal estimate of relative transcript abundances in a sample. Testing has shown that for single-end reads, `isomorph` performs well, while paired-end data performance should be more improved. It is available at <https://github.com/darxsys/isomorph>.

**Keywords:** rna-seq, abundance estimation, em algorithm

## Identifikacija RNA izoformi iz grafa transkripata

### Sažetak

Mjerenje relativnih količina RNA transkripata u uzorku može biti vrlo važno za različite primjene u biologiji i medicini. Sekvenciranje sljedeće generacije postaje primarno korištena tehnologija za omogućavanje takvih analiza pružanjem osnovnih podataka o uzorku. U ovome radu, prezentirao sam statistički model procjene količine RNA koji koristi očitavanja sekvenciranja sljedeće generacije s ciljem pronalaženja optimalnih vrijednosti relativnih količina. `isomorph`, programska implementacija ovog modela je opisana u radu. `isomorph` pokušava pronaći maksimum funkcije izglednosti po parametrima koji predstavljaju relativne nivoe prisutnosti transkripata u uzorku. Testiranje je pokazalo da za očitavanja bez parova, `isomorph` ima zadovoljavajuću točnost, dok za očitavanja u parovima pokazuje potrebu za dodatnim poboljšanjima performansi. Dostupan je na <https://github.com/darxsys/isomorph>.

**Ključne riječi:** rna-sekvenciranje, procjena količine, em algoritam