ABSTRACT

Title of Dissertation: COMPUTATIONAL APPROACHES FOR IMPROVING THE ACCURACY AND EFFICIENCY OF RNA-SEQ ANALYSIS

Hirak Sarkar
Doctor of Philosophy, 2020

Dissertation Directed by: Professor Robert Patro
Department of Computer Science

The past decade has seen tremendous growth in the area of high throughput sequencing technology, which simultaneously improved the biological resolution and subsequent processing of publicly-available sequencing datasets. This enormous amount of data also calls for better algorithms to process, extract and filter useful knowledge from the data. In this thesis, I concentrate on the challenges and solutions related to the processing of bulk RNA-seq data. An RNA-seq dataset consists of raw nucleotide sequences, drawn from the expressed mixture of transcripts in one or more samples. One of the most common uses of RNA-seq is obtaining transcript or gene level abundance information from the raw nucleotide read sequences and then using these abundances for downstream analyses such as differential expression. A typical computational pipeline for such processing broadly involves two steps: assigning reads to the reference sequence through alignment or mapping algorithms, and subsequently quantifying such assignments to obtain the expression of the reference transcripts or genes. In practice, this two-step process poses multitudes of

challenges, starting from the presence of noise and experimental artifacts in the raw sequences to the disambiguation of multi-mapped read sequences. In this thesis, I have described these problems and demonstrated efficient state-of-the-art solutions to a number of them.

The current thesis will explore multiple uses for an alternate representation of an RNA-seq experiment encoded in equivalence classes and their associated counts. In this representation, instead of treating a read fragment individually, multiple fragments are simultaneously assigned to a set of transcripts depending on the underlying characteristics of the read-to-transcript mapping. I used the equivalence classes for a number of applications in both single-cell and bulk RNA-seq technologies. By employing equivalence classes at cellular resolution, I have developed a droplet-based single-cell RNA-seq sequence simulator capable of generating tagged end short read sequences resembling the properties of real datasets. In bulk RNA-seq, I have utilized equivalence classes to applications ranging from data-driven compression methodologies to clustering de-novo transcriptome assemblies. Specifically, I introduce a new data-driven approach for grouping together transcripts in an experiment based on their inferential uncertainty. Transcripts that share large numbers of ambiguously-mapping fragments with other transcripts, in complex patterns, often cannot have their abundances confidently estimated. Yet, the total transcriptional output of that group of transcripts will have greatly-reduced inferential uncertainty, thus allowing more robust and confident downstream analysis. This approach, implemented in the tool terminus, groups together transcripts in a data-driven manner. It leverages the equivalence class factorization to quickly

identify transcripts that share reads and posterior samples to measure the confidence of the point estimates. As a result, terminus allows transcript-level analysis where it can be confidently supported, and derives transcriptional groups where the inferential uncertainty is too high to support a transcript-level result.

COMPUTATIONAL APPROACHES FOR IMPROVING THE ACCURACY AND EFFICIENCY OF RNA-SEQ ANALYSIS

by

Hirak Sarkar

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2020

Advisory Committee:
Professor Rob Patro, Chair/Advisor
Professor Mihai Pop
Professor Hector Corrado Bravo
Professor Stephen Mount
Professor John Dickerson

Dedication

To my parents, brother and my beloved fiancée

Acknowledgments

First of all, I want to thank my advisor, Prof. Robert Patro, who supported me immensely throughout the entire course of my Ph.D. I was one of the first students that Rob had taken under his wings. I believe interesting scientific exchange can only occur in an environment where one has the sense of intellectual freedom. From the inception of our lab, Rob made sure to create an environment where the advisor-trainee relationship has turned into collaboration.

I am tremendously grateful to my committee members Hector Corrada Bravos, Mihai Pop, Steve Mount, and John Dickerson at the University of Maryland. I want to thank numerous collaborators whom I worked with over the years, Mike Love, Charlotte Soneson, Adam Siepel, Steve Skiena, Bruce Futcher, Mike Ferdman, and many others. I also want to thank the supportive academic community that helped me tremendously in many difficult times.

This thesis would not have been possible without the help of family, friends and colleagues who remained my life-support for many years. I want to thank my friends at COMBINE-lab, we started as a small group, but it has grown considerably since then. I want to thank my friends at Stony Brook and Maryland, with whom I spent numerous sleepless nights before the deadlines. I want to thank my parents and my brother, who supported me from thousands of miles away. I want to thank my fiancee Snigdha for all her love and support.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ARD	**A**bsolute **R**elative **D**ifference
BLAST	**B**asic **L**ocal **A**lignment **S**earch **T**ool
BWT	**B**urrows-**W**heeler **T**ransform
DFS	**D**epth-**F**irst **S**earch
DP	**D**ynamic-**P**rogram
Indel	**In**sertion (and) **D**eletion
MARD	**M**ean **A**bsolute **R**elative **D**ifference
NGS	**N**ext **G**eneration **S**equencing
SA	**S**uffix **A**rray
SGS	**S**econd **G**eneration **S**equencing
SIMD	**S**ingle **I**nstruction **M**ultiple **D**ata
SMEM	**S**uper **M**aximal **E**xact **M**atch
T-DBG	**T**ranscriptome - **d**e **B**ruijn **G**raph
TPEF	**T**rue **P**ositive **E**rror **F**raction
TPME	**T**rue **P**ositive **M**edian **E**rror
VBEM	**V**ariational **B**ayesian - **E**xpectation **M**aximization
wMARD	**w**eighted **M**ean **A**bsolute **R**elative **D**ifference
dscRNA-seq	**d**roplet based **s**ingle **c**ell **RNA**-seq
CB	**C**ellular **B**arcode
UMI	**U**nique **M**olecule **I**dentifier
PUG	**P**arsimonious **U**MI **G**raph

Chapter 1: Introduction

1.1 Landscape of genomic data

There has been an explosion [3] in the production of publicly available genomic and transcriptomic data in the last decade. To mention a few in the transcriptomic world the sequencing technology ranges from RNA transcription (PRO-Seq, GRO-Seq) to RNA-protein interactions (RIP-Seq, Pol II CLIP), RNA Structure (SHAPE-Seq) to low level RNA detection (bulk RNA-Seq and single cell RNA-Seq). The series of steps that lead to protein generation in a cell (also known as central dogma of biology) involves replication, transcription and translation. The mentioned diaspora of different sequencing technology measures the process of transcription and processing at various levels, revealing bits of pieces of information about this complex biological process. We further concentrate on the RNA-Seq technology which is a widely used next generation technology for measuring the presence of matured messenger RNAs.

1.2 RNA-seq sequencing assay

Since the advent of RNA-seq sequencing technology on 2008, [4, 5, 6, 7, 8], it has become the de facto standard for measuring transcript expression in both biological and medical research. RNA-seq has brought marked improvement in generating high throughput data that enables the detection of rare transcripts, discovering novel isoforms, single nucleotide variants, indels etc. RNA-seq is also widely used for measuring differential expression and detect novel splicing. It overcome the limitations of other tag-based assays (EST libraries), where short tags were often indistinguishable, thereby making the annotation of structure extremely difficult. While RNA-seq has revolutionized the analysis of transcriptomes and the associated structure, it has also posed several computational challenges ranging from storing the large amount of sequences (often in the order of tens of millions for one sample) to alignment and quantification of such dataset. Before delving deeper into those specific questions and solution presented in this report, I would review the RNA-seq protocol briefly.

1.2.1 Brief overview of an RNA-seq protocol for measuring mature messenger RNAs

A typical RNA-seq experiment starts with an experimental design, where a number of important factors are determined such as, number of technical and biological replicates, depth of sequencing, experimental complexity (nature of groups).

These important factors are determined by the goal of the experiment. The second step involves the isolation of the cellular RNAs. It's a complex biochemical process involving several intermediate steps and followed by the extraction of mature messenger RNAs (mRNAs). These extracted mRNA molecules are generally longer than the typical length (30-400 bp) that widely used sequencing machines (Illumina IG, SOLiD, Roche 454) can sequence. To fit this length requirement, the mRNA molecules undergo fragmentation. RNA molecules are generally unstable, and most sequencing machines can not sequence RNA therefore they are reverse-transcribed to equivalent double stranded cDNA. The appropriately sized fragmented cDNA libraries are attached to adaptors. Adaptors can be attached to one or to both the ends of the fragments. Each molecule that is attached to an adaptor can be amplified by polymerized chain reaction (PCR). This increases the total number of fragments. The sequencing machine sequences the prepared pool of molecules and produces the cDNA sequences, producing a nucleotide level resolution of the mRNA sequences.

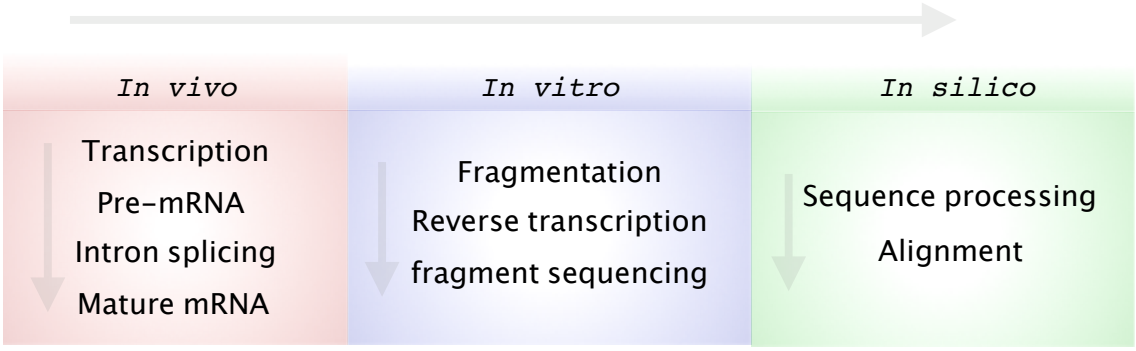


Figure 1.1: Pipeline of a typical RNA-seq protocol

The analysis part of the RNA-seq dataset involves computational tasks such as mapping/alignment. Figure 1.1 shows a schematic view of the complete pipeline. In

the current report I would discuss some computational challenges related to RNA-seq and starts from the raw read sequences obtained from the sequencing machine.

The computational methods that deal with RNA-seq datasets accept raw read sequence in the form of fastq file(s). Depending on the sequencing protocol, the read sequences can be either paired end or single end. In paired-end (PE) format the cDNA library is sequenced from both the directions, whereas in single-end (SE), sequencing starts from only one end. PE reads are more informative in the sense that it effectively encodes the fragment lengths that are processed and therefore provides more information in the downstream analysis. In a stranded protocol, the information about the coding strand is preserved and can be used later to know if the expressed RNA was originated from forward or reverse complement strand of the given genomic position.

Since the advent of RNA-seq, the throughput and other characteristics have evolved greatly over time. The newer sequencing machines (e.g. Illumina's MiSeq) simultaneously enabled longer read lengths and higher throughput for the same cost. fig. 1.2 from Van Den Berge et al. [1] depicts the change of RNA-seq data in terms of read length, throughput and sample size. Iobserve that although there is a considerable shift in the read length and read depth, the number of samples remained similar (median around 8).

Recently there have been newer technologies emerging that are built following the same pipeline as RNA-seq, but fundamentally change the scope of the experiment. The most prominent ones are long read sequencing technology and single-cell RNA-seq technology. Pacbio Biosciences and Oxford Nanopore Technologies en-

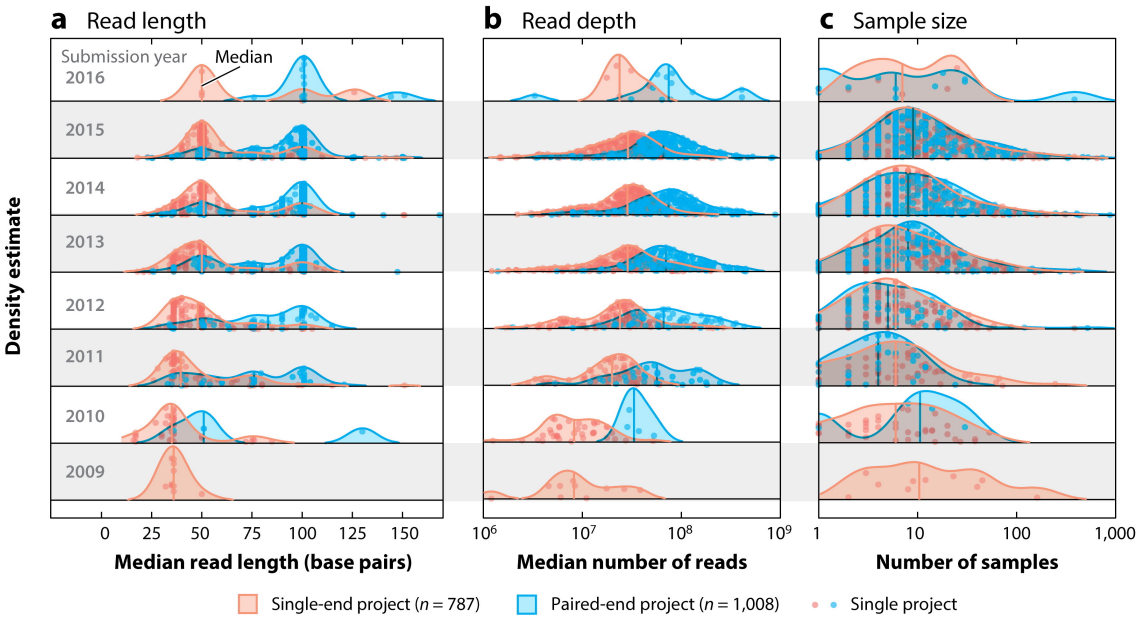


Figure 1.2: Evolution of RNA-seq protocol over the years taken from Van Den Berge et al. [1]

abled sequencing of full length transcript molecules but with diminished quality when compared to standard illumina sequencing. On the other hand scRNA-seq has also become extraordinarily popular very recently enabling the transcriptome analysis at the level of single cells.

1.2.2 Where to map, genome vs transcriptome? Align or map?

Since the introduction of high-throughput, short read sequencing technologies, many algorithms and tools have been designed to tackle the problem of aligning short sequenced reads to a reference genome or transcriptome accurately and efficiently. The choice of the reference sequence also dictates the choice of tool and thereby the computational requirement for running the analysis. Broadly (skipping subtle biological complexities) a gene can be perceived as a long thread of nucleotides,

that consists of two different types of sequences, the regions that are retained in the mature RNA products: *exons*, separated by regions of non-coding sequences known as *introns*. Most eukaryotic organisms (rarely some prokaryotic) undergo a mechanism known as splicing that determines which exons would be glued up together and end up in the final transcript isoform, in other words it also determines which introns would be skipped in the final transcript isoform. The right hand side of fig. 1.3 shows a generic view of isoform formation from a gene. The splicing junctions are shown in curved lines. This shows that the transcriptome sequence carries more redundancy at the sequence level than that of the genome. Reads that originates from the shared sequence of the isoforms (in fig. 1.3 the first exon) are in effect indistinguishable from each other. This structural complexity along with other challenges makes the problem of read mapping a complicated one. Figure 1.3 shows one such case where a read maps to two different isoforms, while in terms of genomic coordinates, it originates from one genomic position. The decision of choosing transcriptome vs genome as reference sequence depends on various different factors. A genomic index can be computationally expensive to store and query due to larger size, on the other hand, although transcriptomic index is much smaller in size, it might lead to more ambiguous mapping to the candidate transcript isoforms from one gene.

Although, so far I have used the terms "mapping" and "alignment" equivalently, in computational terms they are quite different. While alignment generally refers to heuristically solving the dynamic programming problem of string alignment, the objective of mapping is to simply find the potential locations of exact or almost

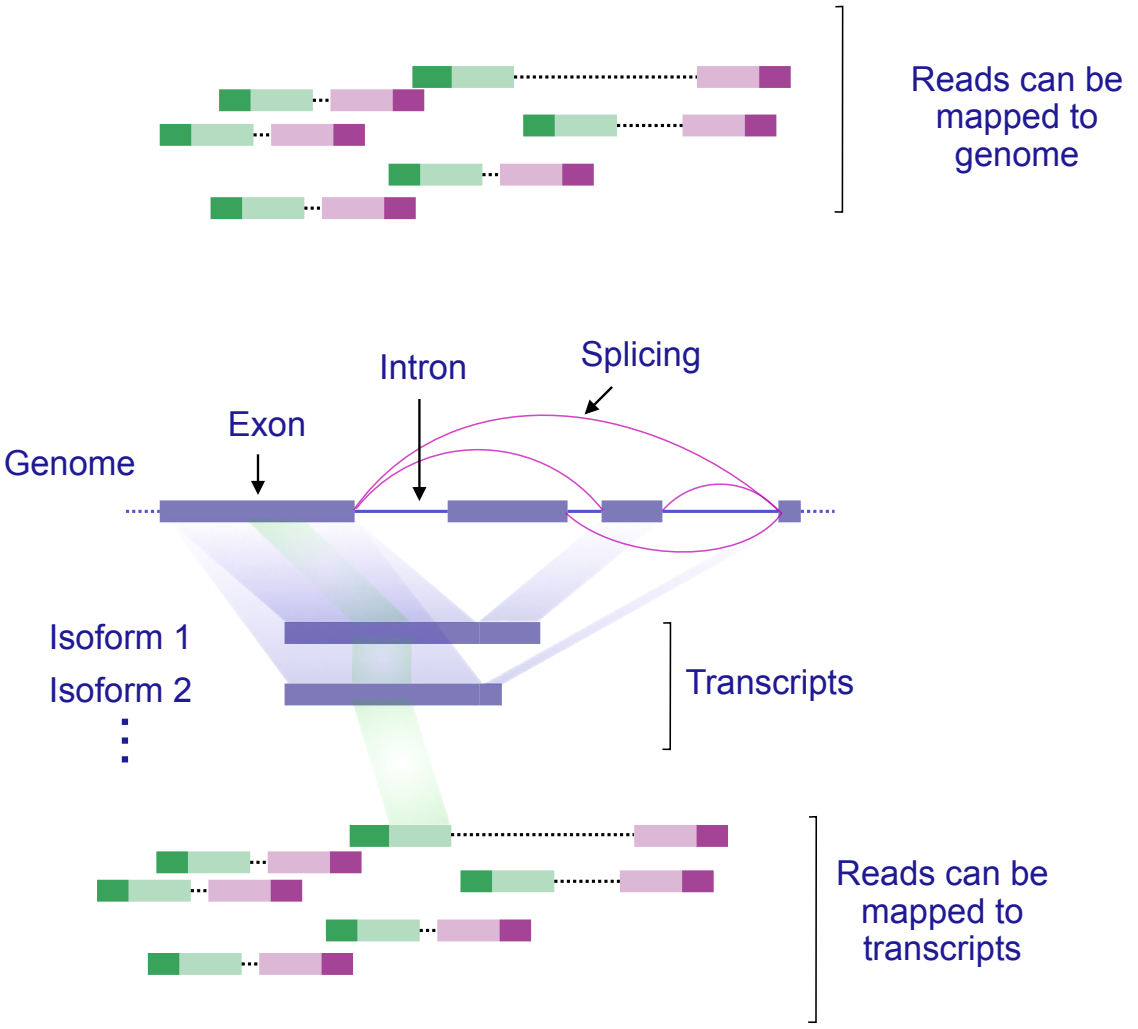


Figure 1.3: Schematic view of paired end reads and structure of reference transcritome and mapping

exact matches in the reference. The trade-off between the two paradigms often boils down the question of accuracy vs efficiency. While there exist "full-sensitivity" aligners (e.g. RazerS3 [9], Masai [10]) which guarantee to find all reference positions within a given edit-distance threshold of a read sequence, the most widely-used tools employ heuristic strategies to enable much faster alignment of reads in the typical case (i.e., only a small number of easy-to-find candidate locations exist for each alignment). The common procedure followed by these tools for aligning reads can

be divided into two major steps. The first is finding potential alignment locations for the read using a pre-processed index that is generated from the reference genome or transcriptome. Then, in the second step, the potential locations are filtered, and reads are aligned to the positions that pass the initial filtering, based on a variety of heuristics. The exact method for generating the initial index varies for each tool. For example, tools like Bowtie [11], Bowtie2 [12], BWA [13], and BWA-mem [14] use Burrows-Wheeler transformation (BWT) based indices (BWT is reversible permutation of a string, generally used in compression), whereas, k-mer based indices are used by tools such as Subread-aligner [15], Maq [16], SNAP [17], and GMAP and GSNAP [18].

Similarly, the heuristic for choosing the most probable locations is also different. However, each method is based on the principle of trying to find the reference loci that support the best (or near-best) alignment score between the read and the reference. Repeating this for a large number of reads comes with a considerable cost, in terms of computation. Some tools, like STAR [19], considerably speed up the alignment process by combining efficient heuristics with data structures (like the uncompressed suffix array) that trade working memory for exact pattern lookup speed. Recently, tools like HISAT [20] have also demonstrated that cache-friendly compressed indices (the hierarchical FM index in this case) can provide similarly efficient pattern search, even with a very moderate memory budget. The alignment of sequenced reads to the reference is the first step in pipelines leading to various downstream studies, such as estimation of transcript abundances and differential expression analysis, calculation of magnitude of splicing change [21, 22], and detection

of fusion events [23, 24].

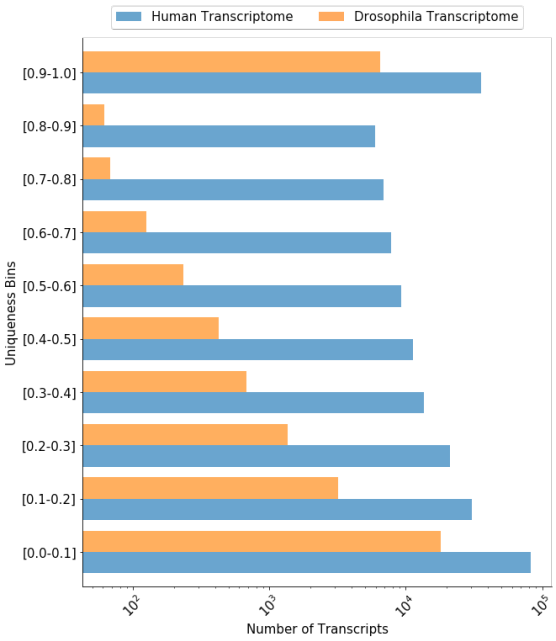
While alignment is a staple of many genomic analyses, it sometimes represents more information than is actually necessary to address the analysis at hand. For example, recent tools like *Sailfish* [25] (including its quasi-mapping-based variant [26]), *kallisto* [27], and *salmon* [28], demonstrate that much of the information provided by aligners may be unnecessary for accurate transcript quantification. By avoiding traditional alignment, these tools are much faster than their alignment-based counterparts. Furthermore, by building the mapping phase of the analysis directly into the quantification task, they dispense with the need to write, store, and read, large intermediate alignment files. However, these "mapping-based" tools, while highly-efficient, have the disadvantage of potentially losing sensitivity or specificity in certain adversarial cases where alignment-based methods would perform well. For example, in the presence of paralogous genes, with high sequence similarity, there is an increased probability that the mapping strategies employed by such tools, and the efficient heuristics upon which they rely, will mis-map reads between the paralogs (or return a more ambiguous set of mapping locations than an aligner, which expends effort to verify the returned alignments, would have) [29]. Similarly, in the case of *de novo* assemblies, poorly assembled contigs may have a larger number of mis-mapped reads due to lower sensitivity (here, the issue would be primarily due to aberrant exact matches masking the true origin of a read).

I presented two different solutions in the domain of alignment. In Sarkar et al. [30] I presented an intermediate approach that falls between the midway of alignment and mapping. The key of that work is to selectively align a special set of

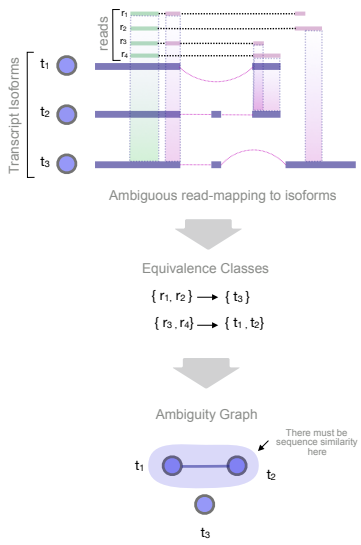
reads that needs a closer inspection while mapping reads. I have demonstrated that to a large extent the sensitivity of an aligner can be matched without compromising the superior speed a mapper.

1.3 Exploring the redundancy in RNA-seq

Figure 1.3 shows that there is a significant amount of sequence shared between the isoforms by the virtue of shared exons.



Number of transcripts in stratified by sequence uniqueness. The [0.0 - 0.1] bins suggest that the sequences in this bin has at most 10% unique k-mers therefore they are the least unique transcripts. On the other hand [0.9-1] bin contains the most unique transcripts and likewise.



The highly non-unique structure can be represented as a graph in a data-driven manner

Figure 1.4: Formation of equivalence classes from shared mapped reads

This redundancy is present both in transcriptome and genomic sequence in organisms that undergo considerable alternative splicing, albeit, in different extents. Figure 1.4 shows the prevalence of ambiguity in the transcriptome world (although not shown this can be extended to genomic world too). The y-axis of fig. 1.4 is

a measure of sequence uniqueness. Here sequence uniqueness is measured by the fraction of sequence unique k-mers vs the total number of k-mers in the sequence. Such a measure naturally would lie between 0 and 1. In the figure, the transcripts are stratified into 10 different bins according to the uniqueness score. It is evident from the plot that, although number of sequence unique transcripts are large, the number of very ambiguous transcripts are also comparable. The uniqueness scores are plotted for both Human and Drosophila (fruit fly).

1.3.1 Encoding the redundancy

A major source of such transcript level ambiguity as noted earlier are directly related to shared exonic sequences. This inherent nature of sequence sharing in the underlying sequence also permeates most RNA-seq datasets. As a natural outcome we frequenly observe equally good mapping of reads to the underlying reference at multiple loci. This phenomenon is depicted in the right hand side of fig. 1.4. In this toy example paired end reads r_3 and r_4 both map equally well to transcripts t_1 and t_2. In other words with respect to an experiment that contains only this read, transcripts t_1 and t_2 are indistinguishable. Turro et al. [31], Nicolae et al. [32] , Srivastava et al. [26] and Srivastava et al. [33] termed these data-driven natural groups as *equivalence classes*, referring to the equivalency of the transcripts in the light of the mapping information. We can extend the notion of equivalence classes further to represent them as graphs, here the edge between the two transcripts are inferred from their co-occurrence in the same equivalence class. Such graph are

proven to be useful for clustering de-novo contigs and and produce cluster level expression [33].

1.3.2 Leveraging over the redundancy

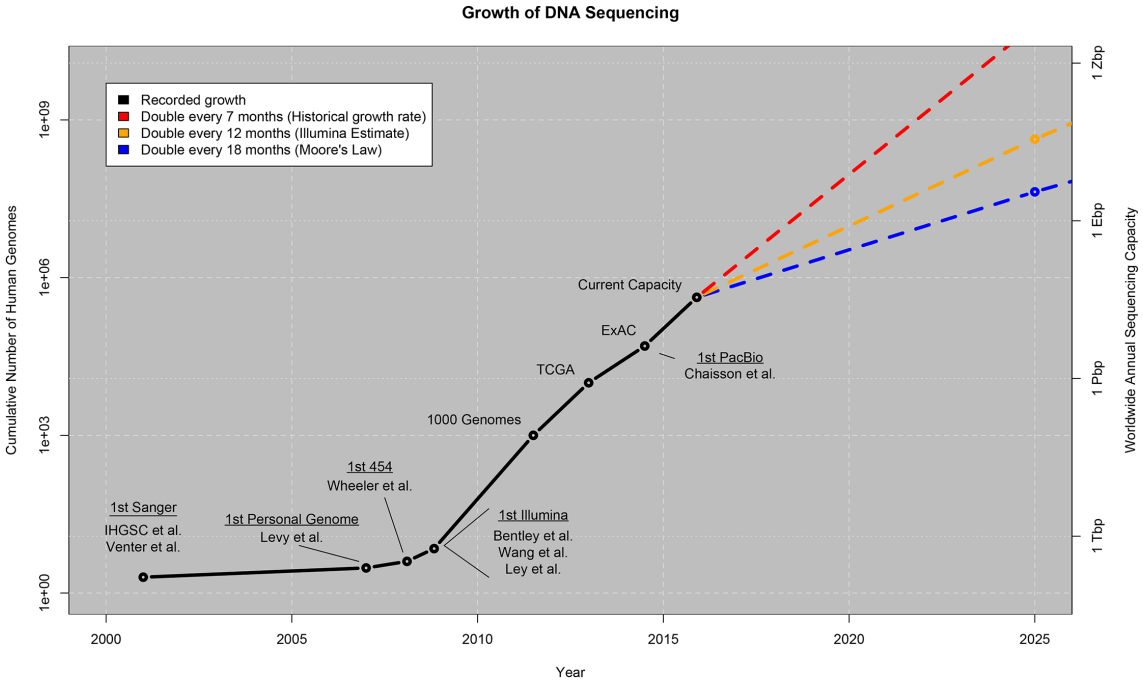


Figure 1.5: The exponential decay of sequencing cost and simultaneous growth of public database over the years. (Collected from Stephens et al. [2])

The advent of high throughput sequencing technology coupled with the exponential decrease in sequencing cost led to the generation of petabytes of data on servers worldwide. Figure 1.5 from [2], is plotted from the publicly available data uploaded to Sequence Read Archive (SRA) over the years. While the decreasing cost over the years has beaten the Moore's law, so does the volume of data. Computational tools that can efficiently process tools are also aided with superior compression algorithms, to match the storage requirement. Apart from size, often succinct representation [34] can yield very similar results with a much smaller memory footprint.

It has been known for some time [35] that pre-processing of sequencing data can "boost" the performance of off-the-shelf compression tools such as `gzip` or `plzip`.

As noted in section 1.3.1 such preprocessing can exploit the redundancy of raw sequence data readily available from equivalence classes. Sarkar and Patro [36] presented a mapping-based pre-processing and encoding of the data that considerably improves the effectiveness of downstream compression tools. In one hand, such encoding can exploit the redundancy of highly repeated sequences from the read. On the other hand, I demonstrate that quasi-mapping, a recently-introduced proxy for traditional alignment [26], enables selective compression of read sequences with respect to the reference sequence.

Chapter 3 provides a brief overview of two different applications where the redundancy in RNA-seq data is utilized for read compression and accelrating the alignment procedure.

1.4 RNA-seq Quantification Problem

After alignment the next crucial step of general RNA-seq pipelines is measuring the gene-level or transcript-level abundances. Most of the downstream analysis depends on these expression values. Just like mapping the quantification can be performed on either transcripts and genes (if not both). Both of these levels has own set of trade-offs. The problem of quantification is closely tied to that of mapping, since the output from mapping tools are directly fed to the quantification tools.

Gene-level quantifiers are relatively more straightforward than their transcript-

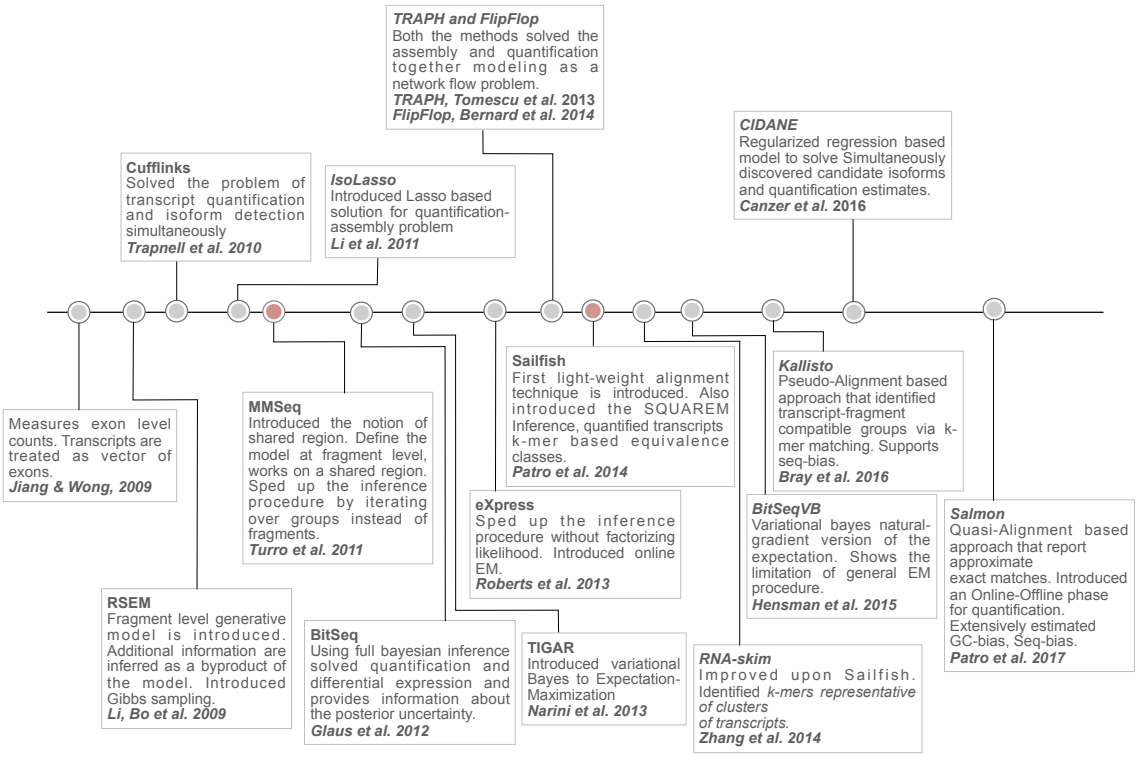


Figure 1.6: Evolution of transcript level quantification tools/methods. The upper half shows the joint problem solving involving both isoform discovery and quantification. The lower half mostly considered the transcript quantification tools.

level counter parts. The fragments mapped to genomic regions are counted to measure the gene-level expression. Often reads that do not map to exonic regions are discarded. Although the direct genomic mapping is efficient and easy, it suffers from many fundamental short-comings, such as, the multi-mapping reads are often thrown away even when that consists of a non-trivial percentage of the total dataset. Since the concept of gene is merely an abstraction it's often unclear which mappings are to be considered as valid, for example, a read that maps to union of all exons might not refer to a valid transcript in reality, or often mask a more viable alternative that maps to a known transcript. For these reasons, transcriptome level quantification is favored over measuring direct gene level expression. On the other hand transcript-level expression when required, can be transformed to gene-level expression values with different summarization techniques.

Transcript-level counts offer a finer resolution while answering biological questions, and do so in a more comprehensive and interpretable way. The main computational challenge of transcriptional expression analysis is building a model that accounts for the multi-mapping reads. This refers to the situation that I discussed in detail in section 1.3. Figure 1.6 shows a chronological order of development of transcript-level quantification tools. Note that fig. 1.6 shows the tools that I am familiar with, and I admit there are many more in the literature since it's a popular area of research. I offered a mathematically rigorous description of the main computational models for RNA-seq quantification in chapter 2.

1.4.1 How can we assess quantification tools ? A case for simulation.

Quantification is difficult problem, the absence of a high confidence ground truth dataset makes the problem of assessing a quantification tool even harder. Unlike other purely computational domains, the domain of quantification lacks easy to create gold standard benchmark dataset. To mitigate the problem, tools are often correlated against each other as a form of testing [37]. Although that provides a sanity check but since the complexity of real experiment can not be tweaked easily (without re-running the biological experiment, which can be expensive), it is not ideal to iterate over while designing the software. The other more comprehensive solution is to design a simulator that can emulate a real experiment, and provides a parametric setting to change the "adversarialness" of the dataset to test the robustness of the tool. [38] is a widely popular bulk-RNA-seq simulator that addressed many aspects of real dataset such as sampling reads from a learned distribution. Although one caveat is the time consuming initiation step that requires alignment of the real sequences, additionally no in-build settings is provided for generating differentially expressed transcripts. [39] is another popular simulator that provides a more biologically inspired model of generating RNA-seq read simulators, although it also does not provide an easy solution for generating dataset that encodes a control-treatment experiment. Relatively recently published tool `polyester` from Frazee et al. [40] mitigated these problems by providing a custom designed differentially expressed dataset.

1.4.2 The missing simulator in the single cell domain

The above tools are widely used for generating simulated dataset in bulk-RNA-seq. While bulk-RNA-seq is extremely helpful for measuring the gene expression for a population of cells, it fails to capture the cell level heterogeneity at the cellular level. Recently developed single cell RNA-seq technologies have mitigated many of the caveats of bulk RNA-seq and are proven to be a game changer in detecting cluster of rare cell types, cell fates, tracking the process of cell differentiation and measuring many other dynamic developmental changes in a tissue and organism.

The same is not available for single cell RNA-seq dataset. More specifically, there exist methods for pseudo-time series analysis, differential cell usage ,cell-type detection RNA-velocity in single cells etc. Most analysis pipelines validate their results using known marker genes (which are not widely available for all types of analysis) and by using simulated data from gene-count-level simulators. Typically, the impact of using different read-alignment or UMI deduplication methods has not been widely explored. Assessments based on simulation tend to start at the level of assuming a simulated count matrix, ignoring the effect that different approaches for resolving UMI counts from the raw read data may produce. Motivated by these problems I introduced *minnow*, a principled sequence-level droplet-based single-cell RNA-seq (dscRNA-seq) experiment simulation framework. *Minnow* accounts for important sequence-level characteristics of experimental scRNA-seq datasets and models effects such as PCR amplification, CB (cellular barcodes) and UMI (Unique Molecule Identifiers) selection, and sequence fragmentation and sequencing. It also

closely matches the gene-level ambiguity characteristics that are observed in real scRNA-seq experiments. Using *minnow*, I explore the performance of some common processing pipelines to produce gene-by-cell count matrices from droplet-bases scRNA-seq data, demonstrate the effect that realistic levels of gene-level sequence ambiguity can have on accurate quantification, and show a typical use-case of *minnow* in assessing the output generated by different quantification pipelines on the simulated experiment.

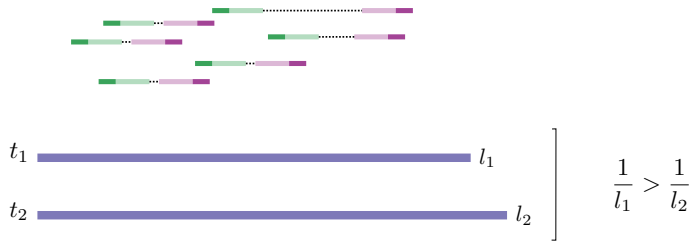


Figure 1.7: An example of highly uncertain assignment

1.4.3 Exploring the hidden uncertainty in the domain of RNA-seq quantification

A close inspection of fig. 1.3 reveals that a scenario such as fig. 1.7 described in is highly plausible. Widely used probabilistic graphical model based quantification algorithms rely on maximum likelihood based solution would greatly prefer one transcript over the other. This problem is magnified when there are a lot of sequence similar sequences. A practical scenario could be the presence of alleles as both the alleles of the same gene are almost identical to each other. Sometimes this can be a result of the underlying model. In the example t_1 is more favorable to the model in terms of assigning reads, by virtue of having smaller length. While

common sampling approaches (like bootstrapping) rely on the posterior distribution and provide uncertainty estimates that partially mitigate the problem, those estimates can also be biased towards the more probable candidates (i.e. the shorter transcript here). Incorrect abundance can further confound downstream differential expression analysis.

In chapter 5, I explored different scenarios where due to the shared substructure of the reference, RNA-seq reads map to multiple references at the same time. This phenomenon, known as multi-mapping affects the quantification process, adversarially, and could be solved by using posterior Gibbs samples, which directly measures the uncertainty in the underlying experiment. I proposed that grouping uncertain transcripts that share significant number of reads may reduce the uncertainty of the group itself. Note that measuring uncertainty is an important problem in itself, since there are different metrics that capture different properties of the distribution. On one hand simplistic measures such as variance is effective for capturing the crude spread of the distribution, but while comparing the variances of two different transcripts, it might be misleading when the actual values of the random variables are not comparable. A more stable metric could be coefficient of variation (CV), often defined as the ratio of standard deviation and mean of the distribution. The are a number of disadvantages with such metric, e.g. having a very low mean might inflate the CV, on the other hand if a distribution strictly follow Poisson distribution there is a chance of having CV of 1 regardless of the actual values. To navigate such cases we resorted to a quantity inferential relative variance (InfRV). Chapter 5 presents an in-depth analysis for using this metric to

measure uncertainty.

Chapter 2: Framework for RNA-seq quantification models

2.1 Introduction

Transcriptome quantification is a well studied problem as discussed in Chapter 1. Due to the nature of the complexity of the problem there are a number of challenges that are yet to be answered. In this chapter I will discuss some of the state-of-the-art computational techniques for RNA-seq quantification that are developed in the past few years. We claim that The RNA-seq quantification problem can serve as a benchmark use case to dissect and develop statistical models in general for modeling high throughput sequencing assays.

2.2 Background

Let the set of M transcripts and N fragments are denoted by $\mathcal{T} = \{t_1, t_2, \ldots, t_M\}$ and $\mathcal{F} = \{f_1, f_2, \ldots, f_N\}$ respectively. Typically M is in the order of hundreds of thousands (human transcriptome annotation currently has 200K transcripts), and N could be in the order of millions. Additionally, each transcript has different attributes, such as the actual alphabets of the sequence, length of the sequence etc. To formalize the notion, we define two different functions S and L that map a

particular symbol of a transcript or a read fragment to a string (defined over the alphabet) and a natural number (non-zero positive integer to signify length of the sequence). Following this convention, $s_j = S(f_j)$ denotes the sequence of a fragment and $s_k = S(t_k)$ denotes the sequence of a transcript t_k. Similarly l_j and l_k denote the length of the corresponding fragment f_j and transcript t_k respectively. Given this set up we set out to determine the relative proportion of each of the M transcripts. The relative proportion, often termed as *abundance*, denoted by $\boldsymbol{\rho} = \{\rho_1, \rho_2, \ldots, \rho_n\}$.

2.3 Models

2.3.1 A naïve model

A naive model for allocating reads to transcript is built on a set of assumptions, such as, all transcripts irrespective of length or sequence are equi-probable etc. These assumptions are in most of the cases inaccurate, however often provide solution that are not far from the truth.

To demonstrate one such model, let's assume there are two transcripts t_1 and t_2, and there are in total 150 short read fragments that are to be distributed them. Further let's assume, while mapping/aligning the reads we encounter 75 reads to be mapped to transcript t_1, 25 reads to be mapped to t_2 uniquely. The rest of the reads map ambiguously to both the transcripts. Given the scenario we wish to estimate the proportion of the two transcripts, that is we want to estimate ρ_1 and ρ_2. We recognise that if there is no prior on the transcripts then the initial values of ρ_1

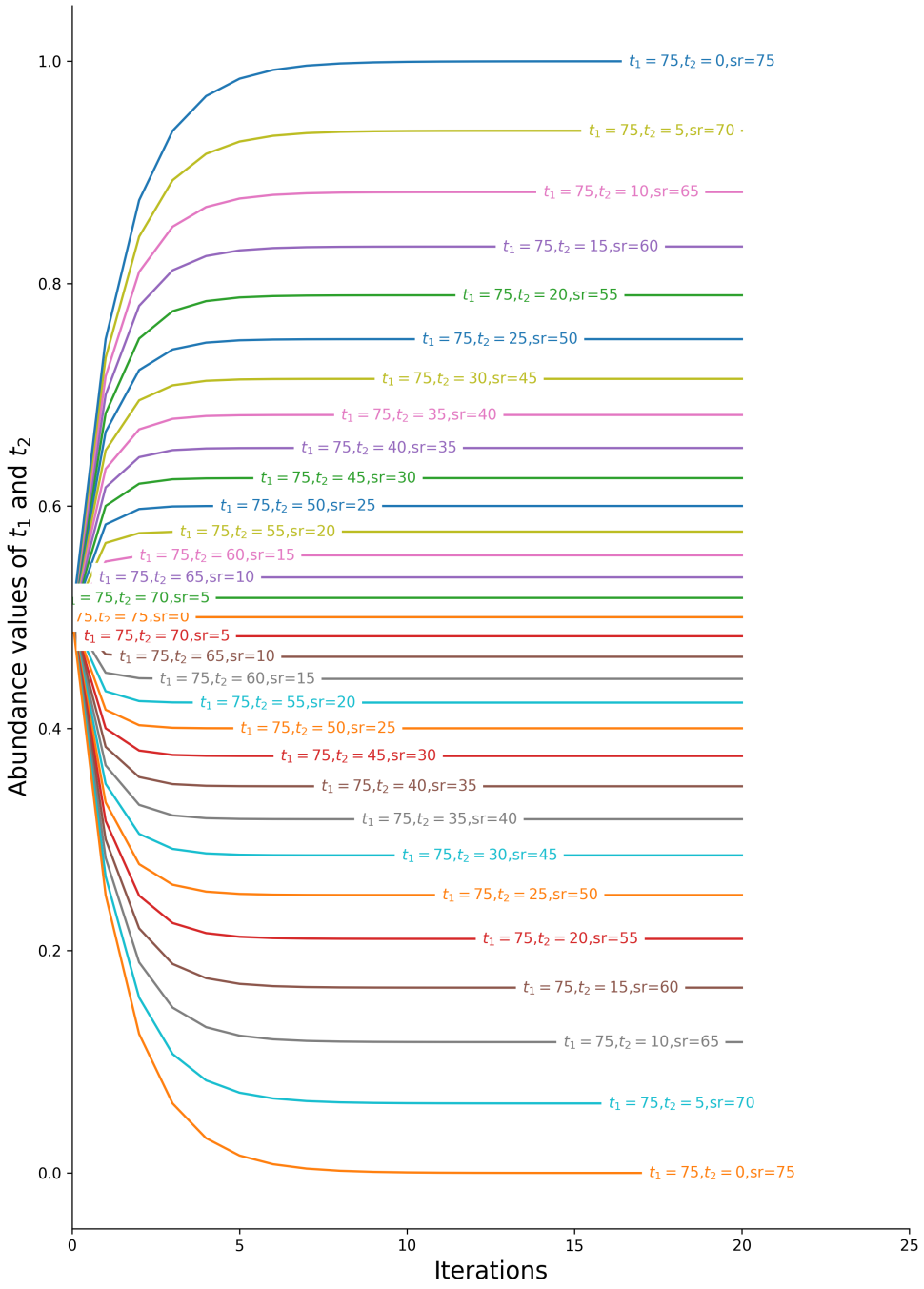


Figure 2.1: Relationship between the abundance values ρ_1 and ρ_2 for two different transcripts and the shared reads under the naive model.

and ρ_2 should be 0.5 (given the equi-probable assumption). Figure 2.1 shows the change of ρ_1 and ρ_2 when we vary the number of shared reads. While both ρ_1 and ρ_2 initiated from 0.5 diverged in different directions depending on the number of shared reads. Each curve in fig. 2.1, labelled with the unique reads for t_1, t_2, and the number of shared reads termed as "sr".

The iterative distribution of the reads is governed by a simple rule, starting with $\rho_1^0 = 0.5$, each step updates

$$\rho_1^{k+1} = \frac{\text{unique reads assigned to } t_1 + \text{shared reads} \times \rho_1^k}{\text{total reads}} \tag{2.1}$$

We observe that when there is evidence of some unique read assignment (eg. as 5 unique reads are assigned to t_2) the model (in essence the update rule) reaches a reasonable conclusion, that is assigning transcript ρ_2 an non-zero fraction. Unfortunately, this might not be the case in all situations. Consider the scenario depicted in fig. 2.3.

Just because there is one read that is uniquely mapped to the longer transcript, all the reads would be allocated to that one. While it is quite possible that there are substantial number of reads that come from the shorter transcripts (fig. 2.3). The above described scenarios clearly suggest that, a sophisticated model is necessary to capture the quirks of RNA-seq quantification. Before delving deeper into a more involved statistical model, we should recognize that this simple read allocation rule for iteratively soft-assigning reads is actually a proxy for an expectation-maximization

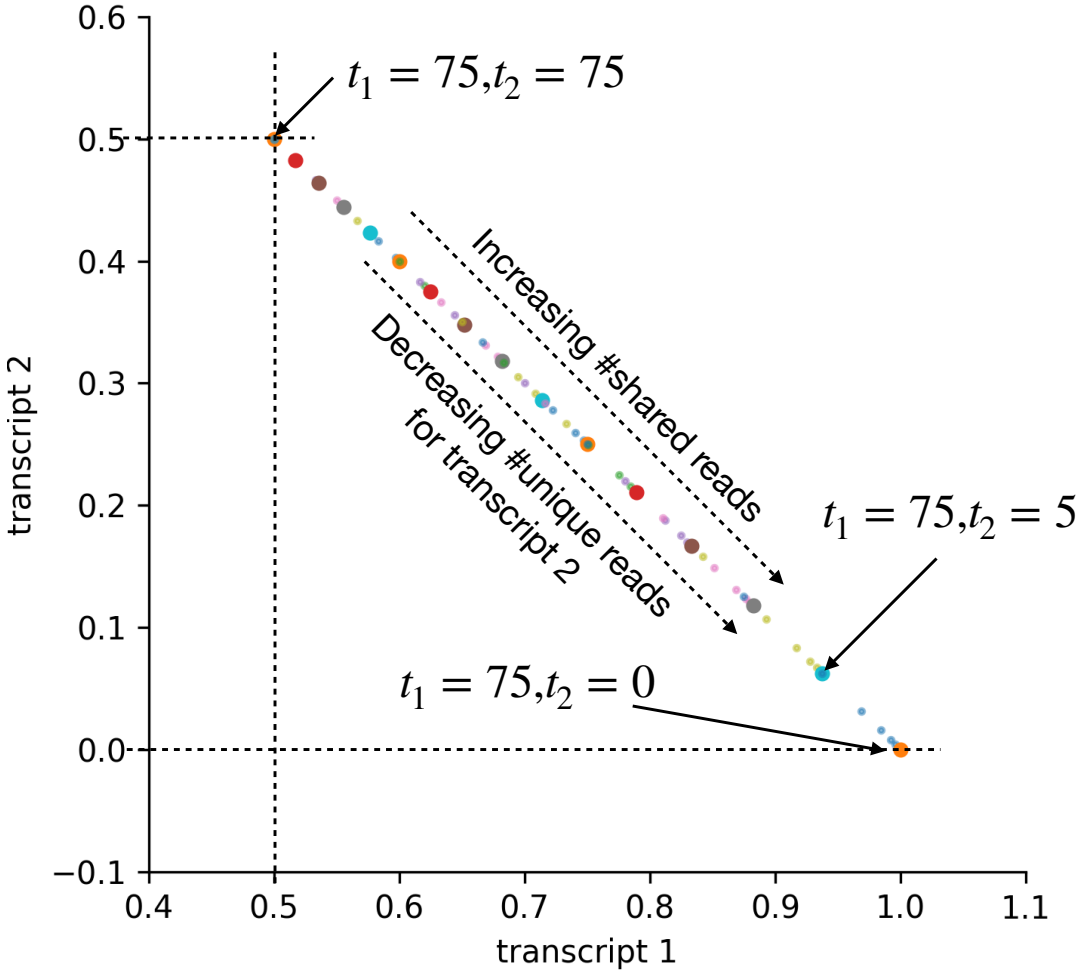


Figure 2.2: The final abundance values ρ_1 and ρ_2 when the number of shared reads decrease gradually under the naive model.

(EM) update rule [41]. In fact as we will see, the EM algorithm also converges to a very similar update rule.

2.3.2 Brief primer on expectation-maximization based algorithms

Most of the RNA-seq quantification models are based on an underlying probabilistic graphical model, and the optimization problem is formed to estimate the

Figure 2.3: One transcript is a complete subset of the other. There is one unique read and there are three shared reads between two transcripts

abundance of the transcriptome. We will review the generic rules related to expectation maximization before discussing each model separately. Given a set of observed variables X and a set of parameters θ, we aim to model the data and estimate the parameter θ. In general the problem is solved by optimizing the value of $p(X|\theta)$. In most of the cases there are other variables in the model that are not observed but dictates how the model works, let's denote the unobserved variable as Z. Given the above characterization we observe the following equation,

$$\mathcal{L}(\theta) = \log(p(X|\theta)) \tag{2.2}$$

$$= \log(\int p(X, Z|\theta)dz) \tag{2.3}$$

The integral is with Z respect to eq. (2.3). We often introduce a probability distribution over Z, $q(Z)$ (with properties $q(Z) \geq 0, \int q(Z) = 1$) and arrive at the following formulation,

$$\mathcal{L}(\theta) = \log(\int p(X, Z|\theta)dz) \tag{2.4}$$

$$= \log(\int q(Z)\frac{p(X, Z|\theta)}{q(Z)}dz) \tag{2.5}$$

$$= \log(\mathbb{E}_{q(Z)}[\frac{p(X, Z|\theta)}{q(Z)}]) \tag{2.6}$$

$$\geq \mathbb{E}_{q(Z)}[\log(\frac{p(X, Z|\theta)}{q(Z)})] \qquad \text{(Jensen's inequality wrt log)} \tag{2.7}$$

$$= \mathcal{F}(q, \theta) \tag{2.8}$$

Equation (2.8) provides a lower bound for the original likelihood. Let's assess the difference between this lower bound and the actual likelihood $\mathcal{L}(\theta)$. To achieve that we recognize the following decomposition,

$$p(X, Z|\theta) = \frac{p(X, Z, \theta)}{p(\theta)} \tag{2.9}$$

$$= \frac{p(X, Z, \theta)}{p(X, \theta)}\frac{p(X, \theta)}{p(\theta)} \tag{2.10}$$

$$= p(Z|X, \theta)p(X|\theta) \tag{2.11}$$

Decomposing the $\mathcal{F}$ further,

$$\mathcal{F}(q,\theta) = \int q(Z)\log\{\frac{p(X,Z|\theta)}{q(Z)}\}dz \tag{2.12}$$

$$= \int q(Z)\log\{\frac{p(Z|X,\theta)p(X|\theta)}{q(Z)}\}dz \tag{2.13}$$

$$= \int q(Z)\log P(X|\theta)dz + \int q(Z)\log\frac{p(Z|X,\theta)}{q(Z)}dz \tag{2.14}$$

$$= \log P(X|\theta)\int q(Z)dz - KL(q(Z)||p(Z|X,\theta)) \tag{2.15}$$

$$= \mathcal{L}(\theta) - KL(q(Z)||p(Z|X,\theta)) \tag{2.16}$$

It is evident from eq. (2.16) that the gap between the likelihood and the approximation function $\mathcal{F}(q,\theta)$ is the $KL(q(Z)||p(Z|X,\theta))$. From the properties of Kullback-Leibler (KL) when it becomes 0, this lower bound is equal to the likelihood. Clearly the best value for $q(Z)$ to approximate $\mathcal{L}(\theta)$, is $p(Z|X,\theta)$. Plugging this value back to $\mathcal{F}(q,\theta)$ we get the following equation,

$$\mathcal{F}(q,\theta) = \int p(Z|X,\theta)\log\{\frac{p(X,Z|\theta)}{p(Z|X,\theta)}\}dz$$

Note that we don't know the optimal value of $\mathcal{L}(\theta)$, since we don't know the exact parameter θ^*. However we can start with an initial value θ^0.

Instead of calculating $\mathcal{L}(\theta^0)$, we can calculate $\mathcal{F}(q,\theta^0)$.

E step: In the step we evaluate the value of function q that maximizes $\mathcal{F}(q,\theta^0)$. We already know that such value of q (reducing the KL divergence to zero) will be $p(Z|X,\theta^0)$.

M step: Given a q (which is at this stage $p(Z|X,\theta^0)$), we want to find a new

θ^1 that maximizes $\mathcal{F}(q, \theta)$. Plugging the value of $p(Z|X, \theta^0)$

$$\theta^1 = \underset{\theta}{\operatorname{argmin}} \int p(Z|X, \theta^0) \log\{\frac{p(X, Z|\theta)}{p(Z|X, \theta^0)}\} dz \tag{2.17}$$

$$= \underset{\theta}{\operatorname{argmin}} \left(\int p(Z|X, \theta^0) \log p(X, Z|\theta) dz - \int p(Z|X, \theta^0) \log p(Z|X, \theta^0) dz \right) \tag{2.18}$$

$$= \underset{\theta}{\operatorname{argmin}} \left(\int p(Z|X, \theta^0) \log p(X, Z|\theta) dz \right) + \text{constant wrt } \theta \tag{2.19}$$

$$= \underset{\theta}{\operatorname{argmin}}\, \mathcal{Q}(\theta, \theta^0) \tag{2.20}$$

In short, M step finds θ^1 that maximizes the function. This can be achieved by widely used optimization methods (such as taking differential with respect to θ). However if the there are additional constraints involved then Lagrange multiplier or such methods has to be incorporated.

2.3.3 The RSEM model

While it is not explicitly mentioned, the above described algorithm 2.3.1, is indeed an optimization algorithm. At each step we maximize the likelihood of the observed reads given the actual transcripts. Li et al. [42] provides a comprehensive probabilistic graphical model to formalize the notion.

2.3.3.1 Problem formulation

The parameter to be estimated is the transcript proportion (abundance) denoted as θ, the observed variables are the reads denoted as R, the intermediate

hidden variable (can encapsulate more than one variable) is denoted as a random variable Z. This Z is crucial to the understanding of the problem of read alignment. Depending on the model specifications $z \in Z$ could encode many possible alignment configuration [43]. For example $z_{n,i,j} = 1$ represents, read $r_n \in R$ aligns to transcript i at position j. One could add orientation and other possible parameters to make the model more complete. As reads are discrete quantities, the treatment presented in section 2.3.2 has to be modified from continuous domain to the discrete domain (changing the integration to summation).

The sampling of a fragment from a transcript is modeled as a generative process. Following the notation of RSEM, using three random variables, a read r_n is sampled from a transcript G_n with a start position S_n. A configuration of alignment can be encoded as a triplet $(r_n, G_n, S_n) = (r_n, i, j)$ which signifies that the read is sampled from transcript i and from position j. Following the similar principle as before this configuration can be represented by an indicator function $z_{n,i,j}$. $z_{n,i,j} = 1$ suggest the occurrence of such a configuration. To put formally

$$p(r_n, z_{n,i,j} = 1|\theta) = p(z_n|\theta) \times p(r_n|z_{n,i,j} = 1, \theta) \tag{2.21}$$

$$= \theta_i(\text{a transcript i is chosen}) \tag{2.22}$$

$$\times \frac{1}{l_i}(\text{a position j within transcript i is chosen}) \tag{2.23}$$

$$\times a_{n,i,j}(\text{confidence of such an alignment}) \tag{2.24}$$

Given the definition of $z_{n,i,j}$, $\sum_{i,j} z_{n,i,j} = 1$

Adhering to the conditional indicator variable we can deduce,

$$p(r_n, z_{n,i,j}|\theta) = \frac{\theta_i}{l_i} a_{n,i,j} \qquad \text{if } z_{n,i,j} = 1 \tag{2.25}$$

$$= 0 \qquad \text{otherwise} \tag{2.26}$$

$$p(r, z|\theta) = \prod_n \sum_{i,j} z_{n,i,j} \frac{\theta_i}{l_i} a_{n,i,j} \tag{2.27}$$

$$= \prod_{n,i,j} \left[\frac{\theta_i}{l_i} a_{n,i,j} \right]^{z_{n,i,j}} \tag{2.28}$$

The change of eq. (2.27) to eq. (2.28) is possible because of the boolean nature and summing to 1. Taking logarithm of both the sides

$$\log p(r, z|\theta) = \sum_{n,i,j} z_{n,i,j} \log \left(\frac{\theta_i}{l_i} a_{n,i,j} \right) \tag{2.29}$$

To perform an EM algorithm we need to maximize the function $\mathcal{Q}$ from eq. (2.20).

$$\mathcal{Q}(\theta, \theta^0) = \sum_Z p(Z|X, \theta^0) \log p(X, Z|\theta) \tag{2.30}$$

$$= \sum_{n,i,j} p(z_{n,i,j} = 1|r_n, \theta_0) \log \left(\frac{\theta_i}{l_i} a_{n,i,j} \right) \tag{2.31}$$

$p(z_{n,i,j} = 1|r_n, \theta)$ describes the probability of sampling a read r_n from a transcript i from a position j. To measure that probability, we use bayes theorem,

$$p(z_{n,i,j} = 1|r_n, \theta) = \frac{p(r_n, z_{n,i,j} = 1|\theta)}{p(r_n|\theta)} = \frac{p(r_n, z_{n,i,j} = 1|\theta)}{\sum_{i',j'} p(r_n, z_{n,i',j'} = 1|\theta)} = \frac{\frac{\theta_i}{l_i} a_{n,i,j}}{\sum\limits_{i',j'} \frac{\theta_{i'}}{l_{i'}} a_{n,i',j'}} \tag{2.32}$$

The above derivation gives us the E step. In the M step we need to take a derivative of $\mathcal{Q}$.

$$\frac{\partial \mathcal{Q}(\theta, \theta^0)}{\partial \theta_t} = \frac{\partial}{\partial \theta_t} \left[\sum_{n,i,j} p(z_{n,i,j} = 1|r_n, \theta^0) \log \left(\frac{\theta_i}{l_i} a_{n,i,j} \right) \right] \tag{2.33}$$

$$= \frac{\partial}{\partial \theta_t} \left[\sum_{n,i=t,j} p(z_{n,i,j} = 1|r_n, \theta^0) \log \theta_t + \text{constant with respect to } \theta_t \right] \tag{2.34}$$

$$= \frac{\partial}{\partial \theta_t} \left[\log \theta_t \sum_{n,j} p(z_{n,t,j} = 1|r_n, \theta^0) + \text{constant with respect to } \theta_t \right] \tag{2.35}$$

$$= \frac{1}{\theta_t} \sum_{n,j} p(z_{n,t,j} = 1|r_n, \theta^0) \tag{2.36}$$

Since the set of transcripts are predefined, $\sum\limits_i \theta_i = 1$. Using this constraint we could write the updated optimization as $\mathcal{Q}'(\theta, \lambda) = \mathcal{Q}(\theta, \theta^0) + \lambda(\sum_i \theta_i - 1)$, where λ is the Lagrange multiplier. We arrive at the following point,

$$\lambda = \frac{1}{\theta_t} \sum_{n,j} p(z_{n,t,j} = 1 | r_n, \theta^0) \tag{2.37}$$

$$\theta_t = \frac{1}{\lambda} \sum_{n,j} p(z_{n,t,j} = 1 | r_n, \theta^0) \tag{2.38}$$

By summing up θ_t

$$\frac{1}{\lambda} \sum_{i} \sum_{n,t,j} p(z_{n,t,j} = 1 | r_n, \theta^0) = 1 \tag{2.39}$$

$$\frac{1}{\lambda} \sum_{n,i,j} p(z_{n,i,j} = 1 | r_n, \theta^0) = 1 \tag{2.40}$$

$$\frac{1}{\lambda} \sum_{n} \sum_{i,j} p(z_{n,i,j} = 1 | r_n, \theta^0) = 1 \tag{2.41}$$

$$\frac{1}{\lambda} \sum_{n} 1 = 1, \text{ given } \sum_{i,j} p(z_{n,i,j} = 1 | r_n, \theta^0) = 1 \tag{2.42}$$

$$\frac{N}{\lambda} = 1 \tag{2.43}$$

$$\lambda = N \tag{2.44}$$

Plugging the value of λ back in eq. (2.38),

$$\theta_t^1 = \frac{1}{N} \sum_{n,j} p(z_{n,t,j} = 1 | r_n, \theta^0) \tag{2.45}$$

To summarize the above analysis, the EM algorithm can be formalized as follows,

while θ *does not converge* **do**
  for $t \in \mathcal{T}$ **do**
    $p(z_{n,t,j} = 1 | r_n, \theta_t^k) \leftarrow \frac{\frac{\theta_t^i}{l_i} a_{n,t,j}}{\sum_{i',j'} \frac{\theta_{i'}}{l_{i'}} a_{n,i',j'}}$;
    $\theta_t^{k+1} \leftarrow \frac{1}{N} \sum_{n,j} p(z_{n,t,j} = 1 | r_n, \theta^k)$;
  end
end

Algorithm 1: EM algorithm for updating abundances of the transcripts

2.3.4 mmseq model

The major contribution of the model proposed by Turro et al. [31] was representing segments of transcripts as individual units of expression. The core of the model includes a matrix which has the similar intuition of that of z matrix described previously. One key difference is the segment-to-transcript matrix M described in Turro et al. [31] consists of boolean values M_{it}, that corresponds to region i and transcript t. Such a construction relies on the fact that in RNA-seq experiment, reads are not mapped back to the entire transcript, but they indeed get mapped to regions of transcripts. Thereby, it could be the case that multiple reads are mapped to the same region of a single transcript. M_{it} denotes the membership of a region to a transcript. Biologically a region might be interpreted as an exon or a group of exons within a transcript. Given such a framework the number of reads generated from region i of transcript t is given by a poisson distribution: $X_{it} \sim Pois(bs_i M_{it} \mu_t)$. Here s_i is the effective length and μ_t is the expression of transcript t. Although X_{it} is not observed, total number of reads from a region i, $k_i = \sum_t X_{it}$ is observed. Given that sum of poisson random variables yields a poisson random variable [1], we

[1] https://nptel.ac.in/content/storage2/courses/108106083/lecture15_Sums_of_RVs.pdf

observe the following property,

$$k_i \sim Pois\left(bs_i \sum_t M_{it}\mu_t\right)$$

We also observe that the effective length of a transcript can be characterized by $l_t = \sum_i s_i M_{it}$. Turro et al. [31] did not directly use s_i, rather use l_i in the actual EM equatuations, as we would observe soon. For a particular region, the reads from all transcripts within that region can be characterized by the vector $\{X_{i1}, X_{i2}, \ldots, X_{it}\}$. The same equation while looked through the lens of probabilistic graphical model is actually the probability of the hidden variable, where the sum of the variables is observed. Therefore we are interested in the quantity $P(Z|R, \theta)$, or $P(\{X_{i1}, X_{i2}, \ldots, X_{it}\}|\{\mu_1, \ldots, \mu_t\}, k_t)$.

Theorem 2.3.1. *Let $X_{i1}, \ldots, X_{it}$ are poisson random variables with means $\lambda_1, \ldots, \lambda_t$, such that $\sum_j X_{i,j} = k_i$, then the joint distribution $\{X_{i1}, \ldots, X_{it}\}$ follows a multinomial distribution with parameters k_i and $\{\frac{\lambda_l}{\sum_l \lambda_l}\}$*

Proof. Well established result[2]. □

Using section 2.3.4, the probability of generated counts follows the multinomial distribution $Mult(k_i, \frac{M_{it}\mu_t}{\sum_{t'} M_{it'}\mu_{it'}})$. The expectation from the multinomial $\mathbb{E}(X_{it}|k_i, \mu_t^{(p)}) = k_i \frac{M_{it}\mu_t^{(p)}}{\sum_{t'} M_{it'}\mu_{t'}^{(p)}}$

As this model is more simplistic, we can directly derive the EM steps for this distribution, which comes out to be,

[2]https://ecommons.cornell.edu/bitstream/handle/1813/32480/BU-39-M.pdf?sequence=1

$$\mu_t^{(p+1)} \leftarrow \frac{\sum_i X_{it}^p}{bl_t} \tag{2.46}$$

$$= \frac{\sum_i k_i \frac{M_{it}\mu_t^{(p)}}{\sum_{t'} M_{it'}\mu_{t'}^{(p)}}}{bl_t} \tag{2.47}$$

$$= \frac{\mu_t^{(p)}}{bl_t} \sum_i \frac{k_i M_{it}}{\sum_{t'} M_{it'}\mu_{t'}^{(p)}} \tag{2.48}$$

Note that if each region is one transcript, eq. (2.48) is very similar to the update equation of RSEM.

A more recent method *salmon* Patro et al. [28] employs a model that is similar to the concept of a region (or segment). *salmon* does not explicitly identify regions within a reference rather collapses similar read-to-transcript assignments to equivalence classes. The notion of equivalence classes are discussed in depth in section 2.4. Note that, Patro et al. [28] introduced many novel concepts such as incorporating the GC bias model, using light-weight mapping in order to accelerate the quantification process, etc.

Apart from RSEM and mmseq model, there are many other interesting models that introduce new directions in RNA-seq quantification problem, such as Glaus et al. [44] introduced a purely bayesian model, and proposed a solution using variational bayes expectation maximization (VBEM).

2.3.5 Quantifying the uncertainty

The model described in Section 2.3.4 can be used to design a sampling scheme by turning the model into a generative one. The general method of designing a sampling scheme within this kind of model is to derive all the conditionals. The corresponding generative model for mmseq is as follows:

- Mean expression of a transcript μ_t is sampled using a Gamma distribution with hyperparameters α and β. (Turro et al. [31] have chosen 1.2 and 0.0001 as hyperparameters).

$$\mu_t \sim Gam(\alpha, \beta)$$

- Given μ_t and a region i with an effective length s_i, the probability of sampling X_{it} reads is distributed as a Poisson random variable.

$$X_{it} \sim Pois(bs_i M_{it} \mu_t)$$

As X_{it} is not directly observed, Turro et al. [31] considered the joint distribution of all reads (k_i) come from segment i (over all the transcripts that share i). Using the previous derivation, the conditional probability for sampling $\mathbf{X}_i = \{X_{i1}, \ldots, X_{it}\}$

can be given by

$$P(\mathbf{X}_i|\{\mu_1,\ldots,\mu_t\},k_i) = Mult\left(k_i, \frac{M_{i1}}{\sum_{t'} M_{it'}\mu'_t},\ldots,\frac{M_{it}}{\sum_{t'} M_{it'}\mu'_t}\right) \tag{2.49}$$

$$P(\mu_t|\{X_{1t},\ldots,X_{mt}\}) = \frac{P(\{X_{1t},\ldots,X_{mt}\}|\mu_t)P(\mu_t)}{P(\{X_{1t},\ldots,X_{mt}\})} \tag{2.50}$$

$$= \frac{P(\{X_{1t},\ldots,X_{mt}\}|\mu_t) \times Gam(\mu_t;\alpha,\beta)}{P(\{X_{1t},\ldots,X_{mt}\})} \tag{2.51}$$

$$= \frac{\prod_j Pois(X_{jt};bs_jM_{jt}\mu_t) \times Gam(\mu_t;\alpha,\beta)}{\int_0^\infty \prod_j Pois(X_{jt};bs_jM_{jt}\mu_t) \times Gam(\mu_t;\alpha,\beta)d\mu_t} \tag{2.52}$$

$$= \frac{\frac{\beta^\alpha}{\Gamma(\alpha)}\frac{e^{-b\mu_t\sum_j s_jM_{jt}-\mu_t\beta}\prod_j(bs_jM_{jt})^{X_{jt}}\mu_t^{\sum_j X_{jt}+\alpha-1}}{\prod_j X_{jt}!}}{\frac{\beta^\alpha}{\Gamma(\alpha)}\frac{\prod_j(bs_jM_{jt})^{X_{jt}}}{\prod_j X_{jt}!}\int e^{-b\mu_t\sum_j s_jM_{jt}-\mu_t\beta}\mu_t^{\sum_j X_{jt}+\alpha-1}d\mu_t} \tag{2.53}$$

$$= \frac{\frac{\beta^\alpha}{\Gamma(\alpha)}\frac{e^{-\mu_t\beta'}\prod_j(bs_jM_{jt})^{X_{jt}}\mu_t^{\alpha'-1}}{\prod_j X_{jt}!}}{\frac{\beta^\alpha}{\Gamma(\alpha)}\frac{\prod_j(bs_jM_{jt})^{X_{jt}}}{\prod_j X_{jt}!}\int e^{-\mu_t\beta'}\mu_t^{\alpha'-1}d\mu_t} \tag{2.54}$$

$$\text{given}(\beta' = b\sum_j s_jM_{jt}+\beta, \alpha' = \sum_j X_{jt}+\alpha) \tag{2.55}$$

$$= \frac{e^{-\mu_t\beta'}\mu_t^{\alpha'-1}}{\frac{\Gamma(\alpha')}{\beta'^{\alpha'}}\int\frac{\beta'^{\alpha'}}{\Gamma(\alpha')}e^{-\mu_t\beta'}\mu_t^{\alpha'-1}d\mu_t} \tag{2.56}$$

$$= \frac{e^{-\mu_t\beta'}\mu_t^{\alpha'-1}}{\frac{\Gamma(\alpha')}{\beta'^{\alpha'}}\int Gam(\mu_t;\alpha',\beta')} \tag{2.57}$$

$$= \frac{\beta'^{\alpha'}}{\Gamma(\alpha')}e^{-\mu_t\beta'}\mu_t^{\alpha'-1} \tag{2.58}$$

$$= Gam(\mu_t;\alpha',\beta') \tag{2.59}$$

From the previous discussion, $\sum_j s_jX_{jt} = l_t$ and thereby one ideally does not need the effective length for each segment.

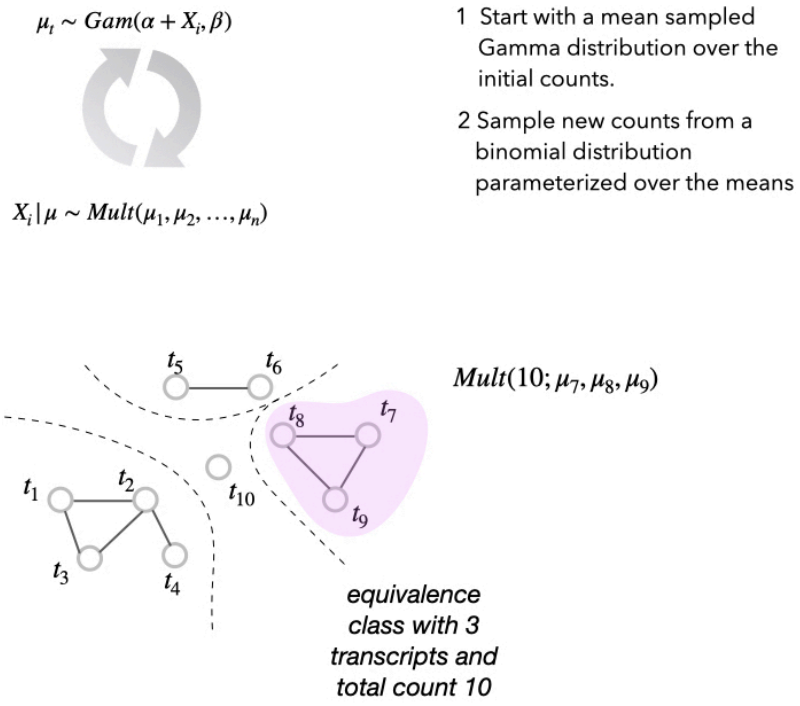


Figure 2.4: Gibbs sampling in mapping ambiguity graph

2.4 Equivalence classes in *salmon*

The concept of fragment equivalence classes[3] is intuitively similar to the idea of segments proposed by Turro et al. [31]. We define an equivalence relation over fragments, based on the set of transcripts to which they map. The set of fragments related under this definition constitutes a fragment equivalence class. Let $\mathcal{M}(f_i)$ be the set of transcripts to which fragment f_i maps, and let $\mathcal{M}(f_j)$ be the set of transcripts to which fragment f_j maps. We say that $f_i \sim f_j$ if and only if $\mathcal{M}(f_i) = \mathcal{M}(f_j)$. Consequently, a fragment equivalence class is a set of fragments, such that for every pair f_i and f_j in the class, $f_i \sim f_j$. An equivalence class can be uniquely labeled based on the set of transcripts to which the fragments contained in

[3]Relevant portion of this description is taken from Srivastava et al. [33], where I am a co-first author.

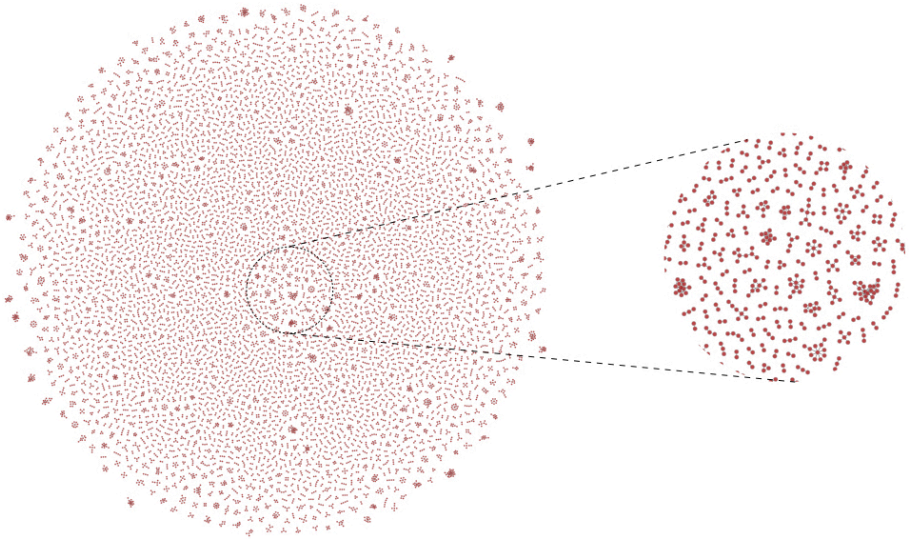


Figure 2.5: mapping ambiguity graph on a real experiment

this class map. We define the label of equivalence class $[f_i] = \{f_j \in \mathcal{F} | f_j \sim f_i\}$ as $\texttt{lab}([f_i])$. It is important to remember that, though the label consists of transcript names, the equivalence relation itself is defined over sequenced fragments and not transcripts. Finally, in addition to the label, we denote the count of each equivalence class C_i by $c(C_i)$; this is simply the number of equivalent fragments in C_i.

2.4.1 Graph deduced from equivalence classes

The fragment equivalence classes, as described above, are already computed internally by *salmon* [45]. We have modified *salmon* to write these equivalence classes to a file once quantification is complete (this behavior is enabled with the `--dumpEq` flag). This yields, for each *sample*, a collection of equivalence classes, along with their associated labels and counts. To construct the complete mapping ambiguity graph of the *experiment* we need to aggregate these equivalence classes. We generate a single collection $\mathcal{C}$ of equivalence classes by merging the classes $\mathcal{C}_1, \ldots, \mathcal{C}_M$. Here,

$\mathcal{C}$ contains the union of equivalence classes from $\mathcal{C}_1, \ldots, \mathcal{C}_M$, and classes that appear in more than once $\mathcal{C}_1, \ldots, \mathcal{C}_M$ simply have their respective read count summed. The time and space requirements for this aggregation algorithm is linear in the size of input.

For a given experiment, the collection $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ of equivalence classes induces a weighted, undirected, mapping ambiguity graph $G = (V, E)$. Here, $V = T$, where T is the set of transcripts in the original transcriptome — and $E = \{\{t_i, t_j\} \mid \exists\, C_\ell \in \mathcal{C}$ where $\{t_i, t_j\} \subseteq \texttt{lab}\,(C_\ell)\}$ — that is, t_i and t_j are connected by an edge if and only if they both appear in the label of at least one equivalence class.

2.4.2 Gibbs sampling in the framework of equivalence classes

Since *salmon* produced equivalence classes are a segmented representation of the transcripts that share reads, it eases the implementation of Gibbs sampling. Figure 2.4 shows the actual Gibbs sampling procedure on a mapping ambiguity graph. Note that an efficient computational implementation can leverage the fact that the reads from the transcripts that do not share reads are independent. This is a key factor that I referred in chapter 5. Figure 2.5 shows the mapping ambiguity graph corresponding to a real experiment. Note that when we zoom into the graph, even though the number of vertices in the graph can be enormous, the graph is actually very parse. Moreover, the graph in fig. 2.5 is a collection of smaller connected components.

Chapter 3: Applications of shared sequences in RNA-seq data

3.1 Introduction

The repetition of sequences in eukaryotic transcriptome is a direct derivative of many biological factors such as, shared exons, presence of paralogous or homologous genes etc. As shown in fig. 1.3 the short reads that generates from these transcripts also share sequences. This redundant sequences within reads can be utilized in various applications from sequence compression to fast alignment. In this chapter I will mention two different methods that I worked on during my doctoral study: Compression of RNA-seq sequences using `Quark`[1] and an accelerated light-weight alignment technique termed as *selective alignment*[2].

3.2 Introduction

Compression of high-throughput sequencing reads becomes crucial with the lowering cost of sequencing technology. The rapid technological development enables the generation of petabytes of data on servers worldwide. Apart from size, often succinct representation [34] can yield very similar results with a much smaller

[1]The relevant sections for `Quark` is taken from Sarkar and Patro [36]
[2]The relevant sections for describing selective alignment are taken from Sarkar et al. [46]

memory footprint. It has been known for some time [35] that pre-processing of sequencing data can "boost" the performance of off-the-shelf compression tools such as `gzip` or `plzip`. In this work, I present a mapping-based pre-processing and encoding of the data that considerably improves the effectiveness of downstream compression tools.

In one hand, such encoding can exploit the redundancy of highly repeated sequences from the read. On the other hand, we demonstrate that quasi-mapping, a recently-introduced proxy for traditional alignment [26], enables selective compression of read sequences with respect to the reference sequence. `Quark` is a compression method specifically designed for high throughput RNA-seq reads. On a conceptual level it introduces the idea of semi-reference-based compression, where reference sequence is used at the encoding end, but is not required for decompression. This allows `Quark` to obtain markedly better compression rates than completely reference-free tools while also eliminating the need for the encoder and decoder to share the same exact reference sequence, which also mitigates the potentially brittle dependence of a reference-based encoder on a specific reference sequence. Specifically, using quasi-mapping [26], `Quark` locates regions of interest in the reference that are specific to the particular RNA-seq experiment being compressed, and stores only these regions for use during decoding. `Quark` is focused on sequence compression, and hence, does not currently provide a mechanism for storing the header and quality information associated with each read. Although there are very efficient approaches to these problems [47], [48], [49] that could easily be coupled with `Quark`. Apart from reducing the total size of `fastq` files, the other motivation is that many tools

(e.g., state-of-the-art quantification tools such as `Sailfish` [50], `Salmon` [45] and `kallisto` [51]) do not make use of this information from the `fastq` files. In fact, the link between `Quark` and transcript quantification methods goes even deeper, as `Quark`'s notion of islands (discussed in section 3.3.1) naturally extends and refines the notion of fragment equivalence classes first introduced in `mmseq` [31], and subsequently adopted by recent lightweight quantification approaches such as `Sailfish`, `Salmon`, and `kallisto`.

`Quark` is the first reference-asymmetric compression methodology of which we are aware (i.e., in terms of only requiring the reference for encoding). Further, it develops certain key connections between the redundant representation of sequence information, in terms of read compression, and the use of related ideas in the efficient likelihood factorization that has been integral to the development of fast quantification methodologies. Our analysis also provides some insights into the typical coverage / usage of unique sequence in specific RNA-seq experiments (section 3.3.1). The idea of semi-reference-based compression appears very effective at improving sequence compression rates, but may not be limited only to RNA-seq data. Rather, we believe that the ideas we present here may also be useful for the compression of genomic data in e.g., non-model organisms, where a reliable reference genome may be unavailable, out-of-date, or highly incomplete.

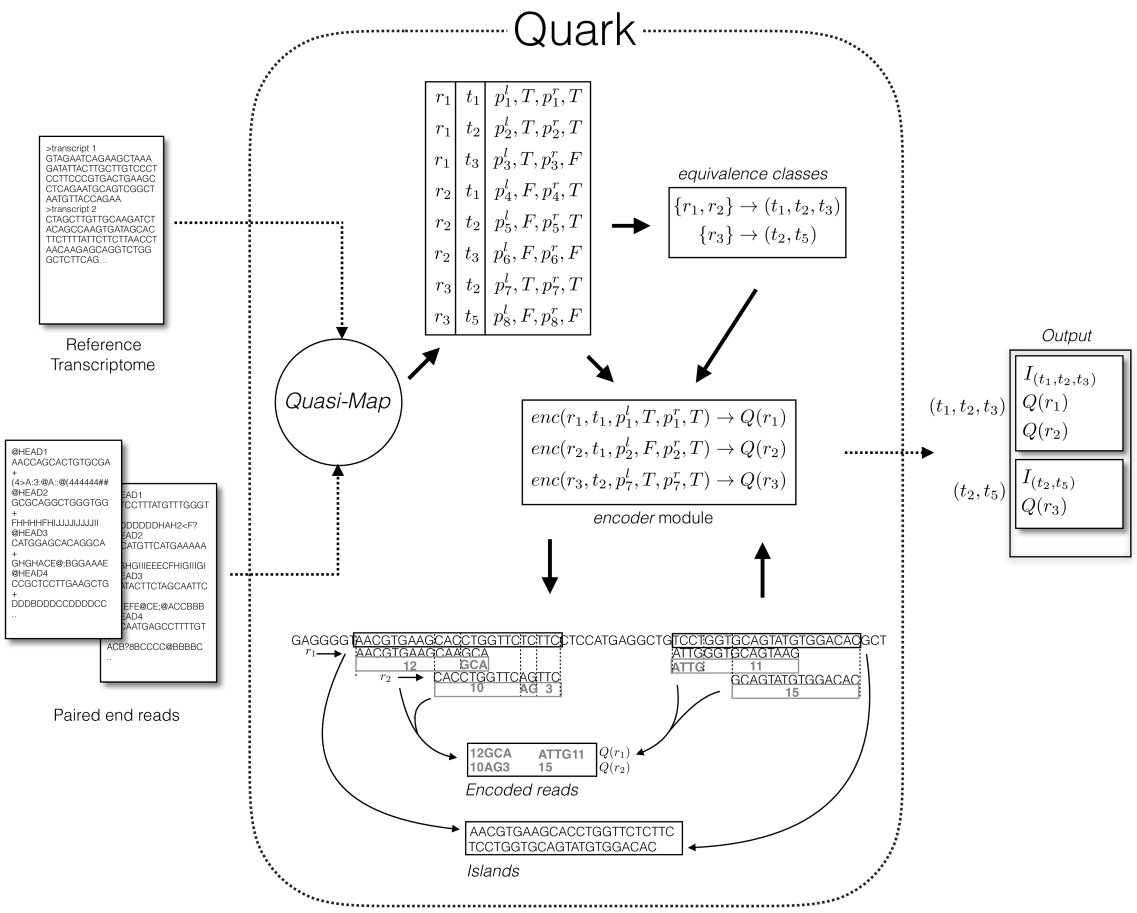


Figure 3.1: Quark uses the core component of *RapMap* which is quasi-mapping. It is used to produce a set of tuples for a paired-end read r_i, where each tuple can be represented as $(p_i^l, F_i^l, p_i^r, F_i^r)$. The table containing the tuples for each read can be summarized to a set of equivalence classes as discussed in section 3.3. The encoding function Q is explained above with two paired end reads r_1 and r_2. For left end of r_1, there are 12 matches followed by unmatched characters. For the right end of r_1, first 4 characters differ from the reference, followed by 11 exact matches, the left and right end together can be encoded as $Q(r_1) = \{12\text{GCA,ATTG11}\}$. The relevant intervals are subsequently stored as islands.

3.3 Quark Method

`Quark` implements a semi-reference-based compression algorithm, where the reference is required for compression but not for decompression. To remove this dependency at the decoder, `Quark` encodes and stores only parts of underlying reference which are required for decompression. Thus, the output of `Quark` is self-contained in the sense that the raw reads can be recovered from the `Quark` output without the aid of any additional file. To be precise, given the reference sequence and the reads, `Quark` generates three files, *read.quark*, *offsets.quark* and *islands.quark*. Before describing the core algorithm of `Quark` in detail, we briefly describe the quasi-mapping concept, and the algorithm to efficiently compute quasi-mappings introduced in Srivastava et al. [26], which is an integral part of the `Quark` algorithm. We note that, throughout the discussion, we have used the term "mate" and "end" synonymously to refer to the paired-ends of a sequenced fragment. Generally, we use the terms left and right "end"/"read" to refer to the paired end reads prior to quasi-mapping, whereas we use "mate" to refer to the mapping for the opposite end of a paired-end read (i.e., the notation used in the `SAM` format). We do not deal here with mate-pair sequencing.

Given some reference (e.g., a transcriptome), quasi-mappings identify each read with some set (possibly empty) of target sequences (e.g., transcripts), positions and orientations with which the read shares a consistent collection of right-maximal exact matches. The quasi-mapping algorithm described in Srivastava et al. [26] constructs a suffix array-based index over the reference sequence. The mapping

process starts with a matched k-mer that is shared between a set of transcripts and the read. If such a match exists, the algorithm tries to extend the match further by searching the interval of the suffix array for a maximum mappable prefix. The match is used to determine the *next informative position* in the reference sequences, and the same mapping process continues from that point within the query. These exact matched sequences play an important role in achieving superior compression rate of `Quark`.

On the basis of the result provided by the quasi-mapping algorithm described above, we can divide the input reads (assumed, for simplicity, to be paired-end reads) into three categories:*Mapped reads:* If both reads of the pair are mapped, this is an ideal situation where we can encode both ends of the read efficiently since each of the reads shares some sequence with the reference; *Orphan reads:* For reads in this category, we can not map both ends of the pair to the same target. In `Quark`, we encode the unmapped end of the read by writing its (encoded) sequence directly;*Unmapped reads:* There is no mapping at all for the read, as determined by the algorithm described above, and so the read is instead encoded using a reference-free approach. In `Quark`, un-mappable reads are encoded using the reference-free compression tool. `Mince` [52].

`Quark` follows a hierarchical approach for compression, where the mapped reads are distributed into equivalence classes according to the transcripts to which they map, and then sorted, within each class, by their starting position. Reads within the same equivalence class are very likely to share overlapping reference sequence, and hence, to be similar to each other. The encoding scheme itself is straight

forward. Given the position and reference sequence, `Quark` does a linear search on the reference sequence to find the matching sequence between the read and the reference at the specified position. Though it is guaranteed to yield a match of at least k nucleotides if given the k-mer criterion of the mapping algorithm, typically the collection of matches covers a large fraction of the read data.

`Quark`'s read encoding. The encoding phase of `Quark` starts with the output produced by quasi-mapping. Given a read r_i mapped to the reference transcriptome, `Quark` produces a tuple $\tau_i^k = (t_k, p_i^l, F_i^l, p_i^r, F_i^r)$ (See fig. 3.1), where t_k is the transcript sequence where the read maps, p_i^l is the position where the left end maps and F_i^l is a flag which is false if the the read has to be reverse complemented to map and true otherwise. Likewise, p_i^r and F_i^r represents the corresponding position and flag for the right end of the read. When a read maps to multiple transcripts, a tuple is returned for each transcript to which the read maps. It should be noted that there are other flags that are maintained internally by quasi-mapping to keep track of orphan reads and other mapping information not currently used by `Quark`. Once the tuples $\tau_1, \tau_2, \ldots$ for a read are obtained, they are used to place each read into an equivalence class based on the transcripts to which they map, and all reads mapping to precisely the same set of transcripts will be placed into the same equivalence class. This notion of equivalence classes has been used in the transcript quantification literature for some time [31], and is described in more detail in Srivastava et al. [26]. As shown in fig. 3.1, given the mapping information for reads r_1, r_2 and r_3, we can extract the corresponding transcripts as follows, $r_1 \rightarrow (t_1, t_2, t_3)$, $r_2 \rightarrow (t_1, t_2, t_3)$

Figure 3.2: In case of reads mapped partly to the reference, beginning before the transcript start point, we append 0 in the beginning and encode the relative position where mapping starts in the read.

and $r_3 \rightarrow (t_2, t_5)$. Intuitively, we expect that r_1 and r_2 are more likely to share overlapping reference sequence with each other than with r_3. In addition to the transcript *labels*, `Quark` also associates a collection of nucleotide sequences (i.e. reference sequence to which the read maps) with each equivalence class.

The core encoding process operates on each equivalence classes individually. Given t_j, a reference sequence for transcript j, the quasi-mapping information $\tau_i = (t_j, p_i^l, F_i^l, p_i^r, F_i^r)$ for a read, and an encoding function Q, `Quark` proceeds as follows. For the left end `Quark` starts a simultaneous linear scan for matches from $t_j[p_i^l]$ and $r_i[0]$, i.e., the start of the read sequence. As discussed previously, it is guaranteed that if quasi-mapping yields a mapping for this read, then the search will also find at least a match of length k. In `Quark`, both ends of a paired end read are compressed simultaneously, and an analogous encoding procedure is used for the right end of the read. We can formalize Q as $Q : \Sigma \rightarrow \Sigma \cup \{0 - 9\}$, where $\Sigma = \{A, T, G, C, N\}$. Integers values are required to represent the number of matched characters.

Efficient four-bit encoding scheme. To make the `Quark` encoding more efficient, we use a four bit encoding scheme to represent characters of each encoded read. The

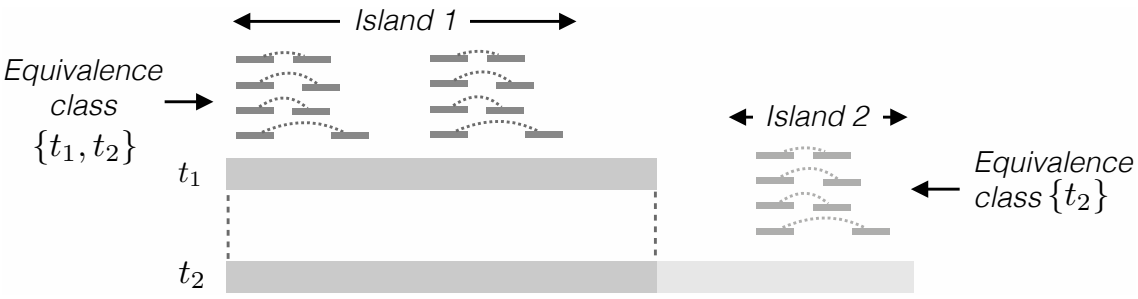


Figure 3.3: t_1 and t_2 are two transcripts that share a sequence. Reads labeled in dark grey are mapped into the shared region where as the reads labelled with light grey are only mapped exclusively to transcript t_2, leading to formation of two equivalence classes.

alphabet size used for encoding is 15, hence we need at least four bits to represent that. In case of overhang, where a part of read falls of the edge of transcript (as shown in fig. 3.2), we put a 0 in front of the number to signify what follows is not the number of matched characters, rather it is the position from where the quasi-mapping starts. The four bit codes are used for all 15 aforementioned characters. Additionally as an ending delimiter after each read we put a four bit code of `0000`. The island id and corresponding position are stored in a separate binary `offset` file. The additional file that contains the sequences of the islands are kept in plain text format and later lzipped (using aforementioned `plzip` program). To summarize, at the end of the a run of `Quark`, it produces an island file, offset file(s) and encoded read file(s) for a single end or paired end read data.

3.3.1 Island Construction

For the purposes of compression, `Quark` makes use of islands of reference sequence that overlap the mapped reads. We define an island as a contiguous substring of some reference sequence that is completely covered by at least one read

(i.e., some read overlaps each nucleotide in this substring). For each read r_i and its corresponding tuple $\tau_i = (t_k, p_i^l, F_i^l, p_i^r, F_i^r)$, we construct an additional list $\mathcal{I}_i$ of intervals containing $\{(p_i^l, p_i^l + len(r_i)), (p_i^r, p_i^r - len(r_i))\}$, where $len(r_i)$ represents the length of the read. Some care must be taken to properly handle boundary conditions, which can result in situations where a read *overhangs* the beginning or end of a reference sequence. Repeating this step for each read within an equivalence class `Quark` constructs a set $\mathcal{I} = \bigcup_i \mathcal{I}_i$.

Given that intervals on the reference sequence might share some nucleotides (i.e., overlap), `Quark` merges the intervals by taking the union of the nucleotides they contain, to form maximal disjoint islands. Construction of islands from intervals is straightforward. `Quark` sorts the intervals with respect to their start positions, and a linear scan through intervals suffices to find the overlaps and merge the islands into disjoint subsets.

The use of islands aids the compression abilities of `Quark`, and additionally makes the resulting compression file self-contained, eliminating the need to assume the decoder has access to the same reference. `Quark` can identify shared regions between transcripts by the use of quasi-mapping. This further enables it to exploit redundancy and store only one island for each cluster of reads that share some nucleotide (see fig. 3.3).

The island generation process removes the redundancy of nucleotides from transcripts that share some region. Figure 3.3 illustrates such a situation, where a large portion of nucleotides from transcript t_2 will be omitted (i.e. not included in any island). Here the reads in equivalence class $\{t_1, t_2\}$ are completely accounted

for by the island formed by the sequence from t_1, so that an island corresponding to the prefix of t_2 is redundant and need not be generated. However the reads in equivalence class $\{t_2\}$ mapped to a disjoint transcriptomic region that is not present in t_1. The final set of islands (island 1 and island 2 in fig. 3.3) will thus contain only one representative for the redundant sequence shared by t_1 and t_2, so that the majority of t_2 won't be used in island creation. We further note here that this process of discovering and removing redundancy in the stored sequence from the underlying transcriptome is completely free of reference annotations, so that `Quark` works equally well when compressing the reads with respect to a *de novo* transcriptome assembly.

To study the effectiveness of constructing islands, we considered dataset from `SRR635193`. After mapping to the Gencode reference transcriptome for human (version 19), we observe that out of $95,309$ transcripts, only $49,589$ transcripts are used by `Quark`. To see the contribution of individual transcripts in the island generation process, we have plotted a histogram of the fraction of each transcript's total nucleotides present in the set of islands. Figure 3.4 shows that there are many transcripts where only a small fraction of nucleotides participate in islands. We note that this need not imply that a transcript has low abundance, since repetitive sequence is encoded only once in the island generation process, and the selection of which transcript is used to support the island for a given equivalence class is essentially arbitrary.

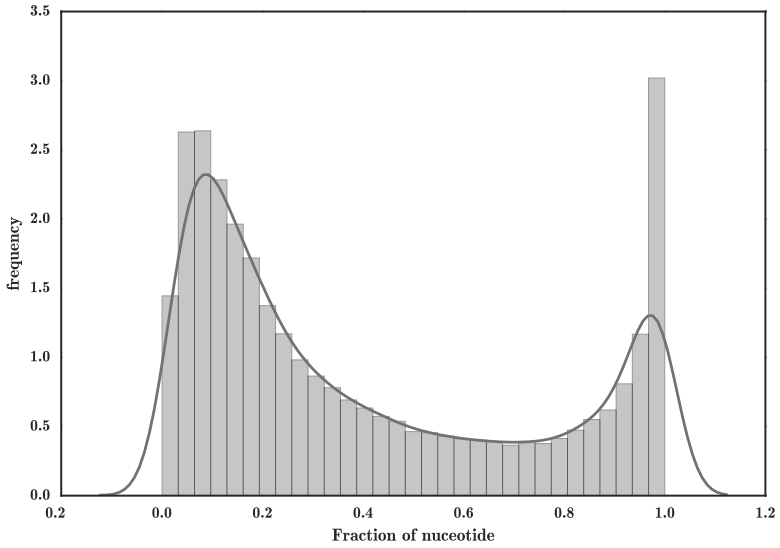


Figure 3.4: A histogram of fraction of nucleotides that are present in a typical set of islands. The representative fraction for a given transcript is simply generated from the ratio of number of nucleotides present in the set of islands and the length of that transcript.

	Organism	Library type	Raw-sequence	gzip	Quark	LEON	SCALCE	SCALCE*	Mince
SRR689233	Mouse	PE	2 986 245 990	898 788 316	**99 028 458**	247 687 471	561 022 336	233 134 542	176 947 631
SRR445718	Human	SE	3 327 310 165	982 390 260	**146 495 063**	328 857 885	581 694 785	252 450 288	162 969 503
SRR490961	Human	SE	4 961 894 468	1 463 105 279	**159 074 135**	466 517 518	745 095 275	299 983 580	174 583 153
SRR635193	Human	PE	2 999 246 910	893 786 599	**149 197 655**	361 795 379	446 853 627	293 842 354	240 274 889
SRR037452	Human	SE	421 663 860	129 649 883	54 691 651	90 774 360	111 469 817	66 630 751	**54 302 557**
SRR1294122	Human	SE	3 384 221 798	1 215 281 439	**180 413 521**	437 077 070	662 252 794	298 757 892	204 850 132
SRR1265495	Human	PE	3 432 947 700	227 236 786	**158 635 313**	305 101 315	709 315 147	320 744 415	242 176 319
SRR1265496	Human	PE	2 985 375 780	210 149 518	**152 612 910**	280 749 535	636 872 285	299 286 153	224 755 144
ERR310208	Human	PE	6 398 133 126	1 874 591 238	**247 839 373**	718 089 231	1 032 016 279	409 551 882	344 072 201

Figure 3.5: Size of the read files after compression along with raw sequence size (bytes) PE: Paired End, SE: Single End. Column `SCALCE*` records the `SCALCE` compressed file sizes when discarding the quality file altogether. This makes the file no longer de-compressible, but provides a reasonable approximation (slight underestimate) of the space used by `SCALCE` to encode just the sequence.

3.3.2 Post-processing

In addition to yielding a reduced representation of the reads, the work done by `Quark` organizes the encoded reads in a format and order that is amenable to further compression by traditional mechanisms (e.g. using programs such as `gzip`, `bzip2`, `lzip`). Given the encoding size benefits of `lzip` described by Patro and Kingsford [52], we further process the `Quark` encodings, using `lzip` to compress the *read.quark*, *offsets.quark* and *islands.quark*. files, which contain the encoded reads, their offsets and the sequences of the islands, respectively. As the size of the unmapped reads can not be improved by taking advantage of quasi-mapping, we use the pre-existing *de novo* compression tool `Mince` [52] to compress these reads.

I have shown the superiority of `Quark` in fig. 3.5 for compressing both single end or paired end reads.

3.4 Application in alignment

In the paper [30] [3], we present a novel concept, selective-alignment, that extends the quasi-mapping algorithm to compute and store alignment information where necessary. The reads for alignment are chosen based on certain criteria calculated using mapping. This strikes a balance between speed and accuracy; not compromising the superior speed of fast mapping algorithms, while also addressing some of the challenges mentioned above. Specifically, the motivation for selective-

[3]This is a joint work with Mohsen Zakeri. Specific, my contributions were, generating and implementing the concept of k-safe-LCP, validating the method on different datasets to show the superiority of the method

alignment is to enhance both the sensitivity and specificity of fast mapping algorithms by reducing or eliminating cases where spurious exact matches mask true mapping locations as well as cases where small exact matches support otherwise poor alignments. We build our selective-alignment algorithm atop the framework of *RapMap* [26], which uses an index that combines a fixed prefix length hash table and an uncompressed suffix array [53]. We introduce a coverage-based consensus scheme to identify critical read candidates for which alignment is necessary.

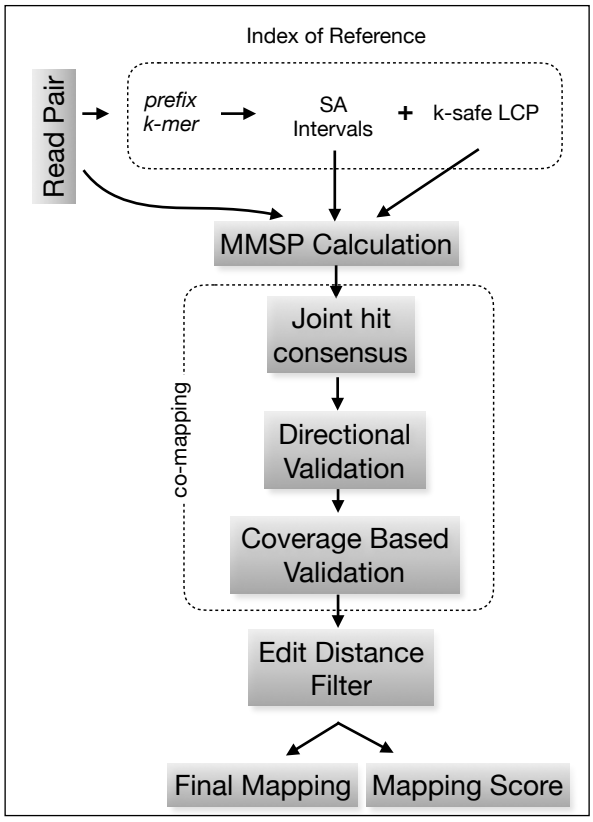
Furthermore, we explore the challenging cases where the heuristics employed by fast mapping algorithms may fail to locate the correct locations for a read, while the traditional aligners do not. We do this by making a number of modifications to the underlying mapping algorithm to increase its sensitivity. We also introduce multiple filters and scoring schemes designed to eliminate spurious mappings (i.e., situations where the best mapping is unlikely to represent the true origin of the read). In this work, we focus on the effect of selective-alignment on transcript quantification estimates, and we leave a thorough evaluation of the alignment qualities themselves as future work. In particular, the evaluation of alignment qualities is considerably complicated due to prevalent multi-mapping in the transcriptome.

3.5 Selective Alignment

The process of selective-alignment builds upon the basic data structures of Srivastava et al. [26], but there are a number of important algorithmic distinctions. Specifically, compared to the algorithm of RapMap, selective-alignment introduces

the k-safe longest common prefix (k-safe-LCP), replaces maximum mappable prefixes (MMP) with maximum mappable safe prefixes (MMSP), increases mapping sensitivity by adopting a different consensus rule over hits, makes use of co-mapping to filter and prioritize potential mapping loci, introduces a new mechanism for selecting a mapping position for a read when multiple candidates exist on the same transcript, and, finally, introduces a fast edit distance filter (with alignment subproblem caching) to remove spurious mappings and provide quality scores for mappings that pass the filter. A block diagram of different steps used in the selective alignment pipeline is shown in fig. 3.6.

Below, we recapitulate the basic data structures and concepts that will be required to explain the selective alignment algorithm. To start with, the index built on the transcriptome in selective-alignment is a combination of a suffix array and a hash table constructed from unique k-mers (substrings of length k) and suffix array intervals. The suffix array of a sequence, T— denoted $SA(T)$—is an array of starting positions of all suffixes from T in the original sequence. The values in the array are sorted lexicographically by the suffixes they represent. Therefore, all suffixes starting with the same prefix are located in adjacent positions of the suffix array. Formally, given a suffix array, $SA(T) = \Lambda$, constructed from the transcriptome sequence, T, we construct a hash table, h, that maps each k-mer, κ, to a suffix array interval, $\mathtt{I}(\kappa) = [b, e)$, if and only if all the suffixes within interval $[b, e)$ contain the k-mer κ as a prefix. We define $\Lambda[i]$, for every $0 \leq i \leq |\Lambda|$, to be the suffix $T[SA[i]]$ (i.e., the suffix of T starting from position $SA[i]$). In the selective-alignment index, in addition to suffix array intervals, we store two extra pieces of information for each



MMSP: Maximal Mappable Safe Prefix
SA: Suffix Array
LCP: Longest Common Prefix

Figure 3.6: The steps for selective-alignment are described in the form of block diagram. The block wrapped in dotted box is the index building phase, which used for mapping. Mapping undergoes numerous steps including co-mapping and filtering.

interval; the longest common prefix (LCP) and the k-safe-LCP corresponding to the interval. The longest common prefix (LCP) of any pair of suffixes in the suffix array is simply the length of the prefix that these suffixes share. Though the LCPs for the suffixes in the suffix array can be pre-computed, we instead compute them on demand using a linear scan. These methods are detailed below. As an alternate to the suffix array and the LCP array, one could make use of other data structures which also encode this information. For example, the recently-introduced method, Fleximer [54] makes use of the suffix tree for selecting informative sig-mers [55]

from the transcriptome, and matching reads against them.

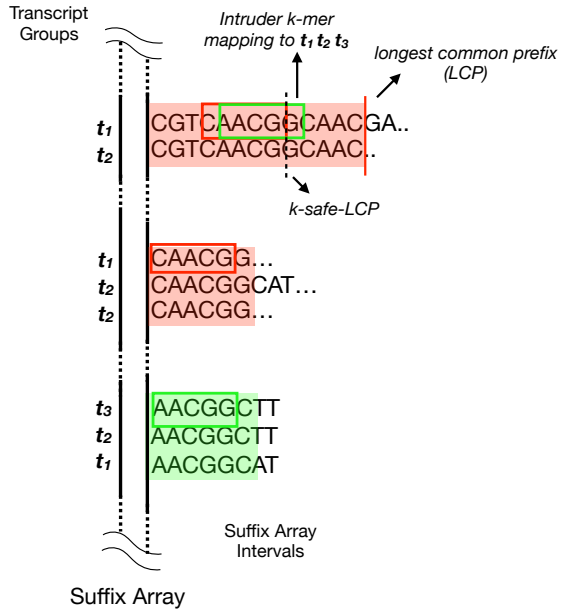


Figure 3.7: Calculation of k-safe-LCP from the suffix array data structure. The transcripts present in each suffix array interval determine the relevant transcript sets, and which k-mers will be considered as intruders. To determine the k-safe-LCP of the suffix array interval starting with the k-mer $CGTCA$, we check all the k-mers sequentially. Some k-mers do not yield an interval with transcripts other than t_1 and t_2, e.g., $CAACG$. Detection of a k-mer ($AACGG$) (as intruder) that maps to suffix array interval labeled (t_1, t_2, t_3) determines the k-safe-LCP here.

3.5.1 Defining and computing k-safe-LCPs

Here, we formally define the concept of k-safe-LCPs (see figure 3.7). The determination of k-safe-LCPs starts by labeling each suffix array interval with the length of its corresponding longest common prefix and the associated transcript set it represents. Formally, $\texttt{LCP}(\Lambda[b], \Lambda[e-1])$ for an interval $[b, e)$ is the length of the common prefix of the suffixes $\Lambda[b]$ and $\Lambda[e-1]$. Given k-mer κ, where $\kappa \in \boldsymbol{\mathcal{K}}$ and $\boldsymbol{\mathcal{K}}$ is the set of all k-mers from the reference sequence T, and the related interval $\texttt{I}(\kappa) = [b, e)$, for all $p \in [b, e)$, we consider each transcript t such that the suffix

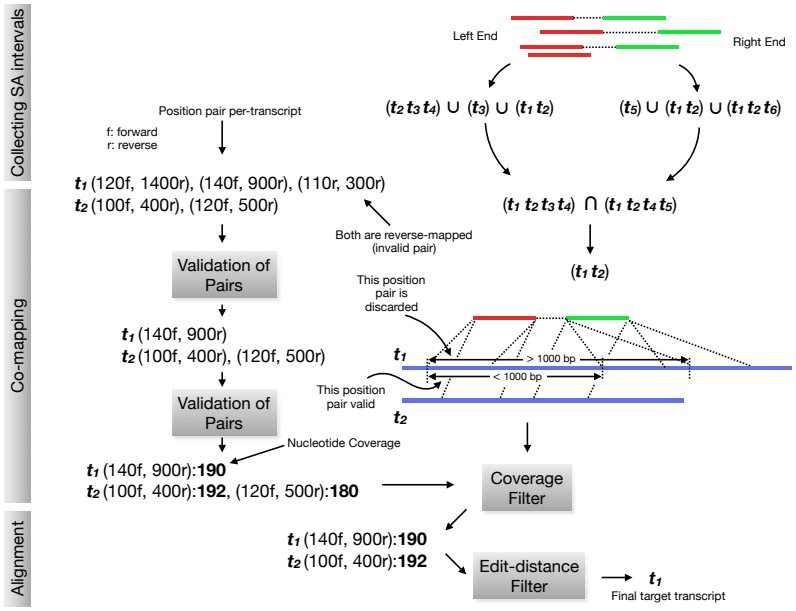


Figure 3.8: The three main steps of the selective-alignment process are demonstrated here. First, suffix array "hits" are collected. Then, in co-mapping, spurious mappings are removed by the orientation filter and then distance filter. At most a single locus per-transcript is selected based on the coverage filter. Finally, an edit-distance-based filter is used to select the valid target transcripts.

$\Lambda[p]$ starts in transcript t in the concatenated text. Then, for this interval, we can construct a set $\mathcal{C}^{\kappa} = \{t_i, t_j, \dots\}$, which denotes the set of distinct transcripts that appear in the suffix array interval, indicated by κ. We note that this notion discards duplicate appearances of the same transcript in this interval.

We compute the k-safe-LCP for an interval indicated by k-mer κ_i iteratively. The initial length for the k-safe-LCP of the interval is k, length of a k-mer. We check, sequentially, each of the k-mers in the longest common prefix of the interval. For each new k-mer, the k-safe-LCP is increased by one character. We terminate the k-safe-LCP extension if any of the following conditions is encountered: (1) we reach the last k-mer contained in the LCP of this interval, (2) we encounter a k-mer κ_j such that $\mathcal{C}^{\kappa_j} \not\subseteq \mathcal{C}^{\kappa_i}$ or (3) we encounter a k-mer κ_j such that the reverse

complement of κ_j appears elsewhere in the transcriptome. When we encounter case (2) or (3), we call the k-mer κ_j an *intruder*. That is, the k-mer will potentially alter our belief about the set of potential transcripts to which a sequence containing this k-mer maps (by strictly expanding this set), or the orientation with which it maps to the transcriptome. We denote the k-safe-LCP of a particular interval $\mathtt{I}(\kappa_i)$ as $\mathtt{k\text{-}safe\text{-}LCP}(\mathtt{I}(\kappa_i))$.

As shown in figure 3.7, the k-safe-LCP determination for the top suffix array interval starts with matching k-mers within the longest common prefix. The k-mer "CAACG" maps to a suffix array interval labeled with (t_1, t_2). The next k-mer "AACGG", on the other hand, maps to a suffix array interval (shaded in green) labeled with (t_1, t_2, t_3), thereby implying the k-safe-LCP, shown as a dotted line. For each k-mer in the hash table, we store the length of the LCP and k-safe-LCP, along with the corresponding suffix array interval.

3.5.2 Discovering relevant suffix array intervals

As shown in figure 3.8, the selective-alignment approach can be broken into three major steps: collecting suffix array intervals, co-mapping, and selecting the high quality mappings. Gathering the suffix array intervals for a query read closely follows the quasi-mapping approach. It involves iterating over the read from left to right and repeating two steps. First, hashing a k-mer from the read sequence and then discovering the corresponding suffix array intervals. The process of k-mer lookup is aided by the k-safe-LCP stored in the index (discussed in 3.5.1). The

inbuilt lexicographic ordering of the suffixes in the suffix array, and the computed k-safe-LCP values of intervals enable safely extending k-mers to longer matches without the possibility of masking potentially-informative substring matches. Given a matching k-mer, κ_r, from the read sequence r, we extend the match to find the longest substring of the read that matches within $\texttt{k-safe-LCP}(\texttt{I}(\kappa_r))$. The matched substring can be regarded as maximum mappable prefix (MMP) [19], that resides within the established k-safe-LCP. We call this a maximal mappable safe prefix (MMSP — eliding k where implied). For a k-mer, κ_r, and interval, $[b, e)$, we note that $\texttt{k-safe-LCP}(\texttt{I}(\kappa_r)) \geq \ell_{MMSP_{\kappa_r}}$, where $\ell_{MMSP_{\kappa_r}}$ is the length of $MMSP_{\kappa_r}$, the MMSP between the read's suffix starting with κ_r and the interval $\texttt{I}(\kappa_r)$. The next k-mer lookup starts from the $(MMSP_{\kappa_r} - k + 1)$-th position. By restricting our match extensions to reside within the MMSP, we ensure that we will not neglect to query any k-mer that might *expand* the set of potential transcripts where our read may map. We note here both the theoretical and practical relation between the $MMSP$ matching procedure, and the concept of a uni-MEM, as introduced by Liu et al. [56]. The k-safe-LCP for suffix array intervals are closely related to the lengths of unipaths in the reference de Bruijn graph of order k. Thus, our procedure for finding $MMSP$s, that limits match extension by the k-safe-LCP, is similar to the uni-MEM seed generation procedure described in deBGA [56], with the distinction that in our method, we only consider extending seeds in one direction, and that we also choose not to terminate the k-safe-LCP when the set of implied reference transcripts corresponding to the interval decreases in cardinality.

Given all the suffix array intervals collected for a read end (i.e. one end of a

paired-end read), we take the *union* of all the transcripts they encode. Formally, if a read r maps to suffix array intervals labeled with $\mathcal{C}^{r_1}, \ldots, \mathcal{C}^{r_n}$, then we consider all transcripts in the set $\mathcal{C}^{r_1} \cup \mathcal{C}^{r_2} \cup \ldots \cup \mathcal{C}^{r_n}$, and the associated positions implied by the suffix array intervals. As shown in figure 3.8; this step is done before co-mapping.

We adopt a heuristic to avoid excessive k-mer lookups when we encounter a mismatch. When extension of an MMP is no longer possible, it is most probable that the mismatch results from an error in the read. If the mismatch is due to the presence of an error, then checking each k-mer overlapping this error can be a costly process. Instead, we move forward by a distance of $k/2$ in the read, and check the k-mer from the read such that the mismatch occurs in the middle position. If this k-mer lookup leads to another suffix array interval, we continue with the MMP extension process there; otherwise, we move again to the first k-mer that does not overlap this mismatch position. We observe that, in practice, the k-safe-LCP, and hence the MMSP lengths can be quite large (Figure 3.9).

3.5.3 Co-Mapping

After collecting the suffix array intervals corresponding to left and right ends of the read, we wish to exploit the paired-end information in determining which potential mapping locations might be valid. Hence, from this step onward, we use the joint information for determining the position and target transcripts. Given the suffix array intervals for individual ends of a paired-end read, the problem of aligning both ends poses a few challenges. First, a single read can map to multiple transcripts,

and we wish to report all equally-best loci. Second, there can be multiple hits from a read on a single transcript (e.g., if a transcript contains repetitive sequence), and extra care must be taken to determine the correct mapping location. Finally, there may be hits that do not yield high-quality alignments (i.e. long exact matches that are nonetheless spurious). To address the first and third points, we employ an edit distance filter to discard spurious and sub-optimal alignments. To address the second challenge, we devise a consensus strategy to choose at most one unique position from each transcript.

Before applying the above mentioned strategy, we remove transcripts that do not contain hits from both the left and right ends of the read. Formally, given two ends of a read r, r^{e_1} and r^{e_2}, and the corresponding suffix array intervals labeled with $\mathcal{C}^{r_1^{e_1}}, \ldots, \mathcal{C}^{r_n^{e_1}}$ and $\mathcal{C}^{r_1^{e_2}}, \ldots, \mathcal{C}^{r_m^{e_2}}$ respectively, we only consider transcripts present in the set $(\mathcal{C}^{r_1^{e_1}} \cup \ldots \cup \mathcal{C}^{r_n^{e_1}}) \cap (\mathcal{C}^{r_1^{e_2}} \cup \ldots \cup \mathcal{C}^{r_m^{e_2}})$. We further refine this set by checking the validity of the alignments these hits might support. Currently, we use two validity checks illustrated in figure 3.8. First, we apply an orientation-based check, and second, we employ a distance-based check. The orientation check removes potential mappings which have an orientation inconsistent with the underlying sequencing library type (e.g., both ends of a read mapping in the same orientation). The distance check removes potential alignments where the implied distance between the read ends is larger than a given, user-defined threshold ($1,000$ nucleotides by default).

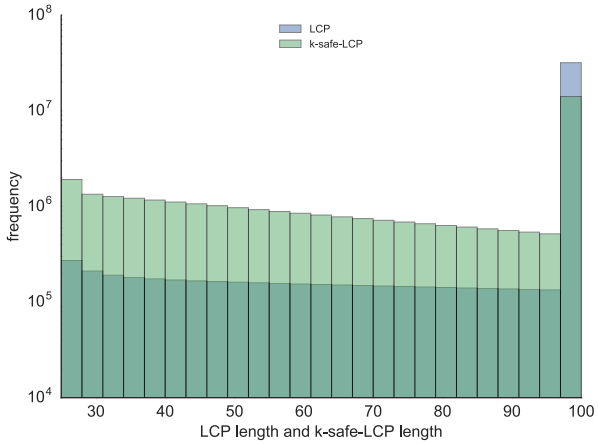


Figure 3.9: The distribution of k-safe-LCP lengths and LCP lengths are similar and tend to be large in practice (human transcriptome). Here, we truncate all lengths to a maximum value of 100 (so that any LCP or k-safe-LCP longer than 100 nucleotides is placed in the length 100 bin).

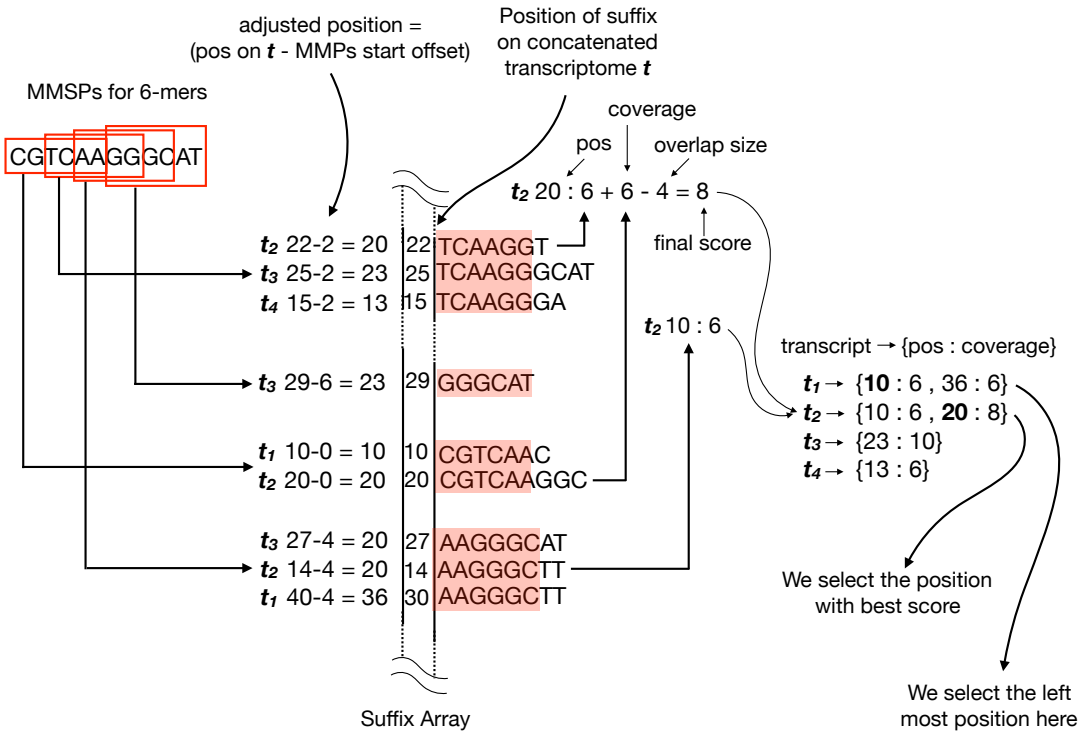


Figure 3.10: The MMSPs corresponding to a read are derived from multiple suffix array intervals. Here, all MMSPs happen to be of length k as LCPs are of size k. The coverage scheme finds out the exact positions on each transcript by adjusting the starting position of the MMSPs. The total score takes into account the positions where matches overlap. The final position is chosen by selecting the locus with maximum coverage.

3.5.3.1 Coverage based consensus

In selective-alignment, the potential positions on a transcript are scored by their individual coverage on the target transcript. Figure 3.10 depicts the mechanism of choosing the best postion on a transcript from multiple probable mappings to the same transcript. The coverage mechanism employed in selective-alignment makes use of the MMSP lengths collected during a prior step of the algorithm rather than simply counting k-mers. In figure 3.10, the transcript t_2 has two potential mapping positions given the reads: position 10 and 20. The coverage consensus mechanism selects position 20 over position 10 due to the higher coverage by tiling MMSPs on the read.

3.5.3.2 Selecting the best candidate transcripts

Once the positional ambiguity within a transcript is resolved, the next step is selecting the best candidate transcripts from a set of mappings. Since mapping relies on finding exact matches, the length of the matched subsequence between the read and reference can sometimes be misguiding when comparing different candidate transcripts. That is, the transcripts with the longest exact matches do not always su A block diagram of the steps described below are depicted in Figure pport optimal alignments for a read. At this point in our procedure, we follow the approach taken by many conventional aligners, and use an existing optimal alignment algorithm to compute the edit distance, by which we select the best candidate transcripts.

When performing alignment, we assume that a given read aligns starting at

the position computed in the previous steps. This helps us to reduce the search space within the transcript where we must consider aligning the read, and thereby considerably reduces the cost of alignment. To align the read at a specific position on the transcript and calculate the edit distance between them, we use $Myer's$ bounded edit distance bit-vector algorithm [57], as implemented in `edlib` [58]. For a fixed maximum allowable edit distance, this algorithm is linear in the length of the read. We note that the bounded edit distance algorithm we employ will automatically terminate an alignment when the required edit distance bound is not achievable.

We remove all alignments with edit distance greater than a user-provided threshold. This is similar to the approach used by many existing aligners, and allows us to specify that even the best mapping for a given read may have too many edits to believe that it reasonably originated from a known transcript in the index. An appropriate threshold should be based on the expected error rate of the instrument generating the sequenced reads, and a very low threshold can lead to a decreased mapping rate.

3.5.3.3 Enhancement of quantification accuracy based on edit distance

We investigated the effect of incorporating edit distance in downstream quantification. Since we integrated the selective-alignment scheme into the quantification tool *salmon* [28], the edit distance scores from selective-alignment can be used as a new parameter to *salmon*'s inference algorithm.

In the framework of abundance estimation, we define the conditional probability of a generating a particular fragment, f_j, given that it comes from a specific transcript, t_i, as $P(f_j \mid t_i)$. Given the edit distance between the fragment and the transcript, we can incorporate this parameter into this conditional probability. *Soft* filtering introduces a new term in the conditional probability based on $d_{i,j}$, which is the sum of the edit distances between the read ends of fragment f_j and transcript t_i. We set this probability according to an exponential function, $P(a_j|f_j, t_i) = e^{-4d_{i,j}}$. The aggregate of threshold filtering and *soft* filtering can be described as follows:

$$\Pr\left(a_j \mid d_{i,j}, t_i\right) = \begin{cases} 0 & d_{i,j} > threshold \\ e^{-4d_{i,j}} & d_{i,j} \leq threshold \end{cases}. \tag{3.1}$$

3.5.4 Shared LCPs prevents redundant alignments

Exploiting the common subsequences in the transcriptome is instrumental to the superior speed of fast mapping, non-alignment-based tools. Reads generated from exonic sequences common to multiple transcripts from the same gene or paralogous genes are the main source of ambiguous mappings. As we rely on the suffix array data structure to obtain the initial set of transcripts to which a read maps, there are cases where exactly identical reference sequences all act as mapping targets for the read. For a suffix array interval $[b, e)$, we identify such common subsequences by examining the *longest common prefix* (LCP) of the interval. If the length of the

Sample (SRR121)	5996	5997	5998	5999	6000
Skipped alignments	50.36%	54.85%	47.92%	48.06%	50.80%

Table 3.1: The percentage of hits that skip the full alignment process on five different experimental samples, due to extension by the maximum mappable safe prefix (MMSP), or projection of duplicate alignments given the longest common prefix (LCP) sequences.

LCP is equal or greater than the length of the read, then the actual alignment against the underlying reference at these positions will be identical. We observed (Table 3.1) that for almost half of the read-transcript pairs, the alignment process can be avoided. Note that if the read sequence shares a complete match with the common prefix, meaning that maximum mappable safe prefix length is equal to read length (i.e., the read matches the reference exactly at some set of positions), we can also bypass the Meyer's edit distance algorithm call completely.

Caching alignment sub-problems further avoids redundant work. We also extend a similar idea to the scenario where only part of the reference sequence is shared between references. Specifically, when performing an alignment between anchoring exact matches, we store the result in a hash table where the key is a tuple $(i, j, h\,(i', j'))$ and the associated value is the computed edit distance. Here, i and j denote the start and end of the read interval being aligned and i' and j' denote the start and end of the reference sequence; $h(i', j')$ is a hash of the corresponding reference sequence (we use `xxhash` [59]). This allows us to detect when a redundant alignment sub-problem for a read is shared between references, and to reuse the cached result in such cases.

Although I have not described the performance of selective-alignment in terms of accuracy of quantification, table 3.1 shows that utilizing the inherent shared sequences from the reference can avoid redundant mapping.

Chapter 4: Using redundancy to simulate realistic droplet based single cell RNA-Seq sequences

4.1 Background

Over [1] the past few years, advancement in massively-parallel sequencing technologies has enabled analysis of genomes and transcriptomes at unprecedented accuracy and scale by dramatically reducing the cost of sequencing and assaying more properties of cells. These new technologies are rapidly evolving, and new assays are constantly being introduced. One of the most exciting and popular recent developments has been the creation of experimental protocols for assaying gene expression in thousands of individual cells [61], allowing scientists to capture a snapshot of the dynamic and complex biological systems at work.

Individual cells are fundamental units of biological structure and function, and understanding the dynamics of gene expression patterns is essential for understanding cell-types, cell-states, and lineages. Single-cell gene expression studies have proven useful for unveiling rare cell types [62], abnormal cell states in the development of disease [63], and transcriptional stochasticity [64], by giving unprecedented insights into the dynamics of gene expression. Specifically, the accuracy,

[1] This section is reproduced from Sarkar et al. [60]

sensitivity, and throughput of droplet based single cell RNA-sequencing (dscRNA-seq) [65, 66, 67] has been particularly useful to scientists for understanding cellular dynamics through pseudo-time inference [68], estimation of splicing dynamics (i.e. RNA-velocity [69]), population balance analysis [67], spatial reconstruction to identify marker gene [70, 71], and numerous other analyses.

Most computational dscRNA-seq analyses pipelines work in multiple phases, the first of which is the generation of a gene-by-cell count matrix from raw sequencing data. This process includes identifying and correcting cellular barcodes (to determine properly-captured cells), mapping and alignment of the sequencing reads to the reference genome or transcriptome, and the resolution of UMIs to determine the number of distinct pre-PCR molecules sequenced from each gene within each cell. Subsequent downstream analysis of the count matrix is then used for a variety of different purposes, like lineage estimation, clustering and cell-type identification, identifying marker genes, estimating splicing rates, etc. Traditionally, gene-count matrices have been used as a fundamental unit of measurement for these analyses, and most research has been focused on developing new tools and improving methods for higher-level analyses (e.g. >90% of the tools described by Zappia et al. [72] deal with post-quantification analyses). Implicitly, these methods assume reliable and accurate input from the quantification phase.

Considerable research has been conducted into developing generative models of the gene-count matrices for a single cell experiment, and producing synthetic count data with known true phenotypes or labels. Many models have been proposed to capture various characteristics of single-cell experiments; for example, modeling

zero inflation [73, 74], characterizing heterogeneity [75], and inferring dropout rates [76]. Building upon this modeling work, a number of different simulation tools for single-cell RNA-seq data have been introduced. Count-level simulation tools like Splatter [77] and powersimR [78] use these generative models to directly simulate the gene-by-cell count matrices, and these simulators have been useful in developing and evaluating new methods for analyzing scRNA-seq data. However, that work has focused on simulating the gene-by-cell count matrices, rather than the raw sequencing reads that, when processed, give rise to this matrix. As such, these tools implicitly assume that the problem of estimating gene counts accurately from the raw sequencing data is relatively well-addressed. However, no principled tool is currently available for simulating the raw cell barcoded and UMI tagged read sequences for validating and assessing the initial processing phases of dscRNA-seq analysis. In contrast, numerous tools and methods have been proposed [38, 40, 79] for read-level simulation of bulk-RNA-seq experiments. Although no simulation tool can perfectly recapitulate all of the characteristics and complexities of experimental data, these tools have been crucial in helping to drive the development of ever-more-accurate approaches for gene and transcript-level quantification from bulk RNA-seq data.

In this paper, we present *minnow*, a comprehensive framework to generate read-level simulated data for dscRNA-seq experiments. *Minnow* accounts for many important aspects of tagged-end single-cell experiments, and models these effects at the sequence level. It models the process of UMI and cellular barcode (CB) tagging, molecular PCR amplification (including PCR errors and deviations from

perfect efficiency), molecular fragmentation, and sequencing. *Minnow* can generate synthetic sequencing reads that mimic other important aspects of experimental data, such as realistic degrees of gene-ambiguous sequencing fragments. Using the *minnow* simulation framework, we demonstrate how various dscRNA-seq quantification pipelines perform in generating gene-by-cell count matrices when validated on synthetic data with known ground truth. We describe and analyze the effect of modeling important characteristics of real experimental dscRNA-seq data, and show how sequence-level ambiguities like those present in real experimental dscRNA-seq data pose quantification challenges to existing pipelines.

4.2 Methods

4.2.1 The *minnow* framework

For droplet-based protocols [65, 66, 67], mRNA molecules are attached to a cell barcode (CB) and a unique molecular identifier (UMI). After reverse transcription within a droplet, the barcoded, tagged-end complementary DNA (cDNA) undergoes amplification and fragmentation, followed by sequencing in a short read sequencing machine (Illumina). Due to the small amount of biological material in each cell, such protocols typically undergo many cycles of PCR, making PCR sampling effects considerable, and necessitating the use of UMIs to discard reads sequenced from duplicate molecules (those that derive from the same pre-PCR transcript). The sequence files generated by such protocols result in paired-end reads and have two core components. One end (typically read 1) of each read pair contains the concatenated

nucleotide sequence of the CB and UMI, while the other end (typically read 2) is the cDNA representation of the mRNA molecule, usually as sequenced from the 3′ end [66]. In essence the scRNA-seq experiment is actually much like a single end RNA-seq protocol, where the CB contains the cell specific information and the UMI is used to identify PCR duplicates.

The core of *minnow* can be described as a composition of three steps; (1) selection of transcript, concordant with the target count matrix and properties of experimental data, for the initial pool of simulated molecules, (2) simulation of CB and UMI tagging, (3) simulation of PCR, fragmentation and sequencing. *Minnow* starts by consuming a gene count matrix as input, that provides the estimated number of distinct molecules within the sample corresponding to each gene within each cell. Then, based on a carefully chosen distribution of transcript isoforms, *minnow* distributes simulated molecules (transcripts) to each gene within each cell. Taking these as the initial biological pool of molecules to be sequenced, *minnow* then tags these molecules with cell barcodes and unique molecular identifiers. Finally, it simulates the PCR, fragmentation and sequencing process. In general, since PCR is an exponential stochastic branching process, it generates more amplified molecules than reads obtained in a real experiment. We simulate the sequencing process by subsampling from the final amplified pool of PCR generated molecules to a sequencing depth specified by the user. During this generation process, we have tried to capture core attributes of single cell RNA-seq at different levels, such as preserving the proportion of ambiguously mapped reads, simulating the PCR amplification bias, introducing sequence error using different error models etc. It is important

to note that *minnow* does not contain a generative model in itself, and depends on the matrix given as input. Therefore if a count matrix produced from a simple generative model is used, *minnow* will produce read sequences consistent with those counts while if an experimental count matrix is provided as input, artifacts like doublets, empty droplets etc. are likely to be reflected in the generated reads. While the development of a generative model that accounts for all of the nuances of experimental droplet-based scRNA-seq data is still an active area of research, *minnow* focuses, principally, on how to generate a realistic set of sequencing reads consistent with the provided count matrix. To the best of our knowledge at the time of writing this tool *minnow* is the only comprehensive framework that simulates droplet based scRNA-seq dataset at read level.

Presently, *minnow* focuses only on droplet-based single cell protocols. So, for the rest of the paper, we refer to such datasets as scRNA-seq data. Following the general principle of common bulk RNA-seq simulators [38, 40], *minnow* follows a two step process of simulation. If the experimental data (the raw `FASTQ` files) is provided, then we use *alevin* [45, 80] to obtain mapping and quantification results, and to learn a number of other auxiliary parameters that we use to generate a realistic simulation. To be specific, *alevin* generates a gene-by-cell count matrix, and other parameters, that *minnow* (invoked in `alevin-mode`) can directly consume. In the absence of a specific experimental dataset, *minnow* makes use of the *Splatter* [77] tool to generate a gene-by-cell count matrix (when invoked in `splatter-mode`). While in the present manuscript we have considered the results related to *Splatter*, we note that *minnow* can be paired with any such method [81] that realistically generates cell to gene count

matrix. Additionally, *minnow* accepts other attributes of a scRNA-seq experiment, starting from a set of pre-specified cell barcodes, UMIs, the number of per-cell molecules, and cell-level cluster information. Subsequently, *minnow* generates the paired-end `FASTQ` files, along with the "true" gene count matrix, cell names and gene names. The `FASTQ` files follow the common 10x-chromium [66] (compatible with both version 2 and 3) format, and can be consumed by any downstream tool capable of processing such files.

4.2.1.1 Sampling from gene count matrix

The main input to *minnow* is the gene count matrix $\mathcal{M}$, where each element m_{ij} denotes the number of UMIs (either estimated or simulated by a count-level simulator) for gene g_j in cell i. How the gene count matrix is used to select transcripts depends upon whether the provided input is estimated empirically from an experimental dataset or is a simulated count matrix.

Minnow takes the complete expression of a particular cell as a vector, and treats the normalized values as the parameters $\boldsymbol{\alpha}_i = (\alpha_{i1}, \ldots, \alpha_{iM})$ of a multinomial distribution f_i. The probability α_{ij} denotes the total probability of selecting some pre-PCR transcript to derive from gene g_j in this simulated cell. Then, *minnow* samples c_i such molecules from the distribution f_i, where $c_i = \sum_j m_{ij}$. This characterization allows the input matrix to have either integral or non-integral abundances — the former is important in the case that the count matrix is derived from a tool such as *alevin*, that may yield non-integral gene counts when attempting to account

for gene multimapping reads. After this sampling step, *minnow* ends up with a *ground truth* gene count matrix of integral counts that we denote as $\mathcal{T}(\mathcal{M})$.

A major challenge for generating simulated sequences from gene counts is the lack of information about the transcript-level expression for a particular gene. Simulating the amplification and fragmentation of molecules and sequencing of fragments requires selecting which specific molecules (i.e. transcripts) contribute to the expression of each gene. Since most genes have multiple isoforms, which can share exons and vary widely in their sequence composition, there is no such thing as the canonical sequence at the level of a gene. However, the molecules selected for amplification, fragmentation, and sequencing can have a tremendous impact on the specific characteristics of the simulated dataset (e.g. how many reads map ambiguously back to the genome, and the resulting difficulty of quantification). Full-length scRNA-seq protocols (such as SMART-seq [82]), aim to achieve uniform coverage over transcripts, meaning that numerous reads covering distinctive splice junctions are often sequenced, and transcript-level abundances can often be assessed [83]. This means that bulk-RNA-seq simulators [38, 40], though by no means a perfect match for such protocols, can plausibly be used for generating read level simulated data. On the contrary, for tagged-end protocols, such frameworks do not seem to replicate the fundamental properties of real data [84].

Minnow addresses this challenging problem in one of two ways, depending upon whether $\mathcal{M}$ is an empirical estimate or a simulated count matrix. When the relevant parameters are trained from an experimental dataset, *minnow* follows an empirical estimation method to determine individual counts for candidate isoforms

as described in section section 4.2.1.4. On the other hand, when the experimental data is not present (e.g. when using *Splatter* to generate $\mathcal{M}$), it uses previously-estimated measures of sequence-level ambiguity from similar experiments in the same species. Alternatively, when neither of the above cases apply, *minnow* can optionally use a Weibull distribution, with a pre-specified shape parameter 0.44 and scale parameter 0.6 to determine the individual dominance of candidate isoforms (motivated by Hu et al. [85]).

4.2.1.2 Indexing inherent sequence ambiguity using read-length de Bruijn graphs

To replicate the gene-level ambiguity present in real scRNA-seq dastasets [80], *minnow* constructs an index that maps segments of the underlying transcriptome to the number of distinct transcripts (and genes) in which they occur. This allows selecting transcripts for amplification and fragmentation in a manner that will lead to realistic levels of ambiguity in the resulting simulated reads.

Specifically, *minnow* starts with a pre-constructed de Bruijn graph [86], with a k-mer size set to the read length, built over the reference transcriptome.

The de Bruijn graph is commonly specified as a graph $G = (V, E)$, built over the collection of k-mers (strings of length k) from an underlying set of sequences S (in the case of *minnow*, the set of transcripts), with an assumption that all members of S are at least of length k. Given a specified k, the set of vertices of G are k-mers from the members of S. An edge exists between two nodes of G if and only if there

exists a $(k+1)$-mer in any of the underlying sequences of S containing both of these k-mers. Given such a representation, any sequence in S can be spelled out as a *path* in G.

Rather than the de Bruijn graph, *minnow* makes use of the compacted de Bruijn graph. In a compacted de Bruijn graph G_c, unlike in the de Bruijn graph, an edge can be of length greater than $(k+1)$, and is obtained by compressing or compacting non-branching paths in G to form unitigs (for detail, see [87]). We adopted TwoPaCo [87] to efficiently construct the compacted colored de Bruijn graph, which can be directly converted to a graphical fragment assembly (GFA) file. The color in the compacted de Bruijn graph also captures the label of the reference sequence (transcript) as an attribute (color) of the k-mer, and only paths having the same color are compacted into unitigs. The GFA file represents the compacted edges as a set of unitigs $\mathcal{U}$, and stores the relation between the set of transcript sequences T and $\mathcal{U}$ by describing how each transcript is spelled out by a path of the enumerated unitigs. Specifically t_i can be represented by an ordered list (i.e. path) of unitigs: $\mathcal{P}(t_i) = (\langle u_{k_1}, o_{k_1}\rangle, \langle u_{k_2}, o_{k_2}\rangle, \ldots, \langle u_{k_n}, o_{k_n}\rangle)$, where $u_{k_i} \in \mathcal{U}$ and $o_{k_i} \in \{+, -\}$. The '+' and '−' respectively specify the orientation of the unitig as traversed in the path, with '+' meaning the unitig appears unchanged and the '−' meaning the unitig appears in the reverse-complement orientation within the path. When concatenated with proper stranded-ness (and accounting for the length k overlap between successive unitigs), the series of unitigs in $\mathcal{P}(t_i)$ reconstructs the sequence of t_i exactly. Given any ordered list $\mathcal{P}(t_i)$, we observe that the locations of occurrence and orientation of a unitig within $\mathcal{P}(t_i)$ can be trivially extracted.

With respect to *minnow*, the relevant information from the GFA file is a combination of the two sets; the set of unitigs $\mathcal{U}$, and the set of transcript sequences T (stored as paths of unitigs). Given the set of unitigs $\mathcal{U}$, *minnow* only considers the unitigs that occur within a `MAX_FRAGLEN` base pair distance (1000 bp by default) from the $3'$ end of at least one transcript. This imposes a primary constraint of the tagged-end scRNA-seq protocol, and restricts *minnow* to generate fragments no longer than `MAX_FRAGLEN`. The intuition behind fixing an `MAX_FRAGLEN` arises from the fragment length limit of a real illumina sequencer [88].

Minnow uses $\mathcal{P}$ and $\mathcal{U}$ to construct a unitig-level mapping, describing which transcripts contain each unitig, and how the unitig covers the transcript in terms of its position and orientation. Recall that for the de Bruijn graph construction we select the size of each $k-mer$ to be the read length. This implies that each unitig is at least as long as a read and therefore, capable of generating a sequencing read that could then align back to all the transcripts containing that unitig. The mapping for a particular unitig $u \in \mathcal{U}$ is given as $\mathcal{E}_{\mathcal{U}}(u) = \mathcal{E}_u = \mathbf{t}_u$, where $\mathbf{t}_u$ is set of triplets of the form $\langle t_i, p_i, o_i \rangle$ and $\forall \langle t_i, p_i, o_i \rangle \in \mathbf{t}_u$, we observe $\langle u, o_i \rangle \in \mathcal{P}(t_i)$. Then $\forall \langle t_i, p_i, o_i \rangle \in \boldsymbol{t}_u$ we have that $t_i[p_i : p_i + \text{len}(u)] = u^{o_i}$, where u^{o_i} is simply the sequence of unitig u when written in orientation o_i. In other words, the mapping contains the information about the set of transcripts that contain the unitig u at least once and how the unitig u *appears* within those transcripts. We refer to the set of mappings for all unitigs as $\mathcal{E}_{\mathcal{U}}(\cdot)$. This representation of $\mathcal{E}_{\mathcal{U}}(\cdot)$ is helpful for two reasons. First, it is independent of any particular experiment and can be prepared just with the reference annotation and a specified read length. Second, it

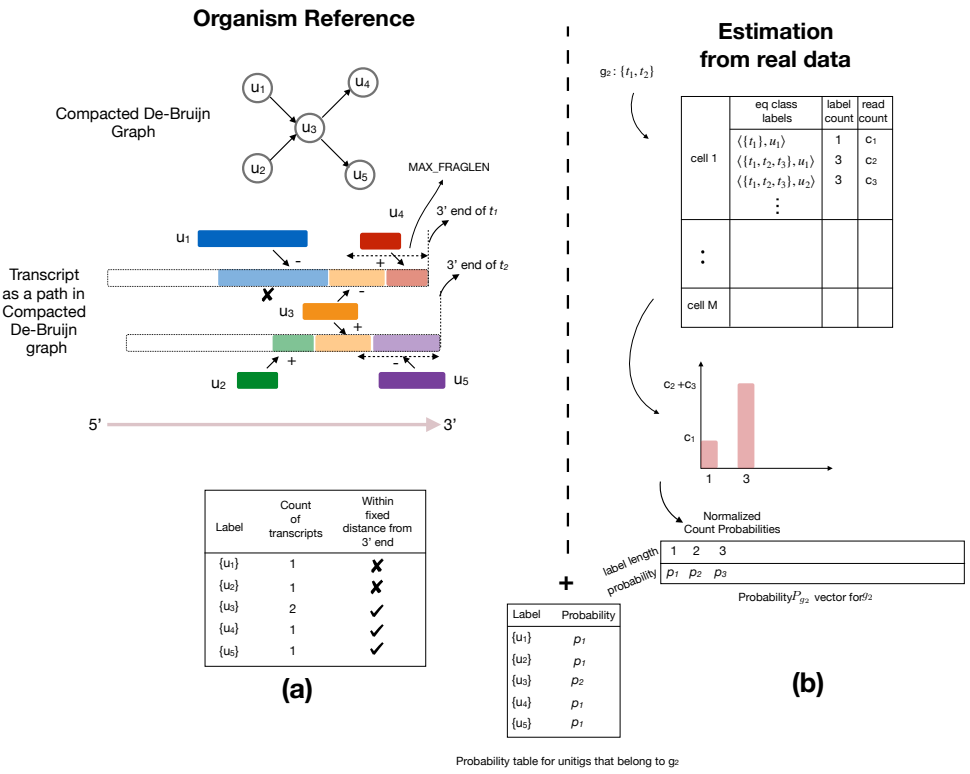


Figure 4.1: Overview of the *minnow* pipeline: On the right hand side (a) the construction of unitig-based equivalence classes is depicted based on the compacted de Bruijn graph constructed from reference sequences. Unitigs u_1 and u_2 are discounted as they are more than `MAX_FRAGLEN` bp away from $3'$ end of transcripts t_1 and t_2. Further the equivalence class is constructed as discussed in section 4.2.1.3. On the right hand side (b) the transcript level equivalence class structure obtained from *alevin* is used to derive a per-gene probability vector. Finally the probabilities are mapped directly to the unitig labels.

naturally implies the positional intervals that are globally shared between reference sequences, allowing the selection of intervals that give rise to the desired degree of sequence multimapping. Moreover the structure of $\mathcal{E}_{\mathcal{U}}(\cdot)$ is gene oblivious, but can be readily induced for the set of genes given the known transcript to gene mapping.

4.2.1.3 Learning realistic sequence level ambiguity

While gene-by-cell count matrices are broadly used for almost all analyses downstream of scRNA-seq quantification, we recognize that they fail to capture many important characteristics of a real experiment, for example the UMI level ambiguity, potential UMI collisions, transcript and further gene level multi-mapping etc. On the other hand, the raw `FASTQ` files obviously encode all of the subtle characteristics of an experiment, but are often enormous in size and are also massively redundant. As shown in the recently-introduced tool *alevin* [80], accounting for gene-level ambiguity in the sequencing data can have important implications for the resulting quantification estimates. UMI graphs [89] or **p**arsimonious **U**MI **g**raphs (referring to the graphs as constructed by *alevin*, and which we denote as PUGs [80], succinctly encode most of the relevant characteristics of the experiment, while being much more concise than the set of raw reads. The construction of PUGs in *alevin* depends upon the structure of equivalence classes of transcripts and UMIs, which themselves depend upon the manner in which the UMI-tagged sequencing reads within a cell map to the underlying transcriptome (detail can be found in [80]).

We use this intermediate structure to estimate the level of sequence ambiguity that is present in a particular experiment. We extract gene level ambiguity information from the structure, and use it to help simulate fragments that simultaneously match the observed gene count and also display a realistic level of gene-level sequence ambiguity. Specifically, given a particular gene, we only look for equivalence classes from *all* cells that contain at least one transcript from that gene. As

depicted in Figure 4.1(b), we are interested in two quantities from such equivalence classes: (1) the equivalence class cardinality, i.e. the length of the set of transcripts, and (2) the equivalence class frequency, i.e. the number of reads that belong to that class. For an equivalence class $\mathcal{E}_{\mathcal{U}}(u) = \mathcal{E}_u$, these are termed as *label count* ($\text{len}(\mathcal{E}_u)$) and *read count* ($\text{freq}(\mathcal{E}_u)$) respectively in Figure 4.1(b). The label count signifies the degree of multi-mapping, and the read count captures the frequency of such multi-mapping. We note that both of these pieces of information are specific to a particular experiment. This transcript equivalence class level information about *label count* and *read count* can be transferred directly to the gene level. Given the gene g_i, we first single out the equivalence classes such that at least one of the transcripts in the equivalence class label belongs to g_i. Then, we produce a probability vector P_{g_i} defined from these equivalence classes, where the random variables are the *label counts* and the probabilities are normalized *read counts* corresponding to them. This provides a representation that allows us to select specific transcripts for amplification and sequencing that simultaneously match both the *gene-level* counts in the input count matrix as well as the degree of sequence-level gene ambiguity that we observe in experimental data.

Alternatively, instead of defining the label count probability vector P_{g_i} for a gene globally over an entire experiment, *minnow* also supports defining cluster-local probability vectors when such information is given. Specifically, instead of normalizing over all cells, we normalize over all cells within a cluster (e.g. predicted cell type), deriving a probability vector $P_{\langle \mathcal{C}_i, g_i \rangle}$, for each expressed gene g_i in a cluster $\mathcal{C}_i$. This allows accounting for the fact that, due to changes in the underlying

transcript expression that vary between clusters, the probability of observing (and hence generating) gene-ambiguous fragments may also change.

4.2.1.4 Assigning probabilities to $\mathcal{E}_{\mathcal{U}}$

Given the $\mathcal{E}_{\mathcal{U}}(\cdot)$ as defined in Section 4.2.1.2, and $P_{\langle \mathcal{C}_i, g_i \rangle}$ or P_{g_i} as defined in Section 4.2.1.3, we finally map the probability for each unitig from $\mathcal{E}_{\mathcal{U}}(\cdot)$. Formally, we define a function $f_{g_i} : \xi \to \mathbb{R}$, for all unitigs within gene g_i, as

$$f_{g_i}(u) = \begin{cases} p_{\ell_i} & \text{if } \text{len}(\mathcal{E}_u) = \ell_i \quad \text{and} \quad P_{g_i}(\ell_i) = p_{\ell_i}, \\ 0 & \text{otherwise} \end{cases}$$

Qualitatively, $f_{g_i}(u)$ represents the probability of sampling unitig u from gene g_i to be equal to the empirically observed probability p_{l_i} of equivalence classes containing g_i that share u's label count l_i. When provided, the cluster specific information can also be used here to derive an analogous function $f_{\langle \mathcal{C}_i, g_i \rangle}$ based on the cluster specific gene level probability vector $P_{\langle \mathcal{C}_i, g_i \rangle}$.

Both $\mathcal{E}_{\mathcal{U}}(\cdot)$ and f_i, when used in conjunction with each other, give *minnow* the ability to sample the reads from references in a realistic manner. Given a gene g with count x, the selection process of candidate transcripts and underlying unitigs are as follows. *Minnow* first selects the set of unitigs that are part of the candidate transcripts. When there are multiple unitigs that can be potentially used, *minnow* initializes a multinomial distribution with parameters according to f_g, where the random variable is the unitig to be selected next. Under this distribution, after x

such random draws, x unitigs are selected (with replacement). For each such unitig, *minnow* randomly assigns the unitig to a corresponding transcript within that gene by scanning $\mathcal{E}_{\mathcal{U}}(\cdot)$. In this composite selection process, while $\mathcal{E}_{\mathcal{U}}(\cdot)$ and $\mathcal{U}$ determine the possible reference interval from which to sample, f_i determines the probability of such sampling and, finally, $m_{i,j}$ dictates the sample size.

4.2.2 Generating RNA-seq sequences

4.2.2.1 Generation of cell barcodes and UMIs

Generating reads that match empirical degrees of gene-level ambiguity is one part of realistic single cell RNA-seq data generation. Another main component is the generation of cell-barcodes (CB) and UMI sequences, properly affected by amplification and sequencing errors. In *minnow* we concentrated on mimicking the process as it occurs in 10x data, though the modular nature of the tool makes designing modules for other droplet-based protocols straightforward. The length of the CB and UMI sequences can be provided by the user, by default we follow 10x v2 protocol for generating the CB and UMI sequences. Given the number of unique molecules (present in the experiment) provided in terms of the gene count matrix, and the randomly generated UMI sequences, we perform a random assignment between UMIs and molecules. Once generated, a complete unit of sequence that undergoes polymerase chain reaction (PCR) consists of a concatenated string of CB, UMI and the corresponding reference molecule (i.e. transcript) as sampled from $\mathcal{E}_{\mathcal{U}}(\cdot)$.

4.2.2.2 PCR amplification and imputing sequence error

Once the set unique molecules are prepared, we simulate amplifying the molecules through multiple PCR cycles to generate a realistic distribution of PCR duplicates. We follow the standard protocol for PCR simulation [90, 91]. The PCR model in *minnow* can be described as a set of unbalanced probabilistic binary trees. The stochasticity of the model is determined by the probability of capture efficiency p_{eff} that can be externally set (default $p_{\text{eff}} = 0.98$). At each cycle, with probability p_{eff}, molecules are chosen for duplication. In each duplication step, a nucleotide in the molecule is mutated with probability p_{error} (default $p_{\text{error}} = 0.01$). At the end of simulating the PCR cycles, *minnow* randomly samples an appropriate number of molecules from the (duplicated) pool of molecules. Apart from providing this simple model, *minnow* also allows an optional flag to mimic the empirically supported model of PCR (described as `Model 6` in [90]), where the efficiency of individual molecules is allowed to vary and is sampled from a normal distribution $N(\mu, \sigma)$ (default values $\mu = 0.45$, $\sigma = 0.2$), and subsequently all the duplicated molecules inherit the efficiency from their parent. This step is highly stochastic, and can be customized to simulate PCR amplification given the uneven distribution of capture efficiency for individual molecules. We have implemented several optimizations, to reduce the computational burden associated with simulating PCR sequence duplication.

4.2.2.3 Start position sampling

Given the set of sampled duplicated PCR-ed molecules, *minnow* simulated the sequencing process of actual reads. As discussed in section 4.2.1.4, *minnow* utilizes the precise location of the unitig on the transcript from which the read is drawn. The unitig acts as a `seed` for sampling read sequences from the reference. Given a unitig u, with offset position p and length l on a transcript t, we resort to two different mechanisms for determining start position. If the user chooses to use the empirical distribution P_{fld} of fragment lengths, then we randomly sample a fragment from a closed interval $[p - slack, p + slack]$ where probability of each fragment length is dictated by P_{fld}, and *slack* variable is set with proportional to the *read_len*. While it is highly recommended to use an empirically-derived read start position distribution (which comes packaged with *minnow*), in absence of such quantity, *minnow* samples from a truncated normal distribution $N(\mu, \sigma)$ with $\mu = (L - l/2)$ and a user defined σ (default set to 10). The intuition behind using such a distribution is to simultaneously use the ambiguity information while avoiding sampling reads from the same exact region again and again. This variance in sampling is important, since fragmentation occurs after the majority of amplification in the protocols we are simulating, and so "duplicate" reads (reads from the same pre-PCR molecule) will tend to arise from different positions. Moreover, this framework creates an option for user to tune the σ in order to move away from unitig dependent sampling. Upon selecting the sequence *minnow* can be instructed to impute sequencing error in the sampled sequence. *Minnow* can accept a variety of error-model for substituting

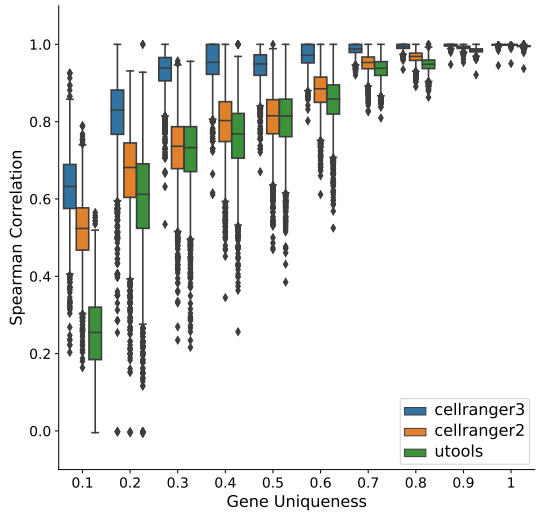


Figure 4.2: Performance of quantification tools, stratified by gene-uniqueness, under a "basic" configuration (based on pbmc 4k dataset).

nucleotides, and follows the same format of error model as used in [40].

As its output, *minnow* produces the true abundance matrix $\mathcal{T}(\mathcal{M})$, along with the raw `FASTQ` files containing the simulated reads. This enables rigorous testing of different quantification tools such as Cell-Ranger [66], UMI-tools [89], etc. Hence, it is important to validate if the simulated reads generated by *minnow* are able to reproduce some of the challenges faced by these tools while consuming experimental data. One such factor that contributes to the performance of these tools is the mechanism by which they deduplicate UMI sequences [80]. We observe that when we stratify the genes with respect to their gene uniqueness, the divergence between different tools start to emerge. As a proof of concept, we have run the popular quantification tools Cell-Ranger (2.1.0 and 3.0.0) and UMI-tools on *minnow*-simulated data. The UMI-tools based pipeline we used is a custom one, which uses STAR [92] for alignment, and UMI-tools [89] for UMI resolution, and featureCounts [93] for deduplicated UMI counting — we refer to this as the *naïve* pipeline.

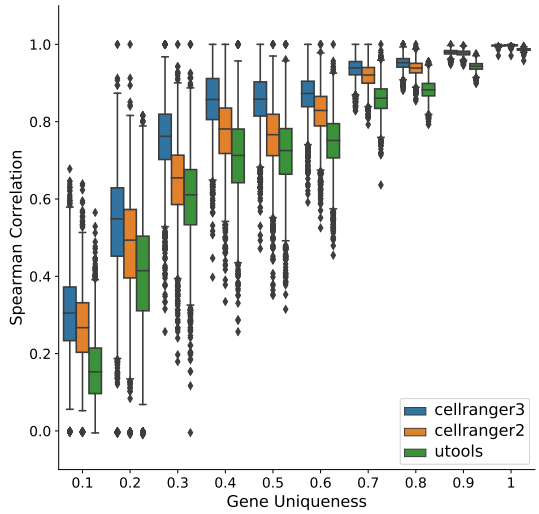


Figure 4.3: Performance of quantification tools, stratified by gene-uniqueness, under a "realistic" configuration (based on pbmc 4k dataset).

The estimated abundances are correlated (spearman and pearson) with the true gene-count provided by *minnow*. We note that number of cells detected by a downstream tool varies from what is initially present in the raw `FASTQ` files. Therefore, while calculating a cell specific local correlations, we consider only the subset of the cells that are predicted by all tools.

Minnow accepts a host of different input parameters, such as PCR-related mutation error, sequencing error, predefined custom error model for substitution, UMI pool size, number of PCR cycles etc. As Cell-Ranger takes considerable time to finish (4-6 hours for 4000 cells), we have limited ourselves to one run of 4K cells of all tools, demonstrating the variance of different tools in the data. We observe that *minnow* scales efficiently with increasing numbers of cells. That is, the bottleneck in our assessments was the time required by the quantification tools, as *minnow* can simulate millions of reads in a few minutes. Thus, for demonstrating other effects (e.g. how the choice of different degrees of sequence ambiguity in mapping

Splatter-generated counts to specific genes affects quantification accuracy), we have limited our assessment to smaller simulated datasets consisting of 100 cells each. An assessment of the computational performance of *minnow* is provided in table 4.2.

4.2.3 Datasets

We have used two different datasets for benchmarking. For a global analysis, in terms of downstream quantification accuracy, we have used the publicly-available peripheral blood mononuclear cells dataset, referred as pbmc 4k dataset [66], obtained from the 10x website. As discussed in section 4.2.1, we first used *alevin* to estimate the gene-count matrix and the relevant auxiliary parameters, like the equivalence classes. The gene-count matrix along with the auxiliary files are then used as an input to *minnow*. *minnow* is run with two different configurations. First, we consider the "basic" configuration, where no information other than the gene-count matrix from *alevin* is used. In the other configuration, referred as "realistic", we have made use of the equivalence classes produced by *alevin*, in addition to the compacted de Bruijn graph built on the transcript by TwoPaCo. The "realistic" configuration also used the empirical fragment length distribution. In both the cases we have used normal mode for PCR with fixed capture efficiency.

The other datasets we used are generated by *Splatter* [77]. *Splatter* accepts multiple parameters as input, including the generative model to use, the number of cells, the number of genes to express, etc. We used the `splat` model from *Splatter* to generate the gene count matrix with default parameters. To manage the time

consuming tools (Cell-Ranger-2,3) the analysis is restricted to 100 cells and $50,000$ genes.

4.2.4 The presence of gene level ambiguity drives the difficulty of UMI resolution and shapes the accuracy of downstream tools

With moderate numbers of fragments being sequenced per-cell and realistic diversity in the UMIs available to tag the initial molecules, existing pipelines appear to do a reasonable job of estimating the number of distinct molecules sampled from each gene within a cell. Yet, one major factor that appears to affect the accuracy of these different approaches is the level of sequence (specifically gene-level) ambiguity present in the simulated data. This is not particularly surprising, as neither Cell-Ranger (either version) nor the *naïve* pipeline are capable of appropriately handling such situations (i.e. to where should a UMI be assigned if its corresponding read maps equally well between multiple genes?). However, understanding this effect seems important, as the degree of gene-ambiguous reads in a typical scRNA-seq ranges from $\sim 13 - 23\%$ of the total reads [80].

To study the effect of gene-level ambiguity on quantification pipelines, we have stratified the global correlation plot with respect to the *uniqueness* score of a gene. The uniqueness score is specified by the ratio of two quantities: number of k-mers unique to a gene (i.e. only appearing within transcripts of this gene) to the total number of distinct k-mers present in the gene. According to this metric, the most unique gene will have a score of 1 and the least a score of 0. The stratified plots

(Figures 4.2 and 4.3) demonstrates that in the "basic" simulation, the performance of all three tools are globally high. Interestingly, we note that Cell-Ranger-3 does slightly better than Cell-Ranger-2 (which in turn performs better than the *naïve* pipeline) for non-unique genes, despite the fact that it does not have a specific mechanism for resolving such cases. On the other hand in the "realistic" simulation, where gene-level sequence ambiguity was sampled to match the degree observed in *even the most unique* experimental data (this dataset exhibits $\sim$ 10% gene-level multimapping meaning over 10% sequencing reads map to multiple genomic loci), the global performance decreased. One possible interpretation of such a considerable performance gap between different tools can be the specific method each method uses to categorize alignments (though all pipelines use STAR [92] as their aligner), as well as differences in the algorithms they use to resolve UMIs. It is clear, however, that failing to resolve the multi-mapped reads can have a detrimental effect the accuracy of all of the quantification pipelines. The performance gap is magnified when we increase the fraction of reads sampled from less-unique gene sequences. Thus, it should be noted that failing to model empirically-observed levels of sequence ambiguity can result in the simulation of unrealistic sequenced fragments, that fail to capture important characteristics of real data, and where the variability of different tools are hard to spot. Thus, to create more realistic and representative simulated data, it appears to be important to carefully match the degree of gene-level multi-mapping observed in experimental data.

In the *Splatter*-based simulations, we have explored the performance of quantification pipelines on data representing a spectrum of degrees of sequence-level am-

biguity. *Splatter* uses generative models to produce the gene count matrix. Therefore, while the distributional characteristics of the simulated counts are designed to accord with experimental data, there is no specific relationship between the generated counts and specific gene present in an organism. While this presents a challenge — different mappings of simulated counts to different genes will result in different quantification performance — it also presents a degree of freedom to explore how the performance of tools changes as we hold the counts fixed, but alter the mappings between counts and specific genes. We use this freedom to model extreme situations, and to explore the sensitivity of different quantification pipelines as we alter how counts are mapped to specific genes. Specifically, we considered three different scenarios. First, we use the gene uniqueness scores defined above to sort the set of genes in increasing order (from least to most unique). Then, we assign the gene with lowest uniqueness score to the highest expressed gene label (aggregated over all cells) from the *Splatter*-obtained matrix and so on. This biased allocation purposefully increases the challenge of resolution for downstream tools. We call this configuration "adversarial". In the second configuration, we have selected a random set of genes and followed the "realistic" configuration by using the reference based compacted de Bruijn graph and learned gene-level ambiguity distribution (as discussed in section 4.2.1.2), following the same convention. We call this simulation "realistic". Finally, the third configuration repeats the process of "adversarial" but in reverse order, i. e., assigning the most-unique gene to the most abundant gene label. This final configuration is a situation where we expect to see minimum level of ambiguously mapped reads, thereby making the quantification challenge relatively

Configuration	Correlation (Spearman)			MARD		
	CR2	CR3	*naïve*	CR2	CR3	*naïve*
adversarial	0.811	0.809	0.723	0.075	0.076	0.107
realistic	0.920	0.915	0.880	0.043	0.046	0.076
favorable	0.957	0.952	0.936	0.031	0.035	0.047

Table 4.1: Spearman correlation and MARD (mean absolute relative difference) are calculated with respect to ground truth under three different configurations based on the same gene-count matrix produced by *Splatter*. CR2 and CR3 stands for Cell-Ranger-2 and Cell-Ranger-3 respectively.

easy for all downstream tools. We call this configuration "favorable".

Table 4.1 shows the global correlation between three tools for all the above scenarios. We observe that for the "adversarial" data, the performance of all pipelines is considerably depressed. Conversely, for the "favorable" dataset (bottom row), the performance of all pipelines are considerably improved — and the gap between the different versions of Cell-Ranger and the *naïve* pipeline is reduced. The "realistic" simulation falls nicely between these two extremes, suggesting that, while not as pronounced as one observes in the "adversarial" scenario, the discarding of gene-ambiguous reads can have a considerable effect on quantification accuracy, even under realistic degrees of gene-level sequence ambiguity.

4.2.4.1 *Minnow* maintains global structure in the simulated data

To compare the internal structure of the *minnow* generated data, we have compared the data metrices that are given to and produced by *minnow*. From a tSNE projection of the real and *minnow* generated data, we observe that the number of clusters remains the same while some of the clusters are perturbed. This

indicates that the minnow produced data can be effectively used for cell type specific analysis. Additionally, we have calculated *variation of information* (VI) [94] as a metric to measure the distances between the clusters obtained by running Seurat's clustering [95] algorithm on the count matrices produced by different methods, with respect to the clustering produced on the true count matrix, as output by *minnow*. The VI distances are 0.589, 0.613 and 0.632 for clusterings on the matrices produced by Cell-Ranger-3, Cell-Ranger-2 and *naïve* respectively. While all of the tools do reasonably well, as expected, Cell-Ranger-3-derived clusters are closer to the truth than the other methods.

reads	cells	PCR cycles	threads	time (hh:mm:ss)	mem. (KB)
100M	1,000	4	8	0:10:44	7,556,108
100M	1,000	4	16	0:5:39	11,163,216
100M	1,000	7	8	0:16:56	8,723,320
100M	1,000	7	16	0:9:01	13,449,888
800M	8,000	4	8	0:56:28	28,249,676
800M	8,000	4	16	0:31:18	31,855,624
800M	8,000	7	8	1:43:32	29,246,148
800M	8,000	7	16	0:53:15	34,217,500

Table 4.2: Timing and memory required by *minnow* to simulate data with various parameters

4.2.4.2 Speed and memory of simulated sequence generation

The execution of *minnow* proceeds in two phases. First, it loads the gene-count matrix (as either a compressed binary or plain-text file) in a single thread. Then, it spawns multiple parallel threads (as many as specified by the user) to simulate reads deriving from different cells. This mechanism enables the most computation-

ally intensive process (simulating PCR and generating read sequences) to happen independently and in parallel for each cell. This enables *minnow* to scale well with the number of threads when writing millions of reads to the disc. In table 4.2 we have varied a number of parameters to test the computational requirements of *minnow* for generating simulated datasets of various sizes. Namely, we have varied the number of PCR cycles, the total number of cells simulated, and number of threads used. The total wall clock time and peak resident memory reported using `/usr/bin/time`. We have limited the highest number of reads (default is 100,000) to be sampled and written to `FASTQ` file by *minnow* for each cell. This results in the approximate number of reads shown in the first column of table 4.2. It should be noted that experiments containing more than $\sim 10,000$ cells are often generated from two or more separate experiments, concatenating the resulting count matrices after analyzing the samples in an independent manner. This can be achieved by running multiple instances of *minnow*. The performance is such cases would scale in a linear fashion as expected.

Simulated reads were generated on a server running ubuntu 16.10 with an Intel(R) Xeon(R) CPU (E5-2699 v4 @2.20GHz with 44 cores), 512GB RAM and a 4TB TOSHIBA MG03ACA4 ATA HDD.

4.3 Conclusion

Single-cell sequencing has enabled scientists to gain a better understanding of complex and dynamic biological systems, and single-cell RNA-seq has been one

of the pioneering biotechnologies in the field. Droplet based assays, in particular, have proven very useful because of their high-throughput and ability to assay many cells. Driven by these exciting biotechnology developments, hosts of different methods have been developed in a relatively small time-frame to analyze the gene-by-cell count matrices that result from the initial quantification of these single-cell assay. Previously, various models have been proposed for simulating realistic count matrices [77]. These approaches implicitly assume that, apart from some fundamental limitations due to the biotechnology (e.g. cell capture, molecule sampling from small finite populations, etc.), the problem of ascertaining accurate gene counts from raw sequencing data by aligning reads and deduplicating UMIs is essentially solved. Consequently, the effect of the failures of these quantification pipelines to produce accurate gene expression estimates is not accounted for in the assessment of downstream analysis methods using this simulated data.

In this paper, we introduced *minnow*, which covers an important gap in existing methods for single-cell RNA-seq simulation [84] — the read-level simulation of sequencing data for droplet-based scRNA-seq assays. We demonstrate the use of *minnow* to assess the accuracy of the single cell quantification methods under different configurations of sequence-level characteristics, ranging from adversarial, to realistic, to favorable. We propose a framework for the simulation of synthetic dscRNA-seq data, which simulates the CB, UMI, and read sequences, while accounting for the considerable effects of PCR and realistic sequence ambiguity in generated reads. Further as a flexible framework *minnow* can be easily used to create a variety of possible configurations, such as changing fragment length distribution, sequenc-

ing error, and collision rate etc, to do robust testing of computational tools. We believe *minnow* will help the community develop the next generation of quantification tools for droplet-based scRNA-seq data. It provides the first comprehensive framework to simulate such data at the sequence level, allowing users to validate the accuracy of different methods and providing useful feedback to determine future directions for improving quantification algorithms. *Minnow* is an open-source tool, developed in `C++14`, and is licensed under a BSD license. It is available at `https://github.com/COMBINE-lab/minnow`.

Acknowledgements: We would like to thank Charlotte Soneson for insightful conversations and for providing valuable feedback during the creation of *minnow*.

Funding: This work was supported by NSF award CCF-1750472 and by grant number 2018-182752 from the Chan Zuckerberg Initiative DAF, an advised fund of Silicon Valley Community Foundation. The authors thank Stony Brook Research Computing and Cyberinfrastructure, and the Institute for Advanced Computational Science at Stony Brook University for access to the SeaWulf computing system, which was made possible by NSF grant #1531492.

Chapter 5: Terminus enables the discovery of data-driven, robust transcript groups from RNA-seq data

5.1 Introduction

[1] RNA sequencing has become the de-facto standard for analyzing transcriptomes, and has found myriad applications from differential expression analysis, to the discovery and assembly of rare isoforms. Despite its widespread use, transcriptome analysis via RNA-seq poses a number of computational challenges. For example, reliably mapping or aligning short RNA-seq reads to the reference transcriptome and quantifying the abundance of transcripts is central to most typical RNA-seq analyses. Due to the nature of shared sequences in the reference transcriptome and genome, the simple problem of finding the locus of origin for a particular read sequence can be quite difficult. The complexity results both from alternative splicing, where the isoforms of a single gene can share multiple identical exons, and from very similar sequences arising in families of related genes. These challenges make the read sequence alone insufficient to determine the origin of the sequencing read. In such a scenario, a single read often maps equally well to multiple reference sequences. This ambiguity in determining the exact target sequence prop-

[1] This section is reproduced from Sarkar et al. [96].

agates directly to the process of quantification, where it becomes hard to determine transcript-level expression when there is insufficient evidence to choose some transcripts over others as the true origin of sequencing reads.

To tackle these challenges, there has been tremendous growth in the space of computational tools that can effectively align [20, 92, 97] short RNA-seq reads to transcriptome and tools that can quantify [31, 44, 45, 50, 98] the abundance of transcripts. However, the inherent uncertainty in transcript abundance that results from ambiguous fragment alignment, even after attempting to model this uncertainty in either a maximum likelihood or Bayesian estimation framework, makes it difficult — and in some cases impossible — to provide a single accurate estimate for the number of reads originating from a specific transcript in a given sample. Gibbs sampling is a useful technique for estimating marginal or joint statistics of complex posterior distributions, and has been used by *mmseq* [31], BitSeq [44], RSEM [98] and *salmon* [45] in uncertainty quantification in expression estimation. This provides downstream tools with the ability to analyze the full posterior, instead of just providing a point estimate.

Furthermore, different downstream tools such as mmdiff [99], IsoDE [100], sleuth [101] and swish [102] make use of these estimates in order to estimate differentially expressed transcripts or genes with higher accuracy and robustness. While taking samples from the posterior probability distribution provides some insight about the validity of the point estimates for transcript abundances, and this uncertainty can be propagated for the purposes of differential testing, it is often possible for a particular transcript to not exhibit expression in the point estimate altogether,

in which case it becomes invisible to the analysis tools. Such cases were noted by [99], who further suggested grouping together transcripts whose abundance could not be confidently estimated into transcriptional groups. Specifically, it was demonstrated that there exist numerous cases where the abundance of an individual isoform cannot be reliably estimated, but the abundance of a small group of related isoforms can be determined accurately and robustly.

In a distinct context, but as a result of the same underlying cause of fundamentally multimapping reads, [103] note that these difficulties in mapping can lead to errors in quantification that affect genes of relevance to human disease. Crucially, they highlight that this issue occurs even at the level of genes, and is of concern even if one is not performing a transcript-level analysis. In addition to describing this issue, they identify specific groups of 958 disease-related genes that are affected by this problem, and suggest a gene-level analysis approach whereby groups of genes that share multimapping reads are treated jointly for the purposes of expression estimation and differential analysis. While this approach is quite robust, it is also very conservative, since it precludes transcript-level analysis altogether. Furthermore, depending on the set of all fragments sequenced in a sample, it may still be possible to confidently assess the abundance of a gene, or even a single transcript, even if it shares a large number of mulitmapping reads with other sequenced targets. There may then be utility in directly examining the posterior distributions of abundance estimates to determine when multimapping leads to a high degree of uncertainty in estimating expression, and when, despite the presence of multimapping reads, the abundance of a transcriptional target can be confidently assessed.

The mmcollapse tool [99] exploits the posterior samples generated by mmseq [31] to identify transcripts with highly anti-correlated posterior distributions. Some of the transcripts in these groups would otherwise not be properly estimated (would have estimated abundances of 0), or would have such variable posterior estimates that they could neither be quantified confidently nor meaningfully tested for differential expression. However, when the transcripts are treated as inferential groups among the experimental samples, they can be robustly quantified and the group can be assessed for differential expression.

One of the major caveats of this approach is the particular choice of summary statistics used in order to identify similar groups. Specifically, *mmcollapse* attempts to group transcripts such that the minimum pairwise correlation is not too low. While this is a useful feature to assess, we find that it does not always accord with intuition about which transcripts should be grouped as it does not specifically account for inferential uncertainty that would be reduced by grouping a specific pair of transcripts. Furthermore, the approach taken by *mmcollapse* is both extremely memory intensive (as all posterior samples, for all expressed transcripts, and for all samples in the experiment, must be held in memory simultaneously) and quite time consuming, as the posterior correlations are computed in every iteration of the algorithm, even among completely unrelated transcripts that have little chance of producing promising candidates for grouping. Finally, *mmcollapse* is, arguably, overly-conservative in the constraints is places on transcripts that may be considered as candidates for grouping. For example, only transcripts with no uniquely mapping reads are considered as potential candidates for collapse. However, we observe that

even transcripts with a few uniquely mapping reads may exhibit a large degree of uncertainty in their quantification estimates depending on the total number of reads mapping to such transcripts and the complexity of the patterns of multimapping with related transcripts.

Terminus, the tool presented in the current paper, attempts to address these shortcomings. It takes motivation from *mmcollapse* [99] as well as from the method proposed in "Surface simplification using quadric error metrics" Garland and Heckbert [104], a notable work in the field of computer graphics, in which densely-tessellated shapes are simplified by approximating (coarsening) the mesh that represents the object. In "Surface simplification using quadric error metrics", [104] argue that one way of achieving a visually-appealing approximation is to start with the equivalent network of the visual model and repeatedly contract edges of the network in a manner that leads to minimal visual distortion of the overall shape.

In the same spirit, terminus, reformulates the problem of discovering meaningful inferential groups as a graph simplification problem in which the (sparse) graph that defines what transcripts should be considered as candidates for collapsing is constrained by the read multimapping and conditional probability structure conveyed via the range-factorized equivalence classes [105] produced by *salmon*. This avoids the need to even *consider* the vast majority of possible collapses. Further, terminus uses the reduction in *inferential relative variance* [102] — the reduction in inferential uncertainty that would result by grouping together pairs of transcripts — directly as a metric for optimization. We show that this approach is extremely computationally efficient, and that it leads to groups of transcripts that are both

biologically and inferentially meaningful. In complex transcriptomes (like human or mouse), our approach reduces the memory requirement by over two orders of magnitude compared to mmcollapse, and is simultaneously two orders of magnitude faster. We validate our results in both simulated and experimental datasets, and present time and memory benchmarks for running these tools.

5.2 Methods

Numerous quantification tools, including *mmseq* and *salmon*, encode the structure of mapping ambiguity in the form of equivalence classes. Terminus makes use of a collection of *range-factorized* equivalence classes [105] obtained from *salmon*. The overview of the mathematical model for *salmon* is outlined in supp_fill. Here we would discuss the data structures that are relevant for terminus. Given a set of transcripts $\mathcal{T}$ and a set of read sequences $\mathcal{R}$, a set of equivalence classes $\mathcal{E}$, is defined as a function from the domain of set of transcripts $\mathbf{t} \subseteq \mathcal{T}$ to a natural number denoting the number of reads that are mapped to that group of transcripts, formally $\mathcal{E} : \mathcal{P}(\mathcal{T}) \rightarrow \mathbb{N}^+$, where, $\mathcal{P}(\mathcal{T})$ is a power set of all transcripts. In practice the size of the domain of $\mathcal{E}$ is much smaller than $2^{|\mathcal{T}|} - 1$. Furthermore *salmon* extends the notion of naive equivalence classes by adding the measure of mapping quality. Range-factorized equivalence classes can be defined as, $\mathcal{E} : \mathcal{P}(\mathcal{T} \times \mathbb{N}^+) \rightarrow \mathbb{N}^+$. In effect each equivalence class consists of a set of pairs (t_i, w_i) denoting transcript t_i and a number $w_i \in (0, 1]$ representing the average conditional probability with which the fragments in this equivalence class arose from transcript t_i. Depending on

the granularity with which the range-factorized equivalence classes are defined, the equivalence relation between fragments is determined by the bin, within the range of conditional probabilities, into which they fall with respect to each transcript to which they map. Complete details of how this relation is defined can be found in [105]. Along with these preliminary structures, *salmon* also computes Gibbs chains $\mathcal{G}$, providing samples from the posterior distribution of the model. To be precise, each element $\mathbf{g}_i \in \mathcal{G}$ contains the estimated fragment counts for transcript i, taken over all (possibly thinned) iterations of Gibbs sampling.

Our goal is to use the range-factorized equivalence classes $\mathcal{E}$ and the posterior samples $\mathcal{G}$ to determine groups of transcripts that exhibit high inferential uncertainty, and then to collect them together into robust transcriptional groups for which the posterior uncertainty is considerably lower. To avoid the exponential space of possible groups, we use the structure over transcripts induced by $\mathcal{E}$ to guide our search, and further restrict each iteration of the algorithm to perform a single pairwise collapse.

First, we collapse transcripts that appear in exactly the same set of equivalence class labels and that have near-identical conditional probability vectors. These are transcripts for which, even without examining the posterior samples, it is clear that no inference algorithm will have sufficient information to tell apart. Specifically, we accomplish this collapse using a partition refinement algorithm [106] where all transcripts start out within a single partition P_0. We then iterate over $\mathcal{E}$ and determine the partitions that should be induced with respect to the current equivalence class $e \in \mathcal{E}$. This is simply the subsets of transcripts that have nearly-identical

conditional probabilities with respect to e. For each such subset r in e, we refine the current partitioning P_i into P_{i+1} by replacing each set S_j in P_i that contain elements from r with $S_j \cup r$ and $S_j \setminus r$. This process is performed iteratively until we have processed all of the equivalence classes in $\mathcal{E}$.

Next, we define a graph $F = (V, E)$, constructed over $\mathcal{E}$ designed to encode the likely candidate transcripts for grouping. We define V to be the set of transcripts quantified, where $\{v_i, v_j\}$ is an edge in E if the transcript v_i and v_j co-occur in some equivalence class, and either of indication function $h(v_i)$ or $h(v_j)$ is true, and $s(v_i, v_j) \leq \tau$. Here $h(\cdot)$ is an indicator function that determines if a transcript is, when considered in isolation, a good candidate for possible grouping. We define $h(\cdot)$ as,

$$h(v_i) = \begin{cases} 1, & \text{if } mean(\mathbf{g_i}) \geq 1 \text{ and } \frac{max(\mathbf{g_i}) - min(\mathbf{g_i})}{mean(\mathbf{g_i})} > \lambda \\ 0, & \text{otherwise.}, \end{cases} \tag{5.1}$$

where λ is a user-defined parameter set to 0.1 by default. The score function $s(v_i, v_j) \leq \tau$ is designed to measure the improvement (decrease) in inferential uncertainty obtained by grouping v_i and v_j together. We define

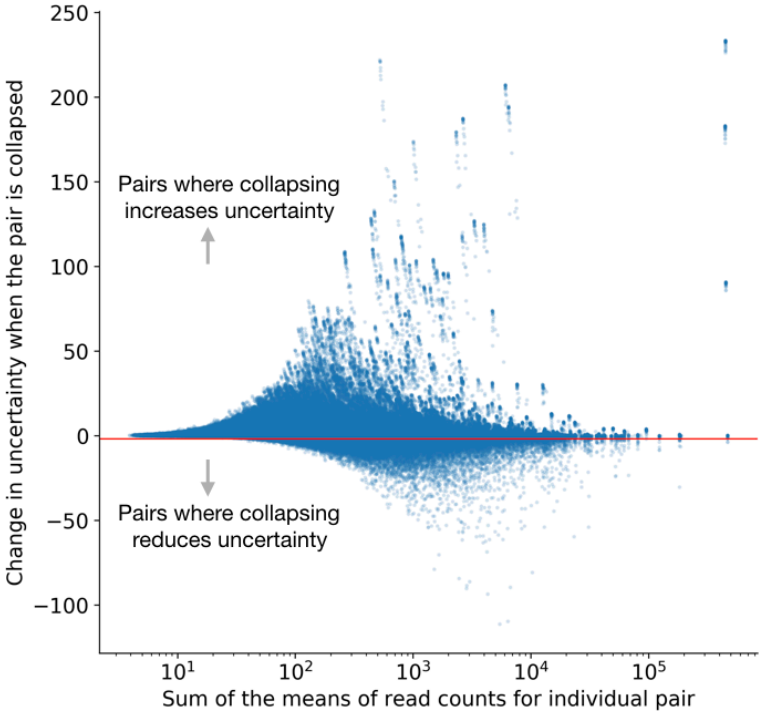
$$s(v_i, v_j) = \text{infRV}(\mathbf{g_i} + \mathbf{g_j}) - \frac{\text{infRV}(\mathbf{g_i}) + \text{infRV}(\mathbf{g_j})}{2} \tag{5.2}$$

where,

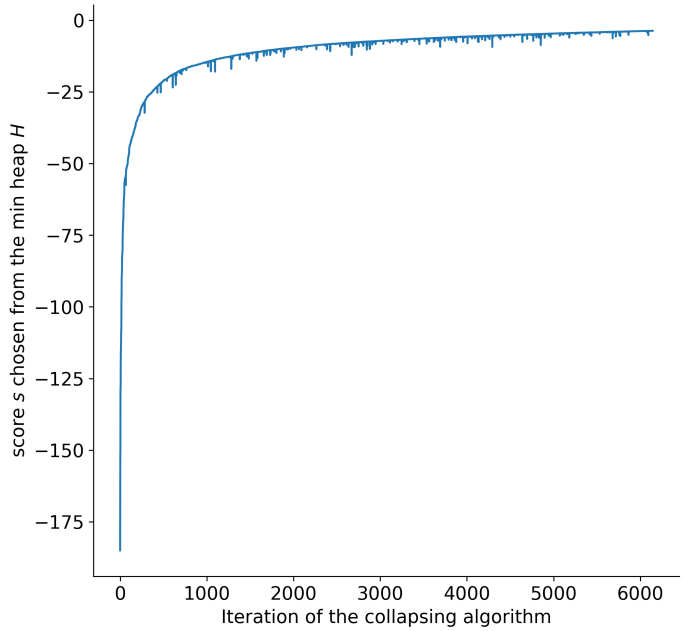
$$\text{infRV}(\mathbf{g_i}) = \frac{\max(0, var(\mathbf{g_i}) - mean(\mathbf{g_i}))}{mean(\mathbf{g_i}) + p} + s \tag{5.3}$$

Where p is a pseudocount (we use 5 in terminus) and s is a small global shift (we use 0.01 in terminus). The infRV$(\cdot)$ function measures the inferential relative variance, and the definition is taken from [102]. The motivation for dividing by the mean is to stabilize the quantification of uncertainty across transcripts with low or high expression. Here, a negative value of $s(\cdot, \cdot)$ indicates that, when we sum the posterior samples of the two transcriptional units, the inferential relative variance is less than the average of the inferential relative variance of the individual units. When the value of $s(\cdot, \cdot)$ is sufficiently low (see fig. 5.1(a)), then grouping these units together results in a substantial reduction in uncertainty by treating the pair of transcriptional units as a single group. A specific example of such collapse for two transcripts from human is shown in fig. 5.2. In practice, we set the threshold for grouping (τ) in a data-driven manner by constructing a "background" distribution over the values resulting from evaluating $s(\cdot, \cdot)$ on a large number of randomly-selected, expressed transcripts. We note that almost all pairs are not good candidates for collapsing, and so these random draws allow us to approximate sampling from the null distribution of scores for transcripts that should not be grouped. We call this background distribution $\mathcal{B}$.

Given a set of transcripts $\mathcal{T}$, the possible number of transcript pairs to consider for collapse can be $O(|\mathcal{T}|^2)$. For an organism with a large number of annotated transcripts (such as human or mouse) it is not computationally feasible to enumerate over such a large distribution. Therefore, for choosing a desirable threshold τ, we use an iterative sampling approach described in algorithm 5.2. Specifically, we



(a) The reduction in uncertainty, defined by the score metric in eq. (5.2) vs. the sum of the mean read counts for the individual transcripts of the pair. The cut off is chosen in a data-driven manner.



(b) The convergence of minimum *InfRV* in consecutive iterations.

Figure 5.1: The inferential gain vs the mean of the candidate pairs and the convergence of terminus algorithm.

exponentially increase the number of samples we draw until the resulting threshold τ that we would infer changes by less than some small quantity (we use 0.1%). The convergence of algorithm 5.2 (steps plotted in Supplementary fig. S1) ensures that the final threshold captures a close enough approximation to the true desired value. We set the threshold as a quantile of the background distribution, and choose 2.5% by default. The choice of the percentile is empirical, and chosen so that very few "independent" transcripts might be mistakenly grouped together. This parameter can be used to control sparsity of the graph.

Figure 5.1(a) plots the relation between the scoring function $s(i, j)$ defined in eq. (5.2) and sum of the means of Gibbs samples $\mathbf{g}_i$ and $\mathbf{g}_j$, corresponding to individual transcripts i and j. The red line shows where the empirical cutoff falls (which changes from one experiment to another). The points below the red line are candidates for grouping.

Given, a graph F, a set gibbs samples $\mathcal{G}$, and a threshold τ, terminus follows an iterative algorithm for collapsing transcripts. By construction of F, an edge $\{v_i, v_j\} \in E$ has three different attributes, (i). the score $s(\{v_i, v_j\})$, (ii). a set of equivalence classes $eqlist(\{v_i, v_j\})$ where v_i and v_j co-occur and (iii). the total number of reads $c(\{v_i, v_j\})$ that are shared between transcripts corresponding to nodes v_i and v_j. Terminus starts off by constructing a min-heap H [107] over the set of edges E where the key for each edge is by the score function evaluated on the vertices sharing this edge. Terminus then iterates over H until it becomes empty, at each step collapsing the edge that was popped from the heap.

In each iteration t, starting with the current state of graph and the Gibbs

Data: $\mathcal{T}, \mathcal{G}$, selection_size
Result: τ_{new}
$\tau_{old} \leftarrow 0$;
$\tau_{new} \leftarrow 0$;
converged $\leftarrow$ False ;
$\mathcal{B} \leftarrow \emptyset$;
while *not* converged **do**
  $S \leftarrow \{(i,j)|i,j \in \mathcal{T}\}, s.t.|S| =$ selection_size ;
  for $(i,j) \in S$ **do**
    $\mathcal{B} = \mathcal{B} \cup s(i,j)$;
  end
  $\tau_{new} \leftarrow$ percentile$(\mathcal{B}, 2.5)$;
  diff $\leftarrow |\tau_{old} - \tau_{new}|$;
  if diff $< \delta$ **then**
    converged $\leftarrow$ True ;
  end
  selection_size $\leftarrow$ selection_size $\cdot 2$;
  $\tau_{old} \leftarrow \tau_{new}$;
  $\mathcal{B} \leftarrow \emptyset$;
end
return τ_{new} ;

Algorithm 2: Threshold τ selection algorithm

samples, $(F_t, \mathcal{G}_t)$, terminus pops an edge $\{v_i, v_j\}$ from the heap with the minimum score and *collapse* the corresponding end points of the edge and produces a state $(F_{t+1}, \mathcal{G}_{t+1})$. The actual collapse process has a number of cases, but in essence, after collapsing, nodes v_i and v_j in the graph F_t becomes a single node in F_{t+1}. Simultaneously, the corresponding vectors $\mathbf{g}_i$ and $\mathbf{g}_j$ from $\mathcal{G}_t$ are added to obtain $\mathbf{g}_i$ (to make collapsing more efficient, we associate the collapsed pair with some node i from the original endpoints) in $\mathcal{G}_{t+1}$.

The collapsing algorithm involves updating the information of the existing edges, and pushing appropriate edges on the heap. At any iteration, given a graph F_t and the edge $\{v_i, v_j\}$ that has to be collapsed, terminus deletes the edge $\{v_i, v_j\}$ and includes either of the two nodes, say v_i (without loss of generality, in practice

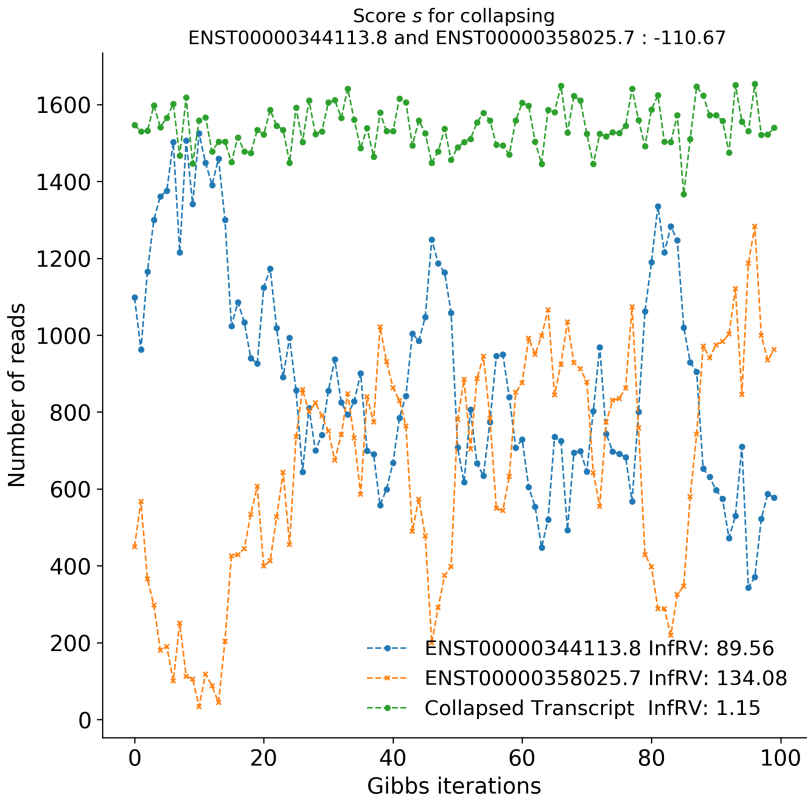


Figure 5.2: Demonstration of uncertainty reduction by collapsing. Individual transcripts ENST00000344113.8 and ENST00000358025.7 are collapsed, and the corresponding *InfRV* is also reduced.

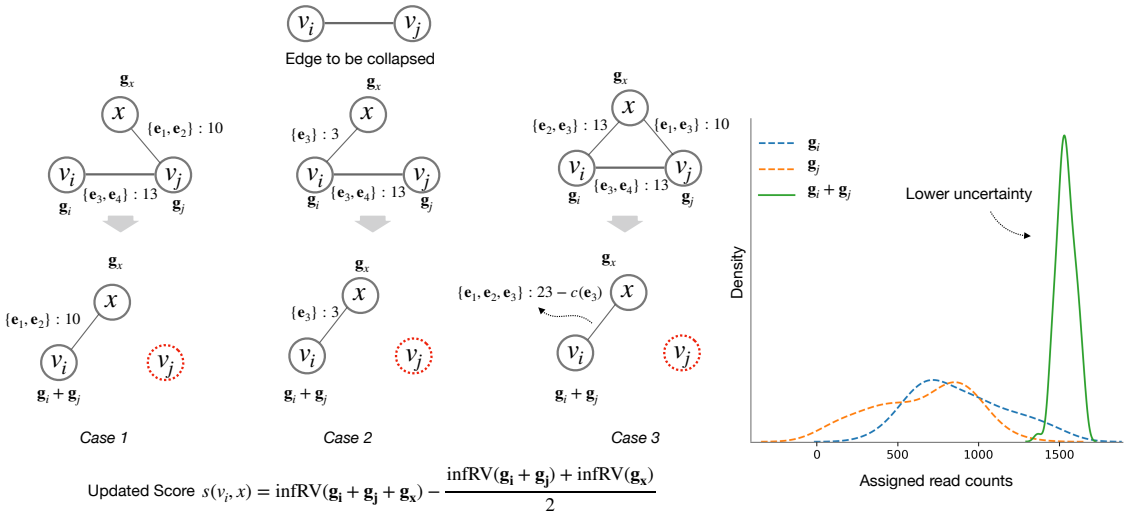


Figure 5.3: Different possible scenarios that can arise while collapsing a node in Graph F. In case 1 and 2, as the edge property either remains same or transferred to the new edge. In case 3, the construction of edge explained in 5.2 handles the problem of over counting already counted equivalence class (subtracting count of equivalence class e_3,$c(e_3)$). The right most plot shows the distribution of the individual gibbs samples in dotted line along with the gibbs samples of the collapsed group with much lower uncertainty.

keeping the smaller numeric index, i.e. $i < j$) in F_{t+1}. Terminus updates $\mathcal{G}_{t+1}$ as $\mathbf{g}_i \leftarrow \mathbf{g}_i + \mathbf{g}_j$.

The edge set of F_{t+1} is determined as follows; given the set of adjacent nodes of v_i and v_j in F_t, denoted as $adj_t(v_i)$ and $adj_t(v_j)$, any affected node $x \in adj(v_i) \cup adj(v_j)$ is handled in one of the 3 cases below:

Case 1: Given $x \in adj(v_i)$ and $x \notin adj(v_j)$, an edge $\{x, v_i\}$ is added to F_{t+1}, terminus recalculates $s(x, v_i)$. If $s(x, v_i)$ is smaller than τ, then the edge with the updated $s(x, v_i)$ is pushed into the min heap H. The eqlist( ) and corresponding counts for the pair remain unchanged as this edge already existed in F_t.

Case 2: Given $x \notin adj(v_i)$ and $x \in adj(v_j)$, a new edge $\{x, v_i\}$ is added to F_{t+1},

terminus calculates $s(x, v_i)$. If $s(x, v_i)$ is smaller than τ, then the edge with the updated $s(x, v_i)$ is pushed into the min heap H. The eqlist( ) and corresponding counts for $\{x, v_j\}$ are copied over to edge (x, v_i), since in the new stated v_i and v_j are deemed to be identical.

Case 3: Given $x \in adj(v_i)$ and $x \in adj(v_j)$, then x, v_i and v_j forms a triangle in F_t. From all three edges only the edge $\{x, v_i\}$ is added to F_{t+1}. Terminus calculates $s(x, v_i)$, if $s(x, v_i)$ is smaller than τ, then the edge with the updated $s(x, v_i)$ is pushed into the min heap H. The equivalence class list for newly added edge $\{x, v_i\}$ is recalculated as: $eqlist(\{x, v_i\}) = eqlist(\{x, v_i\}) \cup eqlist(\{x, v_j\})$. The union of the equivalence class ids saves the edge from over counting those equivalence classes that are shared between all three transcripts x, v_i and v_j. In the same fashion, the reads that are shared between x and v_j are transferred to the edge $\{x, v_i\}$. The shared read counts are calculated as follows

$$c(\{x, v_i\}) = c(\{x, v_i\}) + c(\{x, v_j\}) - \sum_{\mathbf{e} \in \cap_{i,j} x} c(\mathbf{e})$$

Here $\cap_{i,j} x$ signifies a set of shared equivalence classes.

All other nodes and edges remain unchanged in the iteration and are simply "copied over" to F_{t+1}. Similarly, all elements of $\mathcal{G}_t$ would be copied over to $\mathcal{G}_{t+1}$ except the changed element $\mathbf{g}_i$ and removal of the element $\mathbf{g}_j$. We observe that the cardinality of F and $\mathcal{G}$ monotonically decreases, ensuring convergence.

Note that throughout the algorithm, the heap maintains the property that

any edge currently present in the heap has a score less than empirical threshold τ. However, when one endpoint of an edge is modified, the corresponding score of the edge is not directly modified in the heap; such an edge is considered to be "stale." In order to keep the process of grouping efficient, terminus reuses the same graph in all iterations, keeping state information (i.e. a "last modified" timestamp) encoded as an attribute of the edge. When an edge is updated terminus just updates the corresponding attribute. When an edge is popped from the heap, we first check to ensure that it is not stale before processing the collapse. If the edge is stale, then we recalculate its score and either perform the collapse or re-insert the edge in the heap.

Most typical RNA-seq experiments are comprised of multiple replicates. The process described above, however, is carried out individually per-sample. Terminus takes a two-step approach in order to find coherent groups across the multiple replicates that comprise an experiment. It first groups the individual samples separately, and writes the groups for each sample. This step can be trivially-parallelized. After obtaining individual groups, terminus follows a consensus algorithm in order to find a set of groups to use across all samples.

The consensus procedure starts with individual groups, and constructs a union graph by treating each of the groups as a complete connected undirected graph (clique). For example: given two different groups $\{v_i, v_j, v_k\}$ from one sample and $\{v_i, v_j\}$ from another, terminus first constructs a weighted triangle with end points v_i, v_j and v_k where each edge has a weight 1. While considering the second group, terminus increases the weight of edge $\{v_i, v_j\}$ by 1. This iterative procedure gener-

ates a weighted graph ensuring any edge in the graph represents two transcripts that belong to the same group in at least one sample. Terminus further prunes the union graph and removes the edges that have a weight below a user-defined threshold. The consensus mechanism ensures that a pair of transcripts should at least co-occur in a specified number of samples to qualify for the final grouping. Subsequently the final group that is common for all the samples is extracted by writing the connected components of the pruned graph. Note that these groups are deemed to be universal across samples and replicates. The default consensus threshold is $\frac{1}{4} \cdot m$ where m is the number of samples in the experiment.

5.2.1 Datasets and Evaluation

Different datasets, including both simulated and experimental data and ranging across different organisms were used to demonstrate the benefit of using collapsed groups and the ability of terminus to determine meaningful and robust groups.

Simulated dataset on Human: We used polyester [40]-generated simulated RNA-seq data, curated by [108], to assess accurate estimation when the true expression is known. The actual experiment was derived from a joint distribution of mean and dispersion values from GEUVADIS samples [109]. For the current experiment, we have chosen a 4 vs. 4 subset from the original set of 12 vs. 12 samples. The experiments consist of paired end 100 base pair reads (see [108] for more detail). We refer to this dataset as *simulated* 4 *vs.* 4 *human* dataset.

Simulated dataset with diploid transcriptome from Mouse: The presence of a

fully-diploid transcriptome exacerbates the challenges caused by transcript sequence similarity. The mapping uncertainty in such case can greatly impair the quantification estimates. To capture such an extreme scenario, we have produced a diploid transcriptome following the same pipeline described by [110]. The diploid transcriptome (named as the N×P transcriptome) combines a cross between NOD/ShiLtK (NOD) and PWK/PhJ (PWK) strains of mice. The final transcriptome is obtained by running `prepare-rsem-reference` on the hybrid `gtf` and reference genome file. The hybrid `gtf` is produced by running `g2gtools` and `emase` [110]. The full script for producing such a transcriptome is provided in the repository. The paired-end read files from N×P transcriptome are generated using polyester with the true counts obtained from running *salmon* on a real mouse RNA-Seq experiment (accession number `SRR207106`). We refer to this dataset as *simulated allelic mouse* dataset.

Experimental RNA-seq samples from [111]: The experiment widely known as *pasilla* [111] is an ensemble of 6 vs 6 RNA-seq experiment (with NCMI GEO accession numbers GSM461176 to GSM461181) that studies the effect of RNAi knockdown of Pasilla, which is the ortholog in Drosophila melanogaster of NOVA1 and NOVA2 mammallian genes. The same experiment is also used by [99] to demonstrate the effect of grouping.

Given the above mixture of simulated and real experiments, we have run two sets of tools to produce groups, (i) terminus on the output of *salmon* and (ii) *mmcollapse*. There are different ways to produce input for *mmcollapse*. The *mmcollapse* run is preceded by a run of *mmseq*, which takes BAM files as input. To make the comparison of *mmcollapse* and terminus as consistent as possible, we have used

salmon-produced BAM files for running the *mmcollapse* pipeline. For all experiments, *salmon* is run with `--numGibbsSamples 100` option in order to generate Gibbs samples. We used the `--hardFilter` parameter for producing the BAM files. In case of hits with different mapping scores, `--hardFilter` keeps only the hits with the best score. This parameter is chosen carefully to follow the equivalent `Bowtie` [11] parameters mentioned in Turro et al. [99] and Turro et al. [31].

Evaluation of the quality of the collapsed groups is inherently a difficult task. Since there can be many possible groupings given a transcriptome dataset, comparing one grouping verses another requires biological validation. For simulated datasets, we validated the results by using the Spearman correlation and mean absolute relative difference (MARD) between the grouped estimates and corresponding true abundances. To be precise a given a grouping P is defined as a partition over the set of all transcripts, allowing singleton partitions, denoting the un-grouped transcripts form their own groups. While assessing a particular grouping P, we induce the same partitioning over the ground truth abundances. Given a set of true read counts $\rho_{true} = \{\rho_1, \ldots, \rho_N\}$ for N transcripts and the estimated counts $\rho_{est} = \{\rho'_1, \ldots, \rho'_M\}$ where M is the number of groups (including singleton groups), we define a partition P induced on ρ_{true} as,

$$\rho^P_{true} = \{ \sum_{t_k \in p_i, \rho_{t_k} \in \rho_{true}} \rho_{t_k}, \forall p_i \in P\}$$

We use ρ^P_{true} and ρ_{est} for each of the two tools evaluated in the current manuscript.

5.3 Results

5.3.1 Quantification comparison on simulated data

The two different simulated datasets that are used to demonstrate the utility of the grouping algorithm implemented in terminus pose challenges of distinct natures. While the simulated 4 vs. 4 human dataset is designed to capture aspects of real-world human transcript expression, the simulated allelic dataset from mouse represents the tremendous sequence ambiguity imposed by a diploid transcriptome and its resulting effect on quantification. We demonstrated that, on both the datasets, terminus improves the accuracy of quantification results over *salmon* and *mmcollapse* under diverse metrics.

The simulated 4 vs. 4 human dataset contains realistic GC bias estimated by alpine [112] from experimental samples from the GEUVADIS study [109]. If these realistic biases are not properly modeled, accurate quantification at the transcript level is impaired. By virtue of the design matrix this experiment simulates differentially expressed transcripts and the nontrivial variability between the samples can also pose challenging problems to the collapsing algorithm. The global Spearman correlation and MARD values presented in Table 5.1 summarizes those metrics across 8 different samples and takes the average. While *salmon* itself performs fairly well in this dataset, we observe the groups induced by terminus further improves the accuracy of abundance estimates. The poor performance of *mmcollapse* is due to very noisy low abundance values given to truly unexpressed transcripts. The perfor-

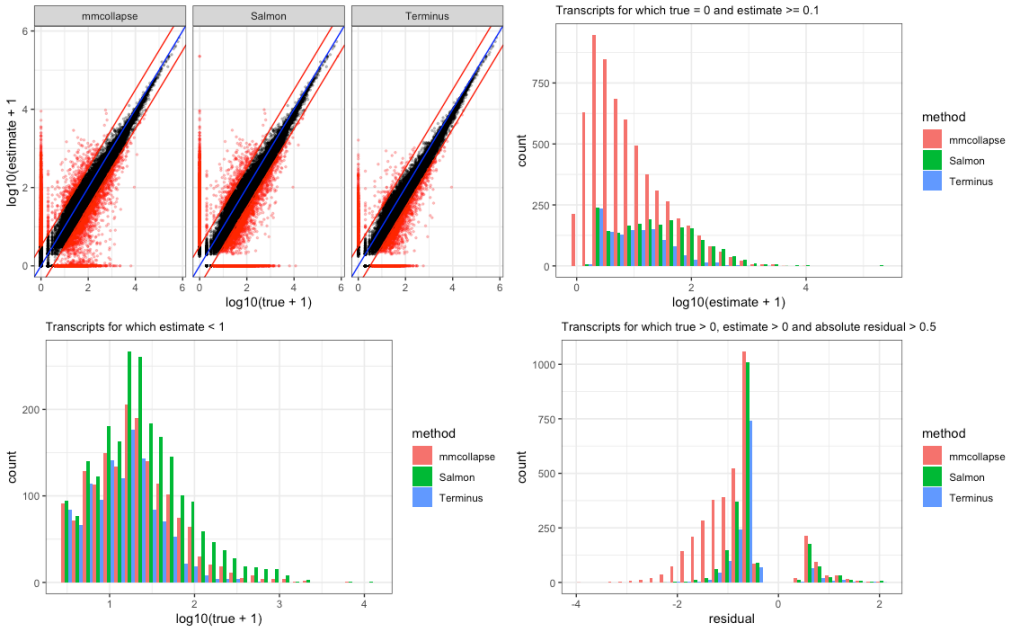


Figure 5.4: A comparative view of three different tools on the simulated 4 vs. 4 human dataset. The scatter plot on the top right panel asseses the performance of estimates vs truth for three tools *mmcollapse*, *salmon* and terminus. The histogram at the top right corner shows the frequency of mis-estimated transcripts which are not expressed in the ground truth. Here only those transcripts for which the tools have at least assigned one read are considered. This helps to get rid of the noise introduced by *mmcollapse* that leads to degenerate values. The histogram at the bottom right shows the frequency of lowly expressed transcripts. The rest of the transcripts (that are expressed) are shown in bottom right plot. Here the x-axis is the difference between the log transformed values of true and estimated expression. Negative residual values signify underestimation while the positive values signify overestimation.

Datasets	Correlation (Spearman)			MARD		
	salmon	*mmcollapse*	terminus	*salmon*	*mmcollapse*	terminus
Human	0.94	0.78	0.96	0.11	0.15	0.09
Mouse	0.91	0.81	0.95	0.12	0.12	0.07

Table 5.1: Spearman correlation and MARD (mean absolute relative difference) are calculated with respect to ground truth under for both the simulated 4 vs. 4 human dataset (termed as Human), and the simulated allelic dataset from mouse (termed as Mouse)

mance of *mmcollapse* improves substantially when only considering truly expressed transcripts.

Figure 5.4 compares the quantification results at a more granular level under different constraints on the true and estimated counts. Among transcripts that are truly expressed, *mmcollapse* more often mis-assigns reads compared to *salmon* or terminus. The skewness of histogram from *mmcollapse* suggests that it tends to underestimate the true counts of transcripts. This effect is also visible in the corresponding scatter plot at the top left corner. The spread of the red points signify deviation from the true counts. The scatter plot from terminus *shrinks* these mis-estimates towards the diagonal by putting them into groups improving the overall correlation.

A similar plot for simulated allelic dataset from mouse is shown in fig. 5.5. Owing to the diploid transcriptome, we see considerable ambiguity in the underlying sequence. It reduces the overall correlation for all the tools and further affects the convergence of the *mmcollapse* collapsing algorithm. Terminus consistently produces estimated counts closer to truth. The corresponding scatter plot capture the shrinkage of the transcripts groups toward the diagonal which are otherwise

mis-estimated by *salmon*.

Allelic imbalance: Due to presence of highly similar pairs in a diploid transcriptome, it is often a challenging task for a quantification tool to assign reads to the correct allele, especially when both of the alleles are equally likely in terms of the confidence of the alignment. The proportion of two different alleles present in an experiment is often termed as the *allelic imbalance*. When the allelic imbalance is close to 0.5 — when both alleles are expressed equally — accurate estimation becomes particularly difficult. The reason being the fact that there is almost equal prior for assigning a read to either of the candidates. In such cases, the maximum likelihood estimators often prefer one allele over the another, mis-estimating the resulting abundances. We pinpointed such cases by investigating cases where the true allelic imbalance is restricted to an interval of 0.45 to 0.55, focusing on the region where the uncertainly among the alleles is highest.

Figure 5.6 plots the allelic imbalance predicted by *salmon* (in y-axis ratio of the expression values for individual alleles estimated by *salmon*) vs. the true allelic imbalance (ratio of true expression of the alleles of a transcript). The color of the point is determined by the fact if the transcript is grouped by terminus (here blue when grouped and orange when not). To get a closer look at the mis-estimation problem in these cases fig. 5.6 zooms the x-axis to the range 0.45 to 0.55. We observe for transcripts with true allelic imbalance 0.5 *salmon* either over-estimates or under-estimates the true allelic imbalance, resulting in the spread through the the entire y-axis. Meanwhile, the cluster of the points near the very end of $x = 0.5$ vertical line suggests that in those case all the reads are assigned to one of the

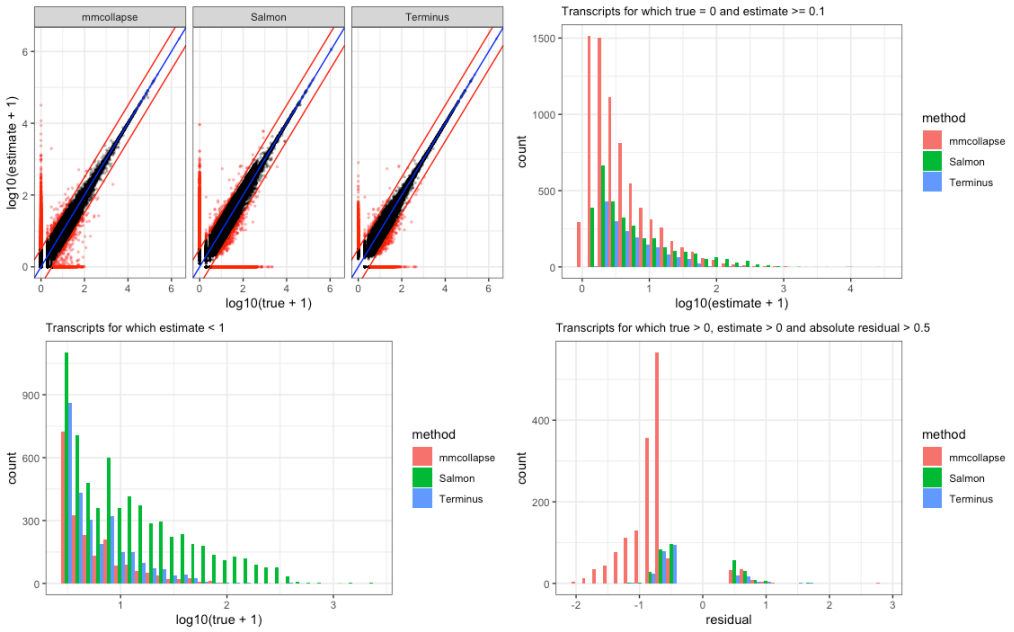


Figure 5.5: Accuracy of *salmon* with and without grouping and *mmcollapse* on the simulated allelic dataset is shown in the plot. The metrics are similar to that of fig. 5.4

alleles leading to one allelic imbalance estimates of 0 and 1. As the color (signifying grouped / non-grouped status) suggests, in those cases, terminus correctly identifies the present uncertainty and groups the alleles together.

To quantify the benefit of grouping in the context of allelic imbalance, we have considered the cases of mis-estimated dominant alleles. Specifically, such cases occur when the allelic imbalance ratio is reversed and the allele that truly has higher expression is estimated to have lower expression and vice-versa. We measured the proportion (r_{sal}) of such cases versus the total number expressed alleles (counting an allele pair only once). We also measured the same metric after grouping. That is, we measured the ratio (r_{term}) where the numerator is the number of number of mis-estimated dominant alleles that are not grouped, and the denominator is the number of expressed alleles where neither allele from the pairs was grouped. For

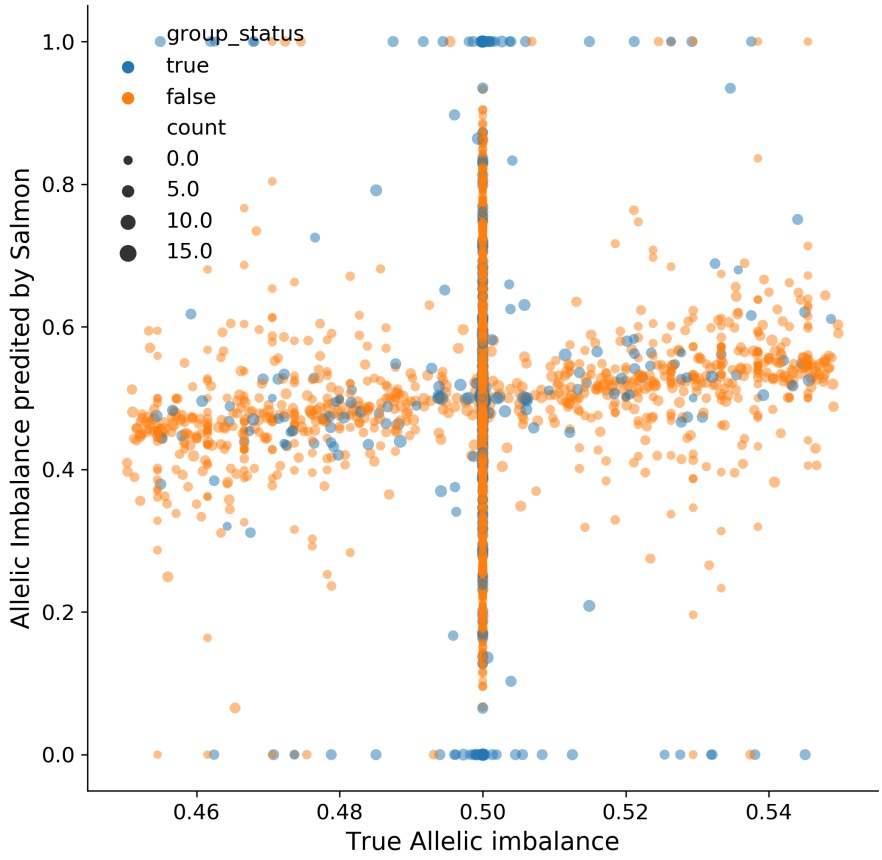


Figure 5.6: The x-axis shows the true allelic imbalance and the y-axis shows the allelic imbalance predicted by the quantification values estimated by *salmon*

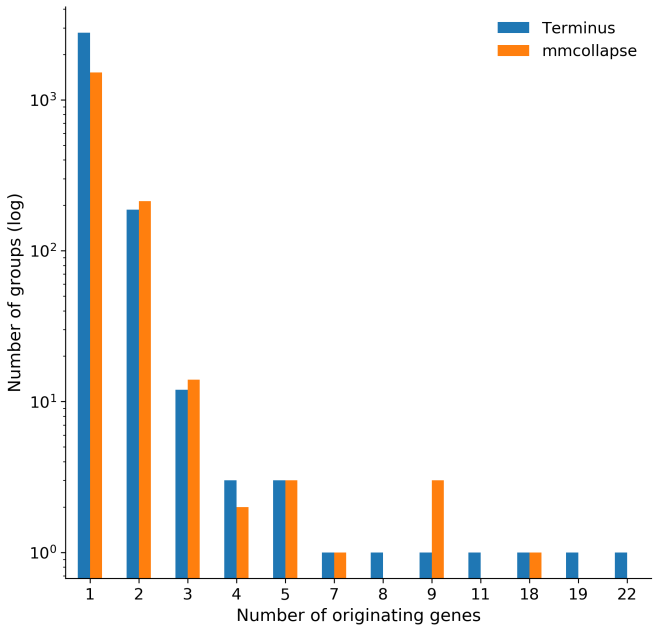
the simulated allelic dataset we found r_{sal} to be 0.1 and r_{term} to be 0.04, signifying the fact that terminus is capable of preferentially grouping together alleles whose allelic ratios are highly uncertain, and where the estimates of *salmon* are otherwise most-likely to be incorrect.

5.3.2 Quantification on real datasets

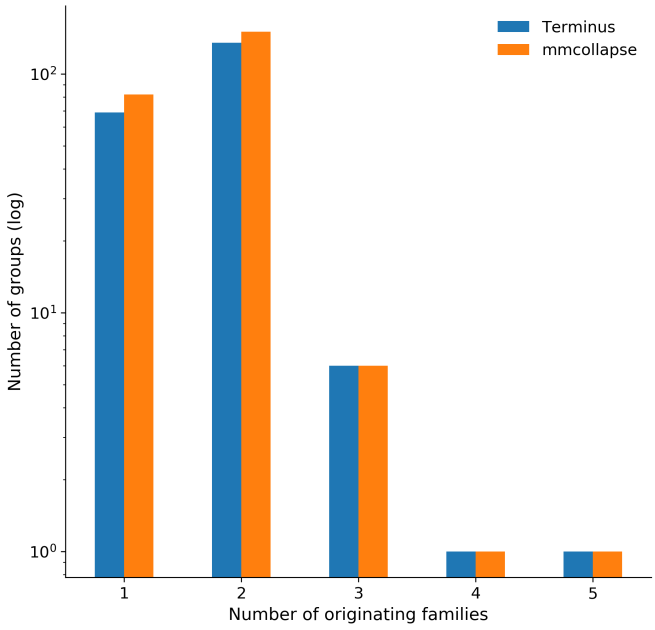
The *Pasilla* dataset is an experimental RNA-seq experiment in D. melanogaster. It comprises of both single-end and paired-end RNA-seq reads. Owing to rapid alternative splicing (AS), prevalent in Drosophila, the organism becomes specifically interesting in the context of evaluating uncertainty induced by extensive AS. Almost 20-37% multi-exon genes are alternatively spliced [113]. The isoforms within a gene that share one or multiple long exons are often very hard to distinguish, resulting in low-confidence estimates. As the dataset does not contain the true quantification values, we measured other biological attributes to validate the grouping.

From the $30,597$ transcripts, terminus and *mmcollapse* grouped 7904 and 4388 transcripts respectively, distributed in 3025 and 1835 groups. As expected, most of the transcripts in the group originate from the same gene. When groups contain transcripts from multiple genes, these genes belong to either the same gene family or to a closely-related gene family.

Figure 5.7 depicts this phenomenon through a histogram plot. In fig. 5.7(a), the x-axis is the number of genes to which all the transcript within one group can be mapped. For example, for groups that belong to the bar corresponding to



(a) The x-axis shows the number of genes that the transcript of the groups can be mapped back to. y-axis shows number of such groups



(b) The x-axis shows the number of gene families that the transcripts can be mapped back to, y-axis is the frequency of such mappings.

Figure 5.7: Biological significance of the groups produced by Terminus and *mmcollapse*.

Datasets	Memory(MB)		Time (h:m:s)	
	mmcollapse	Terminus	mmcollapse	Terminus
Human	370,675	2,841	3:46:37	2:04
Mouse	225,457	485	38:55:44	0.12
Drosophila	4,104	310	8:10	0:10

Table 5.2: The table shows construction time and memory requirements for *mmcollapse* and terminus

$x = 1$, all transcripts come from a single gene; meaning that all the transcripts are isoforms of each other. Similarly, fig. 5.7(b) shows another level of summarization, where the transcripts of a group are mapped back to their gene families. We see in most of the cases that the groups can be mapped back to a small number of gene families. This behavior is expected, as gene families share considerable sequence information and are likely to give rise to related and uncertain expression estimates. Note that *terminus* is agnostic to the annotation of the underlying organism, yet the data driven partitions inherently identify the genes and families annotated in the underlying reference.

We further observe the group sizes (defined by number of transcripts in a group) generated by terminus (largest one being 54) tend to be larger from that of *mmcollapse* (largest one being 18). Often these large groups also identify gene families that share large numbers of exons. One such example from the groups formed by terminus, consisting 54 transcripts is from the *para* gene and from the same family of parathyroid hormone-related protein. This group of proteins are included in many other groups. Another such group comes from gene *slo* or slowpoke, that regulates the release of a neurotransmitter.

5.3.3 Computational Performance

Table 5.2 shows the computational performance of terminus vs *mmcollapse*. terminus takes considerably less time to compute the groups and requires much lower memory to run. This enhanced performance derives from two key attributes of terminus. First, terminus does not consider collapsing transcript pairs that do not appear in any equivalence class. This results in the evaluation of many fewer pairs. Second, terminus uses the underlying graph structure induced by the equivalence classes to order and prioritize the collapses, avoiding the need to constantly recompute candidate pairs. We observe that the performance of *mmcollapse*, both in terms of running time and required memory, greatly varies from one sample to another. One possible reason for such behavior could be the variation in the number of non-unique transcripts. While, in the case of the simulated allelic dataset, *mmcollapse* ran for a long time (we terminated the run after 24 hours), for *Pasilla* dataset it finishes relatively quickly. However, we observe that the memory requirement of the tool often made it very difficult to test it with multiple threads. In those cases, the *mmcollapse* run had to be restricted to 1 thread (e.g. on the simulated allelic dataset, running *mmcollapse* with even 1 thread required $\sim$ 214G of RAM). The enormous speed benefits of terminus suggest that it can be easily incorporated as a part of standard lightweight RNA-seq workflow for finding out groups, with very little computational overhead. We note that we have not included the time and memory requirement for *salmon* and *mmseq* as *mmcollapse* is the tool compared with terminus.

Chapter 6: Discussion and future directions

6.1 Introduction

The tools and methods that I described in this thesis range from understanding the estimation uncertainty in RNA-seq quantification to designing a single-cell RNA-seq sequence simulator. Although the body of work spans from theoretical exploration to purely implementation-heavy applied projects, most of them share a common sub-structure such as the equivalence class. Starting with this compressed representation of the RNA-seq mapping paradigm, I ventured into different application areas such as compression of high throughput short-read sequences, accelerating alignments, and improving the de-novo assembly by clustering contigs etc. On the other hand, I used equivalence classes together with the posterior Gibbs samples to delve deeper into the RNA-seq quantification process and extracted information about the uncertainty of the underlying transcripts. In conclusion, I will describe different cases where the data-driven grouping of the transcripts markedly improves quantification's performance by reducing the uncertainty. Following that, I will describe a few challenges in RNA-seq quantification that a more inclusive model can address. I will conclude with a plan to extend *minnow* to simulate tagged end single-cell RNA-seq reads that can capture the cellular dynamics.

Datasets	Number of transcripts grouped		Number of groups	
	terminus	*mmcollapse*	terminus	*mmcollapse*
Human	12623	1454	4972	640
Mouse	37241	53325	17554	24831
D.Mel	4863	4388	2040	1835

Table 6.1: Group statistics for simulated 4 vs. 4 human dataset (termed as Human), and the simulated allelic dataset from mouse (termed as Mouse) and *Pasilla* dataset (termed as D.Mel)

6.2 Exploring the data-driven groups from terminus

6.2.1 Terminus-produced groups and the effect on posterior variance

Table 6.1 shows the exact numbers of transcripts that are collapsed and the number of groups that are formed during the collapsing procedure. We observe that for the simulated 4 vs. 4 human dataset dataset, the number of transcripts that are collapsed is considerably higher in terminus compared to *mmcollapse*. The reason for such difference in the number of groups can be caused by the specific grouping algorithm that *mmcollapse* follows, and the fact that terminus does not *a priori* exclude transcripts from grouping simply because they have some uniquely-mapping fragments. On the simulated allelic dataset dataset, however, the number of groups produced by *mmcollapse* is much higher than that of terminus. On the *Pasilla* dataset dataset, we observe a comparable number of groups formed by both methods.

To further investigate the effect of grouping in one of the samples from simulated 4 vs. 4 human dataset, we have considered all groups with cardinality 2, and measured the change in variance after merging the Gibbs samples from individual

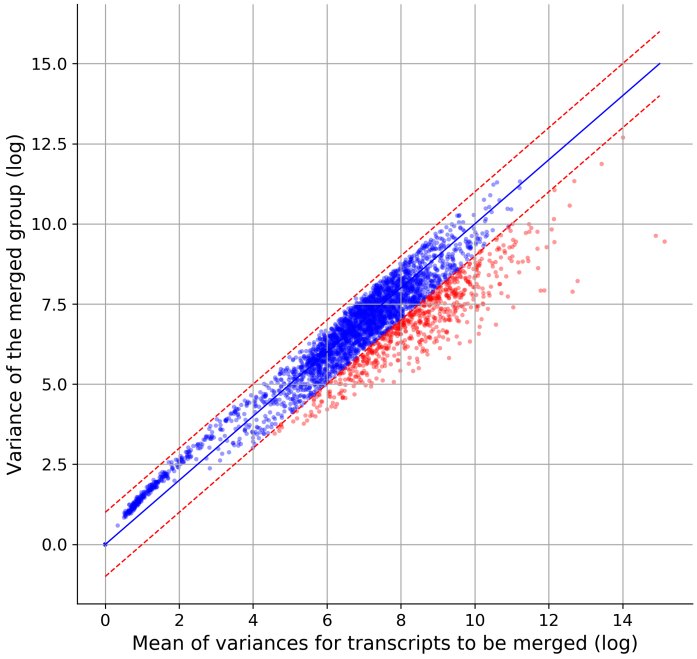


Figure 6.3: The change in posterior variance after merging transcripts compared to the mean variance of the individual transcripts.

candidates. In fig. 6.3 the variance for the merged groups are plotted with respect to the mean of the variances from the Gibbs samples of individual transcripts. We observe that out of 3553 two member groups, 2094 pairs the merged variance is decreased and 1444 cases there is an increase. We further observed while the increase in variance never crosses the difference of 1, the decrease for highly variant values are well beyond that (marked with red color). Noting the *log* scale this positively shows that the terminus produced groups bound the change in variance after the collapse.

In order to highlight the improvement obtained by terminus, we plotted the comparative metrics of *salmon* and terminus in fig. 6.4 and fig. 6.5 for simulated

simulated 4 vs. 4 human dataset and simulated allelic dataset datasets respectively.

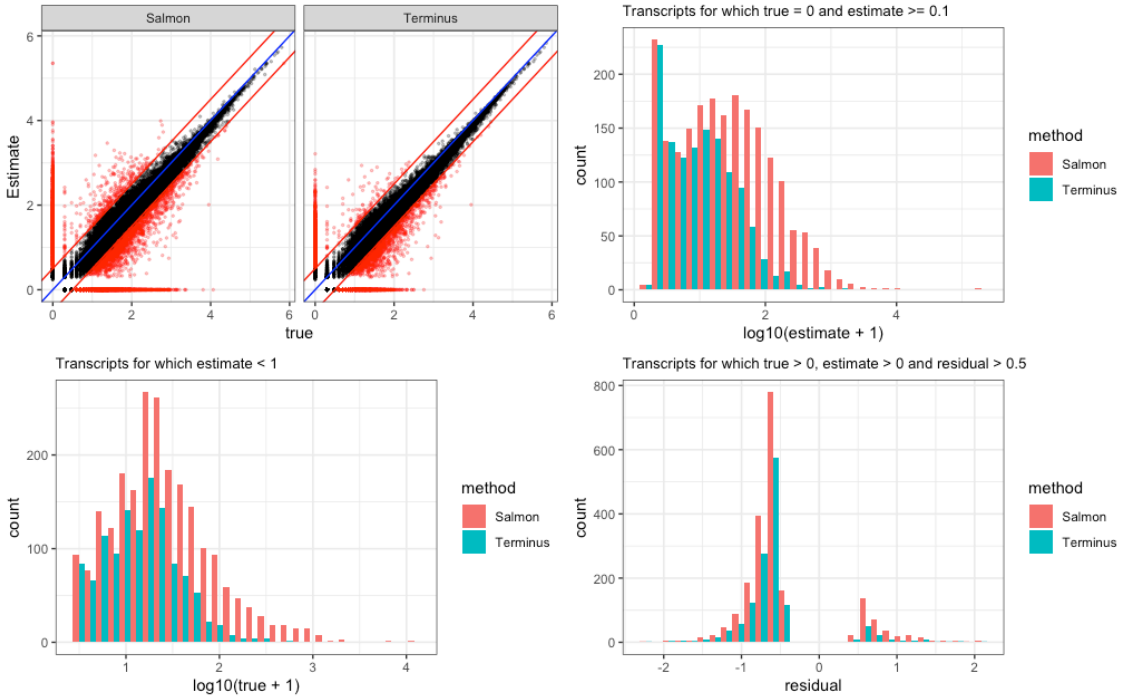


Figure 6.4: A comparative view of *salmon* and terminus on the simulated 4 vs. 4 human dataset. Among transcripts that are truly expressed. Following the same interpretation as described in Fig. 4, terminus grouping reduces the number of mis-estimated transcripts.

Both the datasets categorically show different cases conditioned on the true expression of the transcripts and the corresponding estimates from the respective tools. Starting from the scatter plot at the right top corner, we observe the spread of expressed transcripts that are estimated to be unexpressed by the two tools (the horizontal spread at $y = 0$ marked in red) and the truly unexpressed transcripts that are expressed by the respective tools (the vertical spread at $x = 0$ marked in red). The number of such mis-estimated points are significantly decreased in terminus compared to *salmon*. Additionally, we observe a shrinkage in the mis-estimated points (points away from $x = y$ line) in the scatter plot for terminus, signifying

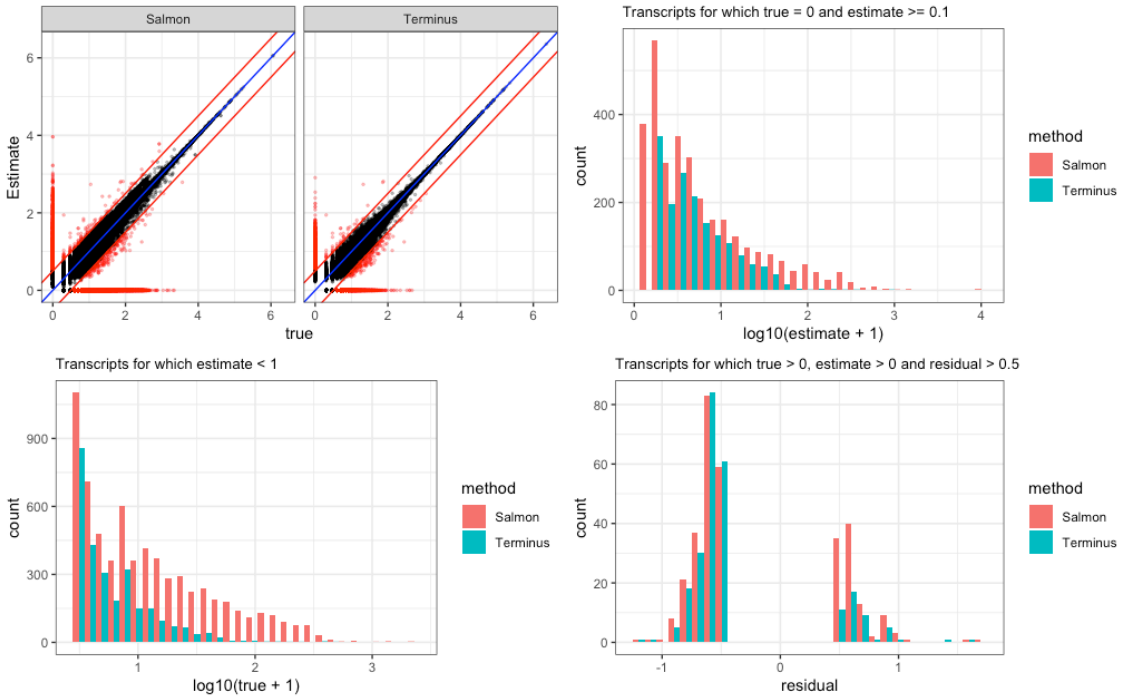


Figure 6.5: A comparative view of *salmon* and terminus on the simulated allelic dataset. The metrics are similar to that of Fig. 4

that the tool has reduced the number of mis-estimated abundances. To capture the magnitude of such mis-estimations in both *salmon* and terminus, we observe the histograms of the abundances conditioned on true expression values. The first histogram (top-right corner of fig. 6.4 and fig. 6.5) considers the transcripts for which the true expression is zero. The shift of the transcript count distribution for different levels of mis-estimated abundances demonstrates that the magnitude of mis-estimation is more severe in *salmon*, at the transcript level, compared to terminus, at the group level. The same trend is to be seen for lowly-abundant transcripts (abundance values less than 1 in the bottom left corner). The last plot shows the histogram for transcripts for which $|\log_{10}(y+1) - \log_{10}(x+1)| \geq 0.5$, where y is the estimated abundance and x is the true abundance. In this case also

we see that *salmon* has more mis-estimated abundances than terminus.

6.2.2 Exploratory analysis for mis-estimated abundances in simulated 4 vs. 4 human dataset

For the simulated 4 vs. 4 human dataset we have selected a few transcripts from the human transcriptome where the transcript abundance estimation from *salmon* deviates from the simulated counts by a substantial margin. To emphasize the effect of such mis-estimation in the downstream pipeline, we have chosen one of the replicates (among 4) where both the control and treatment samples are taken into account. We compared the log fold change (termed as LFC) of the transcript-level fragment counts simulated by polyester [40] and the counts estimated by *salmon* between the two samples. Further, we identified only the transcripts for which, i. the LFCs are reversed (i.e. while the true count based log fold change is positive the LFC from *salmon* counts are negative or vice versa) and ii. the absolute difference of the LFCs are more that 0.5. The goal of such a filter is to consider the transcripts which are estimated to be up-regulated while they are, in reality, down-regulated and vice-versa.

The distribution for the log fold change *for these transcripts with mis-estimated fold changes* is shown in fig. 6.6. We observe that, as expected, the estimated log fold change distribution is different from the true distribution. For this particular experiment, there are 2194 such transcripts. It spans through 232 different gene families. Terminus groups 669 transcripts out of these 2195 into different groups

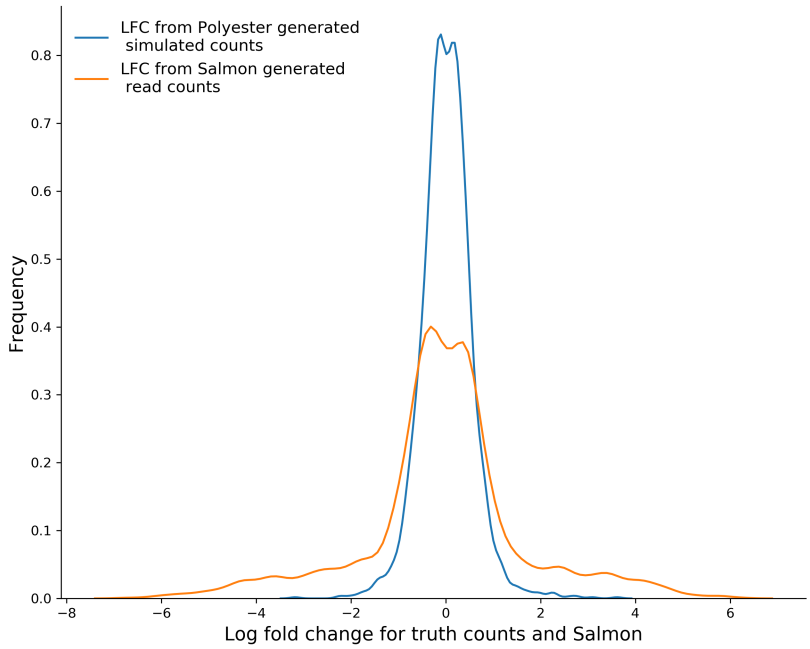


Figure 6.6: Distributions of log fold changes between control and treatment samples from one of the replicates of simulated 4 vs. 4 human dataset. The distribution is on a subset of transcripts as defined in section 6.2.2

(note that the groups may contain transcripts outside this set).

Figure 6.7 captures this phenomenon of groping graphically. The abundance estimates by *salmon* are plotted in blue, while group-level abundance estimates (groups of which these transcripts are members) are plotted in red. An arrow originates at a blue point that is to be grouped by terminus, and points to a red point that is the group-level estimate for the group containing this transcript. The pattern of arrows show that the mis-estimated transcripts are away from the $x = y$ line and, when grouped by terminus, the *grouped estimates* are much closer to the $x = y$ line (i.e. the grouped abundances are much closer to the corresponding grouped true counts).

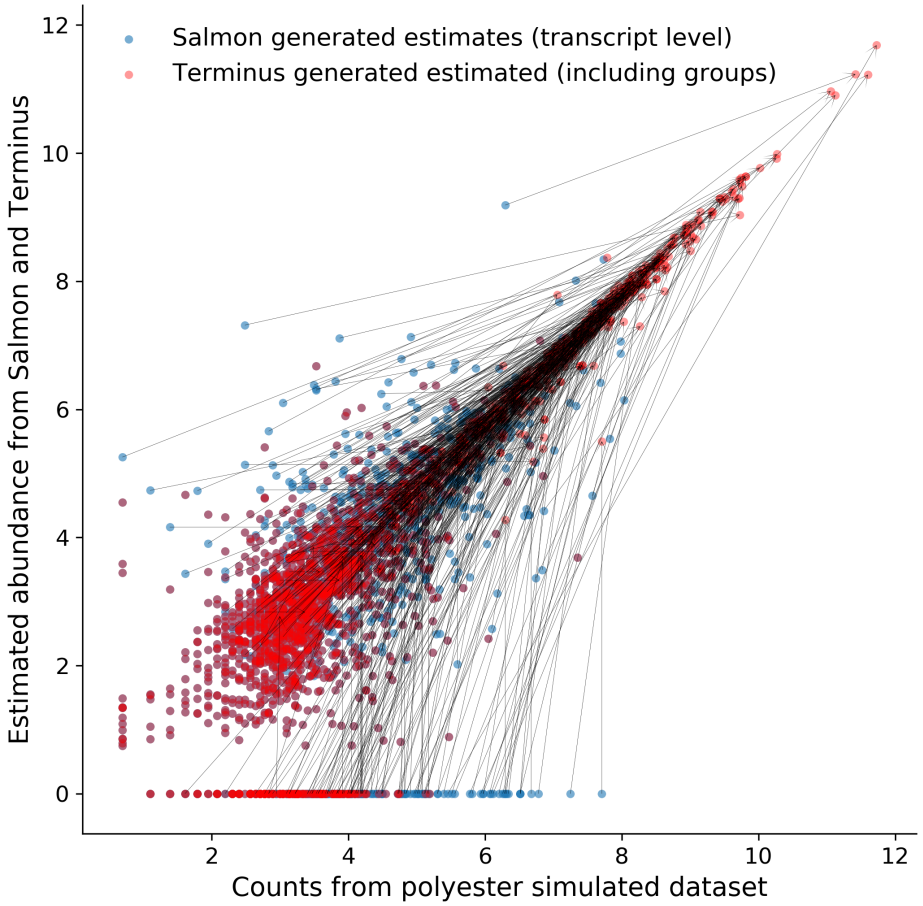


Figure 6.7: The transcripts which are mis-estimated by *salmon* are often grouped by terminus. The arrows originate from an abundance values estimated by *salmon* (marked in blue) for a transcript and points to a group (marked in red) that is formed by terminus.

6.2.3 Exploratory analysis for mis-estimated transcripts in GEUVADIS sample $ERR188204$

We further experimented with a samples from GEUVADIS [114], $ERR188204$. Due to the absence of ground-truth, to asses the performance of *salmon* and terminus, we have created a dataset derived from $ERR188204$ by artificially shortening

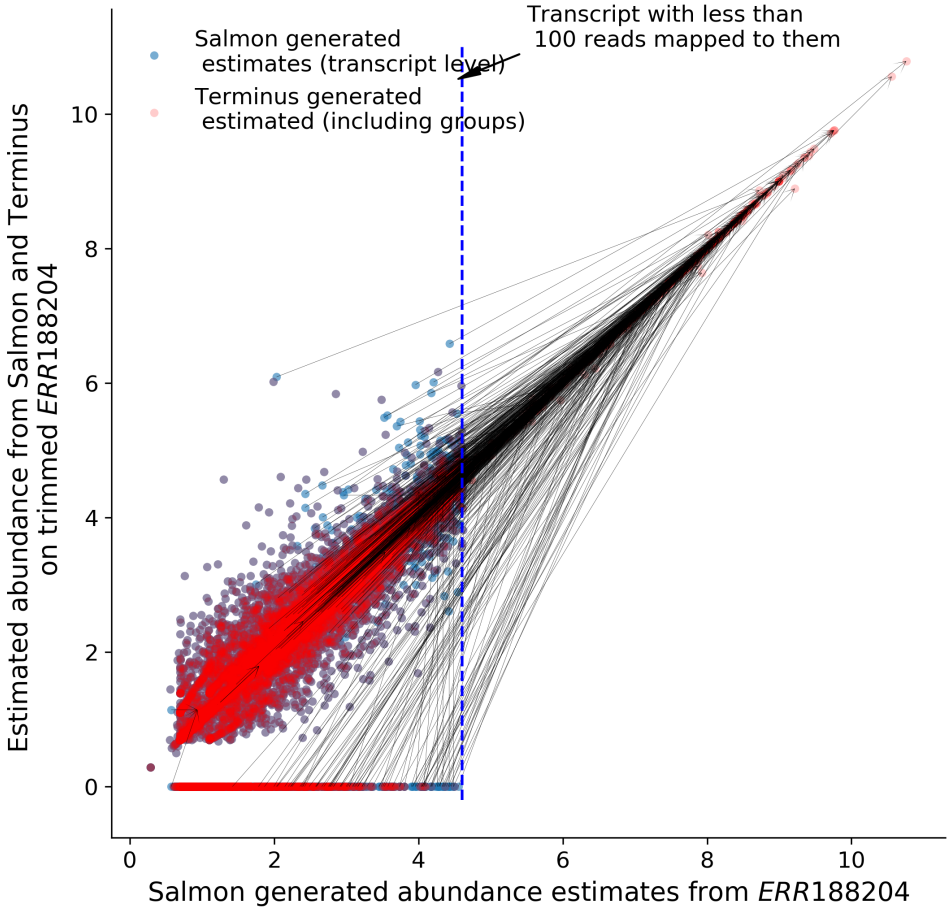


Figure 6.8: The effect of groping transcript by terminus following the same convention of 6.7. The dashed blue vertical line signifies the transcripts which are expressed with a read count of 100.

the reads from the `FASTQ` files. To be specific, 26 nucleotides are trimmed from the 76 nucleotide reads in the original `FASTQ` file. The unaltered dataset is used as the ground truth, while the quantification results on the trimmed dataset are assessed. To further increase the resolution of the performance comparison, we considered transcripts that originate from gene families where there are a numerous transcript isoforms present (the specific selection procedure is defined below).

We identify such gene families of interest by calculating the ratio of total number of transcripts versus the total number of genes within that family. The ratio is termed as splicing repertoire (SR). As an illustrative example, the `N-myc downregulated gene family` (NDRG) has 4 genes and 167 transcripts, making it one of the highest SR-scored gene families (with score $167/4 = 41.75$). For this experiment, we consider transcripts that satisfies two conditions, namely, i. it belongs to a gene family with SR score more than 10 and ii. it is expressed with a coverage lower than 100 reads. Figure 6.8 (following similar convention as of fig. 6.7) shows the scatter plot for such transcripts from *salmon* (labeled with blue) and the effect of terminus (red) grouping leading to a shift towards the $x = y$ line, improving the overall correlation considerably.

6.3 Biological relevance of terminus groups

The groups created from terminus are strictly data-driven, meaning the presence of uncertainty within the dataset drives the formation of specific groups. In an RNA-seq experiment it is often the case that the assigned reads are not enough to resolve the abundance values at the level of transcripts for some transcripts, while sufficient information is present to perform accurate estimation for other transcripts. In such a situation, fully relying on the higher level annotation (genes or gene families) to collapse all transcripts falling under this annotation may not be the ideal solution. Moreover, summing up transcripts at that level would eliminate transcript-level inference for the transcripts for which accurate estimation was possible, thereby

defeating the purpose of transcript-level analysis. Providing an intermediate solution, terminus aims to group the transcripts for which the posterior sampling shows a high degree of uncertainty, while keeping the other transcript estimates unchanged.

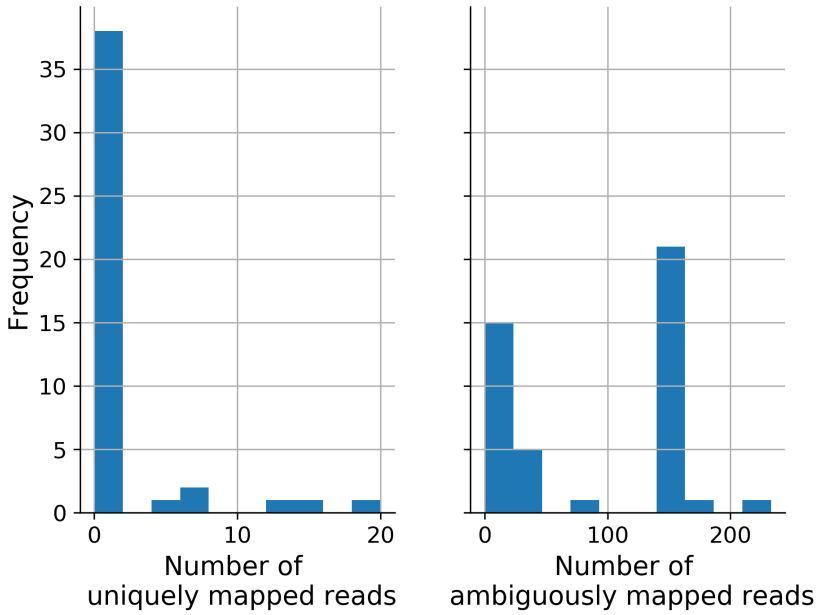


Figure 6.10: Histogram of number of transcripts with respect to uniquely mapped reads and ambiguously mapped reads to from Clustered protocadherins gene family reported by *salmon*

To verify the biological plausibility of groups produced by terminus, we have closely-analyzed the relation between groups generated on the simulated 4 vs. 4 human dataset with the corresponding gene families. One motivating example is the gene family *Clustered protocadherins* or clustered Pcdhs. In the present annotation[1], there are 138 transcripts that are distributed over 59 genes. Instead of grouping the entire family, terminus groups 15 genes within a group. One possible reason for such grouping by terminus is the presence of highly-ambiguous reads that are reported by *salmon*, as shown in fig. 6.10. We believe identifying such grouping within gene

[1]https://biomart.genenames.org/martform/#!/default/HGNC?datasets=hgnc_family_mart

Datasets	Correlation (Spearman)			MARD		
	salmon	random	terminus	*salmon*	random	terminus
Simulated dataset	0.94	0.94	0.96	0.11	0.12	0.09

Table 6.2: Spearman correlation and MARD for simulated 4 vs. 4 human dataset with comparing random partition vs the groups produced by respective algorithms.

family may be useful for many other downstream analyses.

6.4 Comparison with random grouping

In order to verify the efficacy of the groupings produced by terminus, we have generated a random partition within the set of transcripts following the same distribution of group sizes generated by terminus. We observe that a random grouping does not improve the accuracy from the original (ungrouped) estimates at all. Likewise, random grouping does not decrease the accuracy, as one would expect the distribution of errors over random groups to mirror the distribution of estimation errors made at the transcript-level if the grouped transcripts are not related in any meaningful way. The result on one of the simulated 4 vs. 4 human dataset samples are presented in table 6.2.

6.5 Tuning terminus to attain different number of groups

Terminus accepts several tuning parameters that can be used to control the number of groups. The most effective control on the number of groups can be achieved by using changing the consensus threshold that determines what fraction of samples should include the group, in order to count it towards the final group.

Consensus threshold	Number of groups	Spearman Correlation
0.125	7225	0.97
0.25	6192	0.96
0.50	4972	0.96
0.75	3861	0.96
1.00	2544	0.95

Table 6.3: Spearman correlation and number of groups for simulated 4 vs. 4 human dataset with different values of consensus threshold

For simulated 4 vs. 4 human dataset we have changed the consensus threshold parameter from 0.125 to 1.0, which dictates the number of groups when, a group has to be present in at least one sample to the condition where the group has to be present in all samples. Table 6.3 shows the effect in the number of groups and corresponding correlation when we change the consensus threshold.

6.6 Expanding the principle of grouping

The presence of non-unique sequences can greatly affect the accuracy of transcript quantification. Careful analysis of the posterior samples from the underlying probabilistic model not only provides a measure of uncertainty around a point estimate of abundance, but also indicates which groups of transcripts may have abundances that are particularly difficult to distinguish individually but which have estimable abundance as a group. Terminus demonstrates that Gibbs samples can be used to identify groups of transcripts that exhibit high inferential uncertainty on their own, but which exhibit much lower uncertainty as a group. We show how terminus uses the information encoded in range-factorized equivalence classes, readily available after quantification with *salmon*, to tremendously accelerate the grouping

process. Terminus writes the new expression estimates and posterior samples with the group information in the same exact format as that of *salmon*, enabling any downstream pipeline that accepts a similar format [102] to directly run on terminus output.

The groups computed by terminus represent abundance estimates reported at the resolution that is actually supported by the underlying experimental data. In a typical experiment, this is neither at the gene level nor the transcript level. Some transcripts, even from complex, multi-isoform genes, can have their abundances accurately estimated and with low uncertainty, while other transcripts cannot. Rather than pre-defining the resolution at which the analysis will be performed, and subjecting the results to either overwhelming uncertainty or to insufficient biological resolution, terminus allows the determination of transcriptional groups whose abundance can be confidently estimated in a given dataset, and represents, in this sense, a data-driven approach to transcriptome analysis.

Further, we demonstrate that terminus creates biologically relevant groups that reflect the underlying hierarchy of genes and gene families. This shows the potential of terminus to be utilized for applications of data-driven clustering of biological sequences, such as clustering de-novo contigs (where the annotation is not known) or for clustering related strains in metagenomic samples.

From a conceptual perspective, terminus provides a novel approach for grouping complex interactions between biological sequence without having the prior information about the annotation itself. It first prunes the possible space of pairwise collapses by examining the structure induced by the range-factorized equivalence

classes, and later, uses a iterative greedy technique to collapse transcripts that locally maximize the objective being optimized (i.e. the reduction in inferential relative variance). It is possible that this generic framework can be extended to other aspects of grouping and clustering such as taxonomic classification.

6.7 Existing challenges in RNA-seq quantification

6.7.1 Challenge 1: Multi-sample quantification

Chapter 2 discussed the RNA-seq quantification models. Such models unfortunately only account for a single sample, but actual RNA-seq experiments are carried out with multiple replicates under various conditions. Li et al. [115] proposed a mechanism to consider consistent groups and incorporated a common hyperparameter shared between samples. As Li et al. [115] relies on "consistent" groups of samples (experiments) by inspecting the posterior estimates from the individual estimates. The computational performance of such a method unfortunately suffers from the slow convergence of the sampling approach. On the other hand, the model itself does not capture the natural hierarchical structure of a typical RNA-seq experiment. The later issue is addressed by Isolator developed by Jones et al. [116]. Isolator introduced a hierarchical bayesian model consisting of three levels to represent the replicates, conditions and further the replicates within a condition. Additionally, Isolator accounts for many other aspects of the RNA-seq experiment such as splicing, sequence bias, GC bias, etc. Parameters for the model are updated iteratively using Gibbs sampling. Although Isolator provides a promising framework

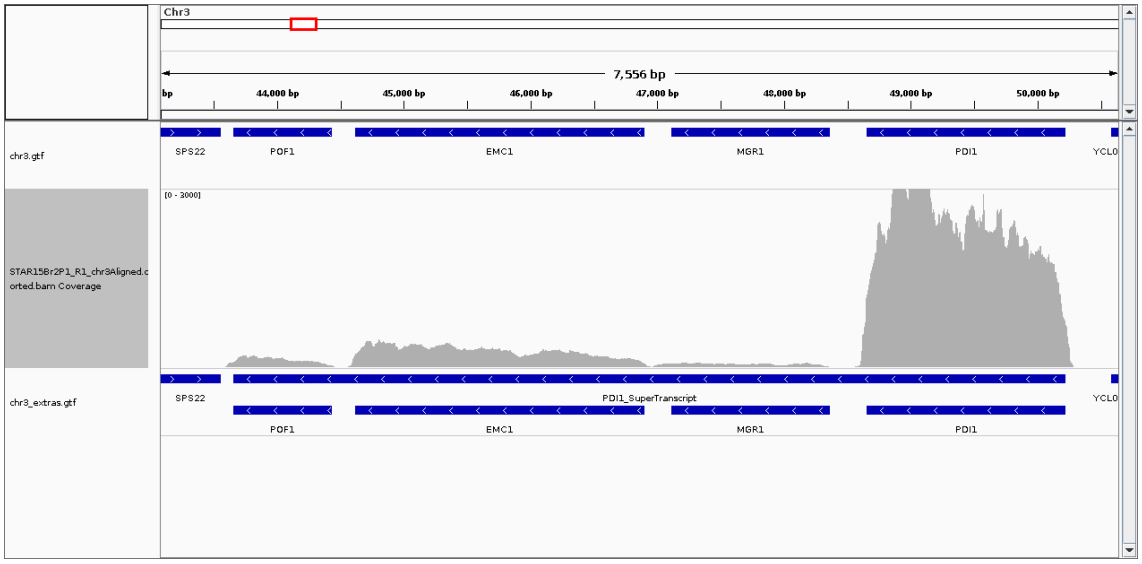


Figure 6.11: A super transcript is receiving all the counts when the sub-transcripts are also abundant

for resolving the RNA-seq quantification problem, one drawback of the method was the innate dependence on the annotation. I believe there is considerable place for improving the aforementioned tools accounting for the facts that isoforms are indeed shared between samples. Various downstream quantities of interest such as differential transctipt expression should be a natural byproduct of such a unified model.

6.7.2 Challenge 2: Anomalies in Quantification

A sophisticated quantification algorithm based on the generative model is not enough to correctly quantify the abundance of all the transcripts. There could be many causes for such mis-estimation, such as uncertainty, missing reference transcripts and the nature of the sequence sharing that happens within isoforms of a gene. Schematic example of the later case is characterized as the "subset effect" in

Transcripts	Before	After
PDI1_SuperTranscript	NA	3582
PDI1	11799	10641
MGR1	229	0
EMC1	2109	847
POF1	283	0

Table 6.4: Abundances before and after adding the PDI1_SuperTranscript

fig. 2.3. A motivating scenario from real biological data is depicted in fig. 6.11(table 6.4) [2].

We believe addressing these challenges can further advance the state-of-the-art RNA-seq quantification methods. These improvements can profoundly impact the discovery of novel isoforms and other downstream analyses.

A closer inspection suggests that the occurrence of such an anomalous abundance estimation can arise from the limitation of the RNA-seq models described in chapter 2. Note that although the coverage plot shown in fig. 6.11 is instrumental in actual analysis of transcript abundance, these kind of coverage metrics are not currently involved in the actual quantification model. One motivating example of such phenomenon could be observed in the expression estimate of transcript POF1. From the coverage plot in the IGV panel it is evident that there are reads that are mapped to POF1, moreover even in the presence of PDI1_SuperTranscript, there is a drop in the accumulation of reads near the junction of POF1 and EMC1. If quantification model takes into account not only the crude read to transcript assignment but rather derives a confidence score for an isoform depending on the continuity of the coverage. Some of these cases are recently identified by Ma and Kingsford [117].

[2]https://github.com/COMBINE-lab/salmon/issues/514 reported by Jason Rogers

Implemented in the tool SAD, Ma and Kingsford [117] characterized the quantification anomalies into two groups, the "adjustable anomalies", that are caused by error in the quantification model, and the "unadjustable anomalies", that are caused by incomplete reference.

The mentioned challenges suggest when the discovery of novel isoforms aids the quantification process, the resulting method can address the case of incomplete reference. Bayesembler [118], SparseIso [119] and FlipFlop [120] attempt to solve this joint problem via an iteratively updating the abundance estimates for possible novel isoforms. While these solutions are useful in specific cases, they are not widely used either due to poor computational performance or lack scalability. Moreover none of these tools acknowledge the coverage profile information explicitly, and rather follow a count based scheme that might suffer from the anomaly alluded briefly in fig. 6.11. I believe an ideal RNA-seq quantification tool will be able to take both the coverage information and isoform discovery while solving the quantification problem.

6.8 Simulating the dynamics of single cell RNA-seq dataset

Single cell RNA-seq experiments generate cell to gene count matrices that represent the gene expression at the cellular resolution. The gene expression is a summarized metric that can be a representative of a cell state. In other words it is a static snapshot of a complex dynamic process. Since, individual cells in the population can represent different cellular states in reality one can utilize a single cell experiment (in contrast to bulk RNA-seq) to extract a range of information that

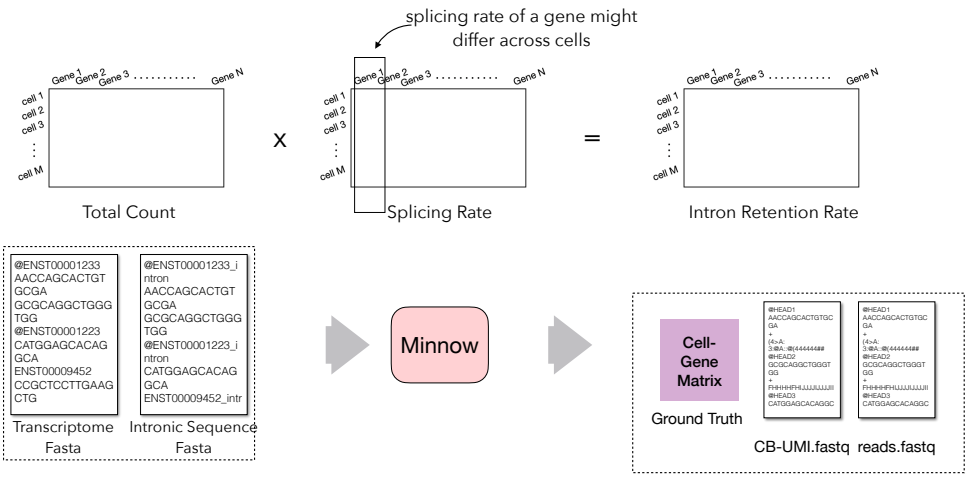


Figure 6.12: Overview of the velocity simulation in *minnow* pipeline: The key input to the velocity simulation pipeline is the presence of splicing rate provided by the user. Reads from introns will be simulated to produce sequences from unspliced mRNAs

capture the dynamic trait of cell development. One prominent example of such a dynamic trait is RNA-velocity [121], that represents a high-dimensional vector that determines the future state of an individual cell on a particular time scale (often hours). A cell state here is determined by the specific stage the mRNA molecules are in along the process of becoming mature mRNA. A simple quantity representing the dynamics of the life cycle of mRNA molecules is the splicing rate, measured by the ratio of unspliced vs. spliced mRNAs. Computationally speaking, the ratio of the number of reads that map with intronic regions and the number of reads that map to the exonic region is a good proxy for the former one.

6.8.1 *minnow* can produce dataset imputed with RNA-velocity

We described the intronic read generation pipeline in fig. 6.12. *minnow* enables the generation of sequence level reads from transcriptome sequences, given a cell to

gene count matrix. A natural extension of *minnow* can generate the simulated reads from unspliced reference by using additional information such as the spliced vs. unspliced ratio and the annotation of intronic regions. Simulated reads that can encode desired splicing rate in the underlying experiment could enable the testing of various statistical frameworks for modeling velocity.

Bibliography

[1] Koen Van Den Berge, Katharina M Hembach, Charlotte Soneson, Simone Tiberi, Lieven Clement, Michael I Love, Rob Patro, and Mark D Robinson. Rna sequencing data: hitchhiker's guide to expression analysis. 2019.

[2] Zachary D Stephens, Skylar Y Lee, Faraz Faghri, Roy H Campbell, Chengxiang Zhai, Miles J Efron, Ravishankar Iyer, Michael C Schatz, Saurabh Sinha, and Gene E Robinson. Big data: astronomical or genomical? *PLoS biology*, 13(7):e1002195, 2015.

[3] Genivaldo Gueiros Zacarias Silva. *Who Is There and What are They Doing? An Agile and Computationally Efficient Framework for Genome Discovery and Annotation from Metagenomic Big Data*. PhD thesis, San Diego State University, 2017.

[4] Ugrappa Nagalakshmi, Zhong Wang, Karl Waern, Chong Shou, Debasish Raha, Mark Gerstein, and Michael Snyder. The transcriptional landscape of the yeast genome defined by RNA sequencing. *Science*, 320(5881):1344–1349, 2008.

[5] Brian T Wilhelm, Samuel Marguerat, Stephen Watt, Falk Schubert, Valerie Wood, Ian Goodhead, Christopher J Penkett, Jane Rogers, and Jürg Bähler. Dynamic repertoire of a eukaryotic transcriptome surveyed at single-nucleotide resolution. *Nature*, 453(7199):1239, 2008.

[6] John C Marioni, Christopher E Mason, Shrikant M Mane, Matthew Stephens, and Yoav Gilad. Rna-seq: an assessment of technical reproducibility and comparison with gene expression arrays. *Genome research*, 18(9):1509–1517, 2008.

[7] Ryan Lister, Ronan C O'Malley, Julian Tonti-Filippini, Brian D Gregory, Charles C Berry, A Harvey Millar, and Joseph R Ecker. Highly integrated single-base resolution maps of the epigenome in Arabidopsis. *Cell*, 133(3): 523–536, 2008.

[8] Ali Mortazavi, Brian A Williams, Kenneth McCue, Lorian Schaeffer, and Barbara Wold. Mapping and quantifying mammalian transcriptomes by rna-seq. *Nature methods*, 5(7):621–628, 2008.

[9] David Weese, Manuel Holtgrewe, and Knut Reinert. RazerS 3: faster, fully sensitive read mapping. *Bioinformatics*, 28(20):2592–2599, 2012.

[10] Enrico Siragusa, David Weese, and Knut Reinert. Fast and accurate read mapping with approximate seeds and multiple backtracking. *Nucleic acids research*, 41(7):e78–e78, 2013.

[11] Ben Langmead, Cole Trapnell, Mihai Pop, and Steven L Salzberg. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome biology*, 10(3):R25, 2009.

[12] Ben Langmead and Steven L Salzberg. Fast gapped-read alignment with Bowtie 2. *Nature methods*, 9(4):357–359, 2012.

[13] Heng Li and Richard Durbin. Fast and accurate short read alignment with Burrows–Wheeler transform. *Bioinformatics*, 25(14):1754–1760, 2009.

[14] Heng Li. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. *arXiv preprint arXiv:1303.3997*, 2013.

[15] Yang Liao, Gordon K Smyth, and Wei Shi. The Subread aligner: fast, accurate and scalable read mapping by seed-and-vote. *Nucleic acids research*, 41(10): e108–e108, 2013.

[16] Heng Li, Jue Ruan, and Richard Durbin. Mapping short dna sequencing reads and calling variants using mapping quality scores. *Genome research*, 18(11): 1851–1858, 2008.

[17] Matei Zaharia, William J Bolosky, Kristal Curtis, Armando Fox, David Patterson, Scott Shenker, Ion Stoica, Richard M Karp, and Taylor Sittler. Faster and more accurate sequence alignment with SNAP. *arXiv preprint arXiv:1111.5572*, 2011.

[18] Thomas D Wu and Serban Nacu. Fast and SNP-tolerant detection of complex variants and splicing in short reads. *Bioinformatics*, 26(7):873–881, 2010.

[19] Alexander Dobin, Carrie A Davis, Felix Schlesinger, Jorg Drenkow, Chris Zaleski, Sonali Jha, Philippe Batut, Mark Chaisson, and Thomas R Gingeras. Star: ultrafast universal rna-seq aligner. *Bioinformatics*, 29(1):15–21, 2013.

[20] Daehwan Kim, Ben Langmead, and Steven L Salzberg. HISAT: a fast spliced aligner with low memory requirements. *Nature Methods*, 12(4):357–360, 2015.

[21] Shihao Shen, Juw Won Park, Zhi-xiang Lu, Lan Lin, Michael D Henry, Ying Nian Wu, Qing Zhou, and Yi Xing. rMATS: robust and flexible detection of differential alternative splicing from replicate RNA-Seq data. *Proceedings of the National Academy of Sciences*, 111(51):E5593–E5601, 2014.

[22] Jorge Vaquero-Garcia, Alejandro Barrera, Matthew R Gazzara, Juan Gonzalez-Vallinas, Nicholas F Lahens, John B Hogenesch, Kristen W Lynch, and Yoseph Barash. A new view of transcriptome complexity and regulation through the lens of local splicing variations. *Elife*, 5:e11752, 2016.

[23] Daniel Nicorici, Mihaela Satalan, Henrik Edgren, Sara Kangaspeska, Astrid Murumagi, Olli Kallioniemi, Sami Virtanen, and Olavi Kilkku. FusionCatcher-a tool for finding somatic fusion genes in paired-end RNA-sequencing data. *bioRxiv*, page 011650, 2014.

[24] Nadia M Davidson, Ian J Majewski, and Alicia Oshlack. JAFFA: High sensitivity transcriptome-focused fusion gene detection. *Genome medicine*, 7(1): 43, 2015.

[25] Rob Patro, Stephen M Mount, and Carl Kingsford. Sailfish enables alignment-free isoform quantification from RNA-seq reads using lightweight algorithms. *Nature biotechnology*, 32(5):462–464, 2014.

[26] Avi Srivastava, Hirak Sarkar, Nitish Gupta, and Robert Patro. RapMap: a rapid, sensitive and accurate tool for mapping RNA-seq reads to transcriptomes. *Bioinformatics*, 32(12):192–200, 2016. doi: 10.1093/bioinformatics/btw277. URL `http://dx.doi.org/10.1093/bioinformatics/btw277`.

[27] Nicolas L Bray, Harold Pimentel, Páll Melsted, and Lior Pachter. Near-optimal probabilistic RNA-seq quantification. *Nature Biotechnology*, 34(5):525–527, 2016.

[28] Rob Patro, Geet Duggal, Michael I Love, Rafael A Irizarry, and Carl Kingsford. Salmon provides fast and bias-aware quantification of transcript expression. *Nature Methods*, 2017.

[29] Michael J Axtell. Butter: High-precision genomic alignment of small RNA-seq data. *bioRxiv*, page 007427, 2014.

[30] Hirak Sarkar, Mohsen Zakeri, Laraib Malik, and Rob Patro. Towards selective-alignment: Bridging the accuracy gap between alignment-based and alignment-free transcript quantification. In *Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, BCB '18, pages 27–36, New York, NY, USA, 2018. ACM.

[31] Ernest Turro, Shu-Yi Su, Ângela Gonçalves, Lachlan JM Coin, Sylvia Richardson, and Alex Lewin. Haplotype and isoform specific expression estimation using multi-mapping rna-seq reads. *Genome biology*, 12(2):1, 2011.

[32] Marius Nicolae, Serghei Mangul, Ion I Măndoiu, and Alex Zelikovsky. Estimation of alternative splicing isoform frequencies from rna-seq data. *Algorithms for Molecular Biology*, 6(1):1, 2011.

[33] Avi Srivastava, Hirak Sarkar, Laraib Malik, and Rob Patro. Accurate, fast and lightweight clustering of de novo transcriptomes using fragment equivalence classes. *arXiv preprint arXiv:1604.03250*, 2016.

[34] Jacob Pritt and Ben Langmead. Boiler: lossy compression of RNA-seq alignments using coverage vectors. *Nucleic Acids Research*, 44(16):e133–e133, jun 2016. doi: 10.1093/nar/gkw540. URL `http://dx.doi.org/10.1093/nar/gkw540`.

[35] Faraz Hach, Ibrahim Numanagić, Can Alkan, and S Cenk Sahinalp. SCALCE: boosting sequence compression algorithms using locally consistent encoding. *Bioinformatics*, 28(23):3051–3057, 2012.

[36] Hirak Sarkar and Rob Patro. Quark enables semi-reference-based compression of rna-seq data. *Bioinformatics*, 33(21):3380–3386, 2017.

[37] Avi Srivastava, Hirak Sarkar, Nitish Gupta, and Rob Patro. RapMap: a rapid, sensitive and accurate tool for mapping RNA-seq reads to transcriptomes. *Bioinformatics*, 32(12):i192–i200, 2016.

[38] Bo Li and Colin N Dewey. Rsem: accurate transcript quantification from rna-seq data with or without a reference genome. *BMC bioinformatics*, 12(1): 323, 2011.

[39] Thasso Griebel, Benedikt Zacher, Paolo Ribeca, Emanuele Raineri, Vincent Lacroix, Roderic Guigó, and Michael Sammeth. Modelling and simulating generic rna-seq experiments with the flux simulator. *Nucleic acids research*, 40(20):10073–10083, 2012.

[40] Alyssa C Frazee, Andrew E Jaffe, Ben Langmead, and Jeffrey T Leek. Polyester: simulating RNA-seq datasets with differential transcript expression. *Bioinformatics*, 31(17):2778–2784, 2015.

[41] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[42] Bo Li, Victor Ruotti, Ron M Stewart, James A Thomson, and Colin N Dewey. Rna-seq gene expression estimation with read mapping uncertainty. *Bioinformatics*, 26(4):493–500, 2010.

[43] Hy Vuong, Thao Truong, Thang Tran, and Son Pham. A revisit of rsem generative model and its em algorithm for quantifying transcript abundances. *bioRxiv*, page 503672, 2018.

[44] Peter Glaus, Antti Honkela, and Magnus Rattray. Identifying differentially expressed transcripts from RNA-seq data with biological variation. *Bioinformatics*, 28(13):1721–1728, 2012.

[45] Rob Patro, Geet Duggal, Michael I Love, Rafael A Irizarry, and Carl Kingsford. Salmon provides fast and bias-aware quantification of transcript expression. *Nature Methods*, 14(4):417–419, mar 2017. doi: 10.1038/nmeth.4197. URL `https://doi.org/10.1038%2Fnmeth.4197`.

[46] Hirak Sarkar, Mohsen Zakeri, Laraib Malik, and Rob Patro. Towards selective-alignment: Bridging the accuracy gap between alignment-based and alignment-free transcript quantification. In *Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, pages 27–36, 2018.

[47] Jiarui Zhou, Zhen Ji, Zexuan Zhu, and Shan He. Compression of next-generation sequencing quality scores using memetic algorithm. *BMC Bioinformatics*, 15(Suppl 15):S10, 2014. ISSN 1471-2105. doi: 10.1186/1471-2105-15-s15-s10. URL `http://dx.doi.org/10.1186/1471-2105-15-S15-S10`.

[48] Greg Malysa, Mikel Hernaez, Idoia Ochoa, Milind Rao, Karthik Ganesan, and Tsachy Weissman. Qvz: lossy compression of quality values. *Bioinformatics*, page btv330, 2015.

[49] Lilian Janin, Giovanna Rosone, and Anthony J Cox. Adaptive reference-free compression of sequence quality scores. *Bioinformatics*, page btt257, 2013.

[50] Rob Patro, Stephen M Mount, and Carl Kingsford. Sailfish enables alignment-free isoform quantification from rna-seq reads using lightweight algorithms. *Nature biotechnology*, 32(5):462–464, 2014.

[51] Nicolas L Bray, Harold Pimentel, Páll Melsted, and Lior Pachter. Near-optimal probabilistic RNA-seq quantification. *Nature Biotechnology*, 34(5):525, 2016.

[52] Rob Patro and Carl Kingsford. Data-dependent bucketing improves reference-free compression of sequencing reads. *Bioinformatics*, page btv248, 2015.

[53] Udi Manber and Gene Myers. Suffix arrays: a new method for on-line string searches. *siam Journal on Computing*, 22(5):935–948, 1993.

[54] Chelsea J-T Ju, Ruirui Li, Zhengliang Wu, Jyun-Yu Jiang, Zhao Yang, and Wei Wang. Fleximer: Accurate quantification of rna-seq via variable-length k-mers. In *Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, pages 263–272. ACM, 2017.

[55] Zhaojun Zhang and Wei Wang. RNA-Skim: a rapid method for RNA-Seq quantification at transcript level. *Bioinformatics*, 30(12):i283–i292, 2014.

[56] Bo Liu, Hongzhe Guo, Michael Brudno, and Yadong Wang. deBGA: read alignment with de bruijn graph-based seed and extension. *Bioinformatics*, 32 (21):3224–3232, jul 2016. doi: 10.1093/bioinformatics/btw371. URL `https://doi.org/10.1093/bioinformatics/btw371`.

[57] Gene Myers. A fast bit-vector algorithm for approximate string matching based on dynamic programming. *Journal of the ACM (JACM)*, 46(3):395–415, 1999.

[58] Martin Šošić and Mile Šikić. Edlib: a C/C++ library for fast, exact sequence alignment using edit distance. *Bioinformatics*, 33(9):1394, 2017.

[59] Cyan. xxHash — Extremely fast hash algorithm. `http://cyan4973.github.io/xxHash/`, 2018. Accessed: 2018-04-30.

[60] Hirak Sarkar, Avi Srivastava, and Rob Patro. Minnow: a principled framework for rapid simulation of dscrna-seq data at the read level. *Bioinformatics*, 35 (14):i136–i144, 2019.

[61] Tamar Hashimshony, Florian Wagner, Noa Sher, and Itai Yanai. Cel-seq: single-cell rna-seq by multiplexed linear amplification. *Cell reports*, 2(3):666–673, 2012.

[62] Dominic Grün, Anna Lyubimova, Lennart Kester, Kay Wiebrands, Onur Basak, Nobuo Sasaki, Hans Clevers, and Alexander van Oudenaarden. Single-cell messenger rna sequencing reveals rare intestinal cell types. *Nature*, 525 (7568):251, 2015.

[63] Cole Trapnell. Defining cell types and states with single-cell genomics. *Genome research*, 25(10):1491–1498, 2015.

[64] Yosef Buganim, Dina A Faddah, Albert W Cheng, Elena Itskovich, Styliani Markoulaki, Kibibi Ganz, Sandy L Klemm, Alexander van Oudenaarden, and Rudolf Jaenisch. Single-cell expression analyses during cellular reprogramming reveal an early stochastic and a late hierarchic phase. *Cell*, 150(6):1209–1222, 2012.

[65] Evan Z Macosko, Anindita Basu, Rahul Satija, James Nemesh, Karthik Shekhar, Melissa Goldman, Itay Tirosh, Allison R Bialas, Nolan Kamitaki, Emily M Martersteck, et al. Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets. *Cell*, 161(5):1202–1214, 2015.

[66] Grace XY Zheng, Jessica M Terry, Phillip Belgrader, Paul Ryvkin, Zachary W Bent, Ryan Wilson, Solongo B Ziraldo, Tobias D Wheeler, Geoff P McDermott, Junjie Zhu, et al. Massively parallel digital transcriptional profiling of single cells. *Nature Communications*, 8:14049, 2017.

[67] Allon M Klein, Linas Mazutis, Ilke Akartuna, Naren Tallapragada, Adrian Veres, Victor Li, Leonid Peshkin, David A Weitz, and Marc W Kirschner. Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells. *Cell*, 161(5):1187–1201, 2015.

[68] Xiaojie Qiu, Qi Mao, Ying Tang, Li Wang, Raghav Chawla, Hannah A Pliner, and Cole Trapnell. Reversed graph embedding resolves complex single-cell trajectories. *Nature Methods*, 14(10):979, 2017.

[69] Gioele La Manno, Ruslan Soldatov, Amit Zeisel, Emelie Braun, Hannah Hochgerner, Viktor Petukhov, Katja Lidschreiber, Maria E Kastriti, Peter Lönnerberg, Alessandro Furlan, et al. Rna velocity of single cells. *Nature*, 560 (7719):494, 2018.

[70] Rahul Satija, Jeffrey A Farrell, David Gennert, Alexander F Schier, and Aviv Regev. Spatial reconstruction of single-cell gene expression data. *Nature biotechnology*, 33(5):495, 2015.

[71] Vladimir Yu Kiselev, Kristina Kirschner, Michael T Schaub, Tallulah Andrews, Andrew Yiu, Tamir Chandra, Kedar N Natarajan, Wolf Reik, Mauricio Barahona, Anthony R Green, et al. Sc3: consensus clustering of single-cell rna-seq data. *Nature methods*, 14(5):483, 2017.

[72] Luke Zappia, Belinda Phipson, and Alicia Oshlack. Exploring the single-cell rna-seq analysis landscape with the scrna-tools database. *PLoS computational biology*, 14(6):e1006245, 2018.

[73] Emma Pierson and Christopher Yau. Zifa: Dimensionality reduction for zero-inflated single-cell gene expression analysis. *Genome biology*, 16(1):241, 2015.

[74] Davide Risso, Fanny Perraudeau, Svetlana Gribkova, Sandrine Dudoit, and Jean-Philippe Vert. Zinb-wave: A general and flexible method for signal extraction from single-cell rna-seq data. *bioRxiv*, page 125112, 2017.

[75] Greg Finak, Andrew McDavid, Masanao Yajima, Jingyuan Deng, Vivian Gersuk, Alex K Shalek, Chloe K Slichter, Hannah W Miller, M Juliana McElrath, Martin Prlic, et al. Mast: a flexible statistical framework for assessing transcriptional changes and characterizing heterogeneity in single-cell rna sequencing data. *Genome biology*, 16(1):278, 2015.

[76] Wei Vivian Li and Jingyi Jessica Li. An accurate and robust imputation method scimpute for single-cell rna-seq data. *Nature communications*, 9(1): 997, 2018.

[77] Luke Zappia, Belinda Phipson, and Alicia Oshlack. Splatter: simulation of single-cell rna sequencing data. *Genome biology*, 18(1):174, 2017.

[78] Beate Vieth, Christoph Ziegenhain, Swati Parekh, Wolfgang Enard, and Ines Hellmann. powsimr: power analysis for bulk and single cell rna-seq experiments. *Bioinformatics*, 33(21):3486–3488, 2017.

[79] Thasso Griebel, Benedikt Zacher, Paolo Ribeca, Emanuele Raineri, Vincent Lacroix, Roderic Guigó, and Michael Sammeth. Modelling and simulating generic rna-seq experiments with the flux simulator. *Nucleic acids research*, 40(20):10073–10083, 2012.

[80] Avi Srivastava, Laraib Malik, Tom Sean Smith, Ian Sudbery, and Rob Patro. Alevin efficiently estimates accurate gene abundances from dscrna-seq data. *bioRxiv*, 2018. doi: 10.1101/335000. URL `https://www.biorxiv.org/content/early/2018/10/24/335000`.

[81] Jingshu Wang, Mo Huang, Eduardo Torre, Hannah Dueck, Sydney Shaffer, John Murray, Arjun Raj, Mingyao Li, and Nancy R Zhang. Gene expression distribution deconvolution in single-cell rna sequencing. *Proceedings of the National Academy of Sciences*, 115(28):E6437–E6446, 2018.

[82] Simone Picelli, Åsa K Björklund, Omid R Faridani, Sven Sagasser, Gösta Winberg, and Rickard Sandberg. Smart-seq2 for sensitive full-length transcriptome profiling in single cells. *Nature methods*, 10(11):1096, 2013.

[83] Ángeles Arzalluz-Luque and Ana Conesa. Single-cell rnaseq for the study of isoforms—how is that possible? *Genome biology*, 19(1):110, 2018.

[84] Jennifer Westoby, Marcela Sjöberg Herrera, Anne C. Ferguson-Smith, and Martin Hemberg. Simulation-based benchmarking of isoform quantification in single-cell rna-seq. *Genome Biology*, 19(1):191, Nov 2018. ISSN 1474-760X. doi: 10.1186/s13059-018-1571-5. URL `https://doi.org/10.1186/s13059-018-1571-5`.

[85] Jianfei Hu, Eli Boritz, William Wylie, and Daniel C Douek. Stochastic principles governing alternative splicing of rna. *PLoS computational biology*, 13(9): e1005761, 2017.

[86] Daniel Zerbino and Ewan Birney. Velvet: algorithms for de novo short read assembly using de bruijn graphs. *Genome research*, pages gr–074492, 2008.

[87] Ilia Minkin, Son Pham, and Paul Medvedev. Twopaco: An efficient algorithm to build the compacted de bruijn graph from many complete genomes. *Bioinformatics*, 33(24):4024–4032, 2016.

[88] Iraad F Bronner, Michael A Quail, Daniel J Turner, and Harold Swerdlow. Improved protocols for illumina sequencing. *Current protocols in human genetics*, 79(1):18–2, 2013.

[89] Tom Smith, Andreas Heger, and Ian Sudbery. UMI-tools: modeling sequencing errors in unique molecular identifiers to improve quantification accuracy. *Genome Research*, 27(3):491–499, 2017.

[90] Katharine Best, Theres Oakes, James M Heather, John Shawe-Taylor, and Benny Chain. Computational analysis of stochastic heterogeneity in pcr amplification efficiency revealed by single molecule barcoding. *Scientific reports*, 5:14629, 2015.

[91] Baraa Orabi, Emre Erhan, Brian McConeghy, Stanislav V Volik, Stephane Le Bihan, Robert Bell, Colin C Collins, Cedric Chauve, and Faraz Hach. Alignment-free clustering of umi tagged dna molecules. *Bioinformatics*, page bty888, 2018. doi: 10.1093/bioinformatics/bty888. URL `http://dx.doi.org/10.1093/bioinformatics/bty888`.

[92] Alexander Dobin, Carrie A Davis, Felix Schlesinger, Jorg Drenkow, Chris Zaleski, Sonali Jha, Philippe Batut, Mark Chaisson, and Thomas R Gingeras. STAR: ultrafast universal RNA-seq aligner. *Bioinformatics*, 29(1):15–21, 2013.

[93] Yang Liao, Gordon K Smyth, and Wei Shi. featureCounts: an efficient general purpose program for assigning sequence reads to genomic features. *Bioinformatics*, 30(7):923–930, 2013.

[94] Marina Meilă. Comparing clusterings?an information based distance. *Journal of multivariate analysis*, 98(5):873–895, 2007.

[95] Andrew Butler, Paul Hoffman, Peter Smibert, Efthymia Papalexi, and Rahul Satija. Integrating single-cell transcriptomic data across different conditions, technologies, and species. *Nature biotechnology*, 36(5):411, 2018.

[96] Hirak Sarkar, Avi Srivastava, Hector Corrada Bravo, Michael I Love, and Rob Patro. Terminus enables the discovery of data-driven, robust transcript groups from rna-seq data. *bioRxiv*, 2020.

[97] Ben Langmead and Steven L Salzberg. Fast gapped-read alignment with Bowtie 2. *Nature methods*, 9(4):357–359, 2012.

[98] Bo Li and Colin N Dewey. Rsem: accurate transcript quantification from rna-seq data with or without a reference genome. *BMC bioinformatics*, 12(1): 1, 2011.

[99] Ernest Turro, William J Astle, and Simon Tavare. Flexible analysis of RNA-seq data using mixed effects models. *Bioinformatics*, 30(2):180–188, 2014.

[100] Sahar Al Seesi, Yvette Temate Tiagueu, Alexander Zelikovsky, and Ion I Măndoiu. Bootstrap-based differential gene expression analysis for RNA-Seq data with and without replicates. In *BMC genomics*, volume 15, page S2. BioMed Central, 2014.

[101] Harold Pimentel, Nicolas L Bray, Suzette Puente, Páll Melsted, and Lior Pachter. Differential analysis of RNA-seq incorporating quantification uncertainty. *Nature Methods*, 14(7):687, 2017.

[102] Anqi Zhu, Avi Srivastava, Joseph G Ibrahim, Rob Patro, and Michael I Love. Nonparametric expression analysis using inferential replicate counts. *Nucleic Acids Research*, 47(18):e105–e105, 2019.

[103] Christelle Robert and Mick Watson. Errors in RNA-Seq quantification affect genes of relevance to human disease. *Genome Biology*, 16(1):177, 2015.

[104] Michael Garland and Paul S Heckbert. Surface simplification using quadric error metrics. *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, pages 209–216, 1997.

[105] Mohsen Zakeri, Avi Srivastava, Fatemeh Almodaresi, and Rob Patro. Improved data-driven likelihood factorizations for transcript abundance estimation. *Bioinformatics*, 33(14):i142–i151, 2017.

[106] Robert Paige and Robert E Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987.

[107] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

[108] Michael I Love, Charlotte Soneson, and Rob Patro. Swimming downstream: statistical analysis of differential transcript usage following salmon quantification. *F1000Research*, 7, 2018.

[109] Tuuli Lappalainen, Michael Sammeth, Marc R Friedländer, Peter AC't Hoen, Jean Monlong, Manuel A Rivas, Mar Gonzàlez-Porta, Natalja Kurbatova, Thasso Griebel, Pedro G Ferreira, et al. Transcriptome and genome sequencing uncovers functional variation in humans. *Nature*, 2013.

[110] Narayanan Raghupathy, Kwangbom Choi, Matthew J Vincent, Glen L Beane, Keith S Sheppard, Steven C Munger, Ron Korstanje, Fernando Pardo-Manual de Villena, and Gary A Churchill. Hierarchical analysis of RNA-seq reads improves the accuracy of allele-specific expression. *Bioinformatics*, 34 (13):2177–2184, 2018.

[111] Angela N Brooks, Li Yang, Michael O Duff, Kasper D Hansen, Jung W Park, Sandrine Dudoit, Steven E Brenner, and Brenton R Graveley. Conservation of an RNA regulatory map between drosophila and mammals. *Genome Research*, 21(2):193–202, 2011.

[112] Michael I Love, John B Hogenesch, and Rafael A Irizarry. Modeling of RNA-seq fragment sequence bias reduces systematic errors in transcript abundance estimation. *Nature Biotechnology*, 34(12):1287, 2016.

[113] Lauren Gibilisco, Qi Zhou, Shivani Mahajan, and Doris Bachtrog. Alternative splicing within and between drosophila species, sexes, tissues, and developmental stages. *PLoS Genetics*, 12(12), 2016.

[114] Tuuli Lappalainen, Michael Sammeth, Marc R Friedländer, Peter AC't Hoen, Jean Monlong, Manuel A Rivas, Mar Gonzalez-Porta, Natalja Kurbatova, Thasso Griebel, Pedro G Ferreira, et al. Transcriptome and genome sequencing uncovers functional variation in humans. *Nature*, 501(7468):506–511, 2013.

[115] Wei Vivian Li, Anqi Zhao, Shihua Zhang, and Jingyi Jessica Li. Msiq: Joint modeling of multiple rna-seq samples for accurate isoform quantification. *The annals of applied statistics*, 12(1):510, 2018.

[116] Daniel C Jones, Kavitha T Kuppusamy, Nathan J Palpant, Xinxia Peng, Charles E Murry, Hannele Ruohola-Baker, and Walter L Ruzzo. Isolator: accurate and stable analysis of isoform-level expression in rna-seq experiments. *BioRxiv*, page 088765, 2016.

[117] Cong Ma and Carl Kingsford. Detecting, categorizing, and correcting coverage anomalies of rna-seq quantification. *Cell Systems*, 9(6):589–599, 2019.

[118] Lasse Maretty, Jonas Andreas Sibbesen, and Anders Krogh. Bayesian transcriptome assembly. *Genome biology*, 15(10):501, 2014.

[119] Xu Shi, Xiao Wang, Tian-Li Wang, Leena Hilakivi-Clarke, Robert Clarke, and Jianhua Xuan. Sparseiso: a novel bayesian approach to identify alternatively spliced isoforms from rna-seq data. *Bioinformatics*, 34(1):56–63, 2018.

[120] Elsa Bernard, Laurent Jacob, Julien Mairal, and Jean-Philippe Vert. Efficient rna isoform identification and quantification from rna-seq data with network flows. *Bioinformatics*, 30(17):2447–2455, 2014.

[121] Gioele La Manno, Ruslan Soldatov, Amit Zeisel, Emelie Braun, Hannah Hochgerner, Viktor Petukhov, Katja Lidschreiber, Maria E Kastriti, Peter Lönnerberg, Alessandro Furlan, et al. RNA velocity of single cells. *Nature*, 560(7719):494, 2018.