Efficient quantification of bulk and droplet-based single-cell RNA-sequencing data

A Dissertation presented

by

Avi Srivastava

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Department of Computer Science

Stony Brook University

November 25, 2019

Stony Brook University

The Graduate School

Avi Srivastava

We, the dissertation committee for the above candidate for the

Doctor of Philosophy degree, hereby recommend

acceptance of this dissertation.

Rob Patro – Dissertation Advisor
Research Assistant Professor, Department of Computer Science, Stony Brook

Steven Skiena – Chairperson of Defense
Distinguish Teaching professor; SUNY Empire Innovation Professor and Director, AI Institute, Department of Computer Science, Stony Brook

Michael Bender
Professor, Department of Computer Science, Stony Brook

Joshua Rest
Associate Professor, Department of Ecology and Evolution, Stony Brook

This dissertation is accepted by the Graduate School

Eric Wertheimer

Dean of the Graduate School

Abstract of the Dissertation

Efficient quantification of bulk and droplet-based single-cell RNA-sequencing data

by

Avi Srivastava

Doctor of Philosophy

in

Department of Computer Science

Stony Brook University

2019

RNA Sequencing (RNA-seq) technologies are evolving rapidly, and with them, the requirement for fast and accurate tools to analyze the generated data. The first step of many RNA-seq analyses requires us to solve the problem of read-alignment. When reads are aligned to a collection of reference sequences that share a substantial amount of sub-sequence (near or exact repeats), a single read can have many potential alignments, and considering all such alignments can be crucial for downstream analyses, such as quantification. This proposal discusses the evolution of the tools developed to handle the problem of read-alignment for next-generation sequencing, their impact on RNA-seq analysis, introduces a novel concept, quasi-mapping, and an efficient algorithm implementing this approach called `RapMap`, which maps RNA-seq *reads* (sequences) to the reference sequence(s) (transcriptome). By attempting only to report the potential loci of origin of a sequencing read, and not the base-to-base alignment by which it derives from the reference, `RapMap` is capable of *mapping* sequencing reads to a target transcriptome substantially faster than existing alignment tools. The algorithm we employ to implement quasi-mapping uses several efficient data structures and takes advantage of the special structure of shared sequence prevalent in transcriptomes to rapidly provide highly-accurate mapping information.

Over the last decade, single-cell sequencing technologies have gained popularity and, while bulk RNA-sequencing is an established method to perform genome-wide quantification of gene expressions, quantification tools for bulk RNA-seq cannot be directly used for droplet-based single-cell RNA-sequencing (dscRNA-seq) data. We introduce `alevin`, a fast end-to-end pipeline to process droplet-based single-cell RNA sequencing data, performing cell barcode detection, read mapping, unique molecular identifier (UMI) deduplication, gene count estimation, and cell barcode whitelisting. `Alevin` 's approach to UMI deduplication considers transcript-level constraints on the molecules

from which UMIs may have arisen and accounts for both gene-unique reads and reads that multimap between genes. This addresses the inherent bias in existing tools which discard gene-ambiguous reads and improves the accuracy of gene abundance estimates. `Alevin` is considerably faster, typically 8 times than existing gene-quantification approaches, while also using less memory.

Finally, we generalize both methods to make them more robust and extend their capabilities. While quasi-mapping is a useful novel concept, it trades-off some accuracy for speed. In *relatively* complex datasets, the loss in accuracy can be significant. We have updated `RapMap` to employ alignments to improve quantifications in an efficient, dynamic (dataset dependent) way. This optimizes the speed/accuracy tradeoff, making the estimates almost as accurate as end-to-end alignment-based methods, while still using resources similar to `RapMap`. Moreover, we propose a principled approach and study the effect of further extending `alevin`'s framework by optimizing the variational bayesian objective.

"Nature uses only the longest threads to weave her patterns so that each small piece of her fabric reveals the organization of the entire tapestry."

Richard Feynman

Contents

List of Figures

List of Tables

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**setion (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

Acknowledgements

My experience as a Ph.D. student at Stony Brook has been life changing as it has helped me grow both personally and professionally. A big fraction of that is due to my advisor, Rob Patro, whom I'd like to thank for his patience, guidance, and encouragement for letting me explore the field of computational biology. Working with him has been very exciting as he taught me how to do science. His vision and tenaciousness to work and improve RNA-seq quantification has helped me learn how looking at a problem from many different angles can generate exciting new ideas. I also feel that his love for statically-typed programmed programming languages has helped me realize the importance of designing efficient tools which I hope to keep for the rest of my career.

I'd like to thank other members of my committee, Steve Skiena, Michael Bender and Joshua Rest whose important contributions through various lectures, talks, and feedback on this dissertation have been pivotal for my academic development while at Stony Brook University. I'm honored to have them as my committee members.

I'm very grateful to my colleagues and it was an exciting experience to learn and grow alongside them over the years. In particular, Laraib Malik, Hirak sarkar, Mohsen Zakeri, and Fatemeh Almodaresi made Combine-Lab an interesting place to work as they helped me through insightful discussions. I can't imagine working through all the paper deadlines being an enjoyable and pleasant experience without them. I'd also like to thank Mike Love, Charlotte Soneson, Geet Duggal, Richard Smith-Unna, Owen Dando, Darya Filipova, Patrick Marks and Paul Ryvkin for their insightful discussions regarding various aspects of the work.

I can't thank my family enough for their support throughout my journey of learning. Especially my father, Y K Srivastava, who guided me from my childhood to learn actively and motivated me to think out-of-the-box. I'd like to thank my mother Ritima Srivastava, brother Prakhar Srivastava, late grandparents, Govind Prasad Srivastava, Savitri Srivastava, and all my cousins for providing a comfortable and loving environment for me while growing up. And finally, I'd like to thank my partner, Renu, for constantly reminding me there is more to life than my research. Most of all, for standing by my side and providing the much needed emotional support throughout the hardships of grad life.

At last, I would also like to thank Stony Brook Research Computing and Cyberinfrastructure, and the Institute for Advanced Computational Science at Stony Brook University for access to the high-performance SeaWulf computing system, which was made possible by a National Science Foundation grant (#1531492).

Chapter 1

Introduction

1.1 Bulk RNA-seq quantification

DNA sequencing is generally performed in three phases (Schadt, Turner, and Kasarskis, 2010): fragmentation, physical-sequencing, and assembly. The first phase of fragmentation breaks the reference DNA into several small pieces and amplifies it into multiple copies, based on the requirements of the analysis. In the physical sequencing phase, individual units (called *bases*) of the fragments are identified in sequential order to create a read and the number of contiguous bases in a read defines its length measured in base-pairs (bp). In the last phase of assembly, the collection of *reads* (called *library*) is analyzed by bioinformatics software to find overlapping regions in a *library* of *reads* and create "most of" the contiguous sequences of the genome. The phrase "most of" is particularly important because based on the species, sequencing technology, and many external factors, sequencing can face various different biological and computational challenges and can give different results. For example collapsing the repetitive sequence of the genome is a problem that can be solved in several ways, resulting in varying outputs.

The physical-sequencing of the *reads* is generally performed in one of two modes, namely single-end (SE) or paired-end (PE). To give the fragments a notion of direction as shown in Figure 1.1 fragments are sequenced from 5′ end to 3′ end (This refers to the 5′ and 3′ carbons on the sugar present at each end). In SE sequencing, the fragments are sequenced only from one end. But, in PE sequencing, the fragments are first sequenced from one end for some fixed length, then from the opposite end to form a pair of *reads* (called *mate-pairs*). Knowing the distance between mate-pairs of a fragment in PE sequencing helps improve the specificity of the alignment of the fragment, especially when the matching region of the read in the genome is not unique. In this case, the distance between the mate-pairs can help resolve ambiguity.

Sequencing of the Human Genome (Venter et al., 2001) spawned many novel sequencing technologies such as 454 Life sciences (Roche), Illumina (also called Solexa Sequencing), Applied Biosystem's SOLiD, Pacific Biosciences, and Oxford Nanopore, etc. As a result of these sequencing technologies, scientists have found many biological applications

FIGURE 1.1: From top to bottom: Single-End Sequencing, Paired-End Sequencing, information retained after Paired-End sequencing.

for sequencing a genome, such as studying cancer (cancer genomics), gene regulatory network study and differential expression analysis, etc. Most of these biological studies require that we solve the problem of read-alignment before further analysis can be done. Moreover, the accuracy of the downstream analyses greatly depends on how well the problem of read-alignment is solved, which from a computational perspective, makes the problem of read-alignment particularly interesting. The classical problem of read-alignment and its general mathematical notion can be defined as follows:

Given: A set of sequences (called *reads*) R, a reference sequence T, a distance function $d(u, v)$ which gives the distance between two sequence u and v, and ϵ (maximum edit distance) where $\epsilon \in \mathbb{Z}^+$.

$$R = \{r_i : r_i \in \Sigma^{l_i}, \Sigma \in \{A, C, G, T\}, i \in [1, n]\} \tag{1.1}$$

where l_i is the length of the *read* i and n is the total number of *reads*.

$$T = \Sigma^k, \Sigma \in \{A, C, G, T\} \tag{1.2}$$

where k is the length of the reference sequence.

$$d : \Sigma^{|u|} \times \Sigma^{|v|} \to \mathbb{Z} \tag{1.3}$$

Read Alignment Problem: Find $\forall r_i \in R$, a set of tuples $S = \{(s, p, c) : p \in \{[1, k] \cup \emptyset^1\}\}$

[1]Represents no match condition.

such that substring s of T starting from a position p, can be generated from r_i using the transformation defined by c (commonly encoded as *CIGAR string*) where $d(r_i, s) \leq \epsilon$ and c represents an edit script.

Sequencing technology has evolved very quickly and already has three major generations of sequencers. Sanger-sequencing (First Generation sequencing) (Sanger and Coulson, 1975) was developed by Sanger in 1975 and produces *reads* of 1000 bp. Second-generation sequencing (SGS) (Reis-Filho, 2009) (sometimes called Next Generation Sequencing) technologies can produce 35bp-400bp (Schatz, Delcher, and Salzberg, 2010) *reads* with very high throughput and much lower per-base cost than Sanger-sequencing. Third generation sequencing (Schadt, Turner, and Kasarskis, 2010)(TGS) has its advantage of read length which is tens of thousands of bp. TGS suffers from a relatively high error rate and produces comparatively fewer reads than NGS, which is important in certain applications requiring high coverage.

The bioinformatics community has put tremendous efforts into building a wide array of different tools to solve the problem of read-alignment and based on the generation of the sequencer these tools use many different strategies to quickly find potential alignment for the *reads*. Since the field is being explored intensively, it is very hard to discuss all the tools. We have restricted the literature of the report to SGS tools only and discussed them chronologically based on major methodological advancements over previously developed tools in their time.

1.1.1 Smith-Waterman (Smith and Waterman, 1981)

The Smith-Waterman (Smith and Waterman, 1981) algorithm is a classical algorithm for solving the problem of sequence-alignment which allows both mismatches and indels under a given scoring scheme. It guarantees finding optimal alignment under the time-bound of $\boldsymbol{O}(|\boldsymbol{T}|\boldsymbol{l_i})$ for each read r_i (notations as described before Equation (1.1)). For a set R of *reads* this time-bound becomes $\sum_{\boldsymbol{r_i} \in \boldsymbol{R}} \boldsymbol{O}(|\boldsymbol{T}|\boldsymbol{l_i})$ of which most is spent on filling DP sub-solutions that have no hope of satisfying the edit distance bound of ϵ and if we use Smith-Waterman directly on the large reference genomes (like Humans) for read-alignment, it becomes infeasible. The large computational cost of the algorithm and the presence of huge numbers of *reads* motivated scientists to use heuristics (greedy type approach) to solve the problem.

The family of BLAST (Altschul et al., 1990; Altschul et al., 1997) algorithms was one such initiative, initially designed for comparing two long sequences but used also for read-alignment. These algorithms attempt to optimize a specific local similarity measure which uses heuristics to give the similarity $d(u, v)$ between two sequences u and v based on their biological significance. BLAST gives approximate alignment results and was more than an order magnitude faster than previous methods with similar accuracy. BLAST follows a hash table based approach for preprocessing the set of *reads* and a simple seed-and-extend strategy on the reference sequence to solve

the problem of read-alignment. BLAST maintains a database of a hash table on all the k-mers (k length substrings) of the *reads* and traverses the reference sequence to find a hit for seed(s) by checking the database of the hash table (called the *index*). BLAST first joins the seeds without gaps and then refines them by Smith-Waterman for gapped alignment. Generally, in a big reference sequence, only a few subsequences have sufficient similarity to the query; BLAST finds these by filtering irrelevant matches using two threshold parameters S and T. Each seed's similarity score is checked to determine if it's greater or equal to the parameter T. Then, BLAST tries to extend seed(s) exceeding the threshold using the Smith-Waterman algorithm to get the maximum matching score (under their similarity measure) and reports only those alignments exceeding the similarity threshold S.

Even though BLAST approach was faster than existing methods for read-alignment, it has some shortcomings when viewed in the context of NGS. The index needs to be built for every new read, and for alignment, BLAST scans through the entire reference sequence many times (once for each query) resulting in large computational requirements.

1.1.2 MAQ (Li, Ruan, and Durbin, 2008)

MAQ was initially developed for NGS technologies which produced tens of millions of 30-40bp *reads*. With *reads* this short, most genomes contain repetitive regions or nearly repetitive regions (edit distance $< \epsilon$), at least as long as these *reads*. Because of the repetitive regions, *reads* can align to the reference on multiple positions (called ambiguous alignments) and even a few mutations can make the read align to the wrong location. One of the major shortcomings of the read-alignment tools was that for better accuracy a lot of *reads* are ignored based on the ambiguity of their alignment. Although conservatively discarding ambiguous *reads* simplifies the read-alignment problem, it leaves out the important information from repetitive regions, which is essential for many biological applications. Instead of just ignoring ambiguous *reads*, MAQ (Li, Ruan, and Durbin, 2008) was one of the first tools to use this information. MAQ uses an approach similar to phred (Ewing and Green, 1998) base-calling, i.e. not to discard the ambiguous *reads* as soon as they are discovered. Instead, for each ambiguously aligned read, MAQ calculates the likelihood of the read being wrongly positioned and assigns a quality score to each alignment. Using the posterior error probability of each alignment, more information is retained than just discarding them. In a nutshell, MAQ uses a seed-and-extend type approach for aligning *reads*. It builds a hash on *reads*, parses the reference for a hit, but with modification in the alignment strategy of assigning a mapping quality score for each alignment (which is a measure of the confidence that a read actually comes from the position it is aligned to). Also, MAQ doesn't scan the reference sequence for every read. Instead, it builds the index on a set of reads and scans the reference a fixed (small) number of times.

To select a seed, the first 28bp of each read are hashed using six templates (12 for paired-end reads), sorted and stored in the form of a hash-table (called the *index*). Then the subsequences of the reference sequence are scanned for a hit in the hash-table of all six templates. For each hit, MAQ assigns a mapping quality score, which is the sum of the qualities of the mismatching bases over the length of the whole read, extending from the initial 28bp seed without a gap and with at most 2 mismatches. Later, orphaned alignments (mate-pairs where only one mate gets aligned) are searched for gapped alignment using Smith-Waterman in the region of mapped mate-pair. The region for gapped alignments is taken to be two standard deviations of the distribution of the distance between aligned PE *reads*. MAQ was also special in a sense that it utilizes the mate-pair information of paired-end *reads* to correct potentially wrong alignments and accurately align *reads* to repetitive sequences. The idea used by MAQ represented an important advancement but it was still far from satisfactorily solving the problem of read-alignment. For example, MAQ always reports a single alignment and in the case of equally good alignments, it randomly picks one and assigns a score of 0 (identifiable by downstream analysis). Also, no special indexing technique is used for the reference, and even for single-end reads the template hashing ensures only 2 mismatch hits, as every k-mismatch hit requires $\binom{2k}{k}$ templates. Overall, it takes 1100 CPU hours for MAQ to align 100Million 35bp PE *reads*. This is still a high computational cost given the exponentially increasing pace of sequencing.

At the same time, SOAP (Li et al., 2008) was developed to handle the problem of both ungapped and gapped alignment. Unlike MAQ, SOAP hashes the reference sequence into the memory and builds an index table for the reference sequence. It uses 2 bits per base encoding[2] to convert the read and the reference to numeric data-type and to obtain a matching score, a simple XOR of the reference subsequence with the *reads* is done. It allows either a certain number of mismatches or one continuous gap for aligning a read to the reference sequence. As an example, to acquire hits with two mismatches, a read is split into four fragments and with the use of the pigeon-hole principle it made sure that at least two fragments must have no mismatches. By using all six combinations of possible fragments from a read, we can find all the hits with two mismatches. Since mismatches and gaps are not allowed simultaneously, for gapped hits SOAP uses an enumeration algorithm that tries to insert a continuous gap or deletes a fragment at each possible position in a read. In the end, the best hit for each read having a minimal number of mismatches or smaller gap is reported. Moreover, an option for specifying the number of mismatches makes it more versatile. However, tools at this time were facing a dilemma: if an index is built on *reads*, then there is overhead for scanning the whole genome many times when the number of alignments is small, but, if the index is built on the genome, then the memory requirements can become immense.

[2]Since we have only 4 different bases to encode i.e. A, C, T, G

1.1.3 Bowtie (Langmead et al., 2009)

The next significant advancement in the world of read-alignment came with Bowtie (Langmead et al., 2009) which, astonishingly, at the time was able to map 25M *reads* per CPU hour on the human genome using just 1.3G of memory. Bowtie's speed derived largely from using a compressed index of the reference genome. Bowtie index uses the Burrows-Wheeler transform (BWT) (Burrows and Wheeler, 1994) and FM-index (Ferragina and Manzini, 2000; Ferragina and Manzini, 2001). The small size of the resulting index makes it possible to load it into the memory of a computer with as little as 2G of RAM. The main advantage of building the index on the reference instead of *reads* is that the index can be build once and reused many times (e.g. for different sequencing experiments). Bowtie can find exact matches of any length l in $\boldsymbol{O(l)}$ time and can report them all in $\boldsymbol{O(l) + k}$ time where k is number of occurrences, using the "backward search" (Ferragina and Manzini, 2000) algorithm. For the inexact match (with mismatches) problem, bowtie essentially builds inexact alignment by finding multiple supporting exact matches.

Inexact match search (done only when no exact match is found) is performed by using the depth-first search (DFS) on the query and backtracking by replacing and inducing mismatches. Replacement is done using the minimum quality value at that base and ties are broken randomly. An exhaustive search for a read has an exponential time-bound in the number of mismatches m. To avoid full-length backtracking, Bowtie uses a BWT index on both the forward and reversed reference and stops backtracking after reaching the middle of the read which makes backtracking faster. The reverse reference can work well for one mismatch but is not able to avoid excessive backtracking for more than one mismatch. To prevent further backtracking, Bowtie avoids low-quality alignments by stopping backtracking at around 125 backtracks and allowing default mismatch of 2. One of the other advantages of Bowtie was that one can increase the sensitivity of the tool by increasing the default mismatch rate but with the trade-off that more time will be required because DFS will go deeper during the backtracking.

As much as the strategy of Bowtie was novel (though similar approach was taken by SOAP2 (Li et al., 2009)) and vastly improved the state-of-the-art in terms of running time, it started to fall short with the improvements in the sequencing throughput and read-length of the NGS, and also the launch of big research projects required much faster short-read alignment methods to handle the data analysis of large-scale sequencing experiments. With large reads ($\geq$100bp) becoming more common, Bowtie allowing any number of mismatches in the non-seed portion of the read becomes less well-suited to the data. Also, Bowtie does not allow gapped alignment which can be very important in some contexts.

Extending the concept and using the Suffix-Array (SA), BWT and FM-index; BWA (Li and Durbin, 2009) solved the problem of inexact matching by allowing gapped alignment. For exact matching, it simply finds the interval of the suffix array where the query substring occurs (which can be done in linear time using backward search (Ferragina

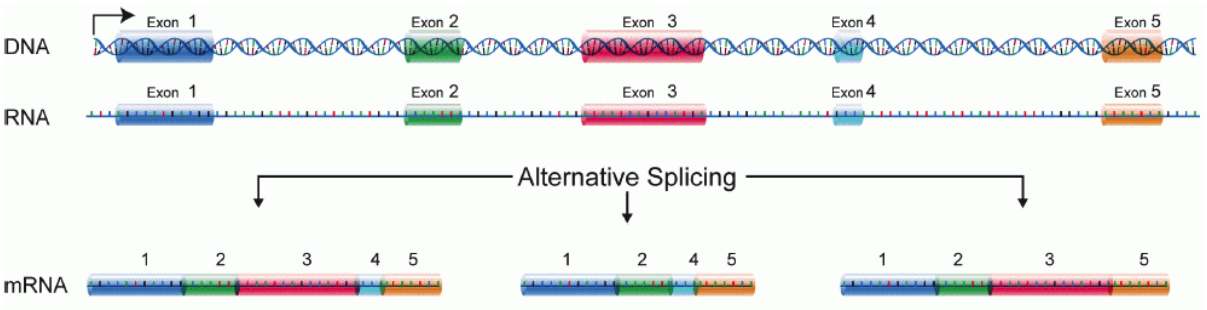


FIGURE 1.2: From Top to bottom (*Alternative Splicing*). : DNA, RNA and mRNA.

and Manzini, 2000)) independent to the length of the reference. For inexact matches, the process is more subtle. Given the maximum no. of indels/mismatches allowed, BWA preprocesses the query string with the BWT of the reverse (not reverse complement) of the reference. This helps BWA to avoid unnecessary backward depth-first traversal resulting in faster and accurate alignment. Many other implementation details like limiting the number of differences in the first few tens of bases of the read were done by BWA to increase the alignment rate and accuracy even further.

1.1.4 BWA-mem (Li, 2013)

In the timeline of read-sequencing, sequencers started producing Terabytes of data and handling them became tedious even for optimized tools like BWA. The need for faster methods resulted in even faster tools like `Bowtie 2`, BWA-mem (Li, 2013) and MrsFast (Hach et al., 2010). `Bowtie 2` improved over Bowtie, by allowing gapped alignment using the seed and extend (difference in extension stage uses the intelligent dynamic program) approach, `Bowtie 2` benefits extensively from using single-instruction-multiple-data (SIMD) parallel processing to vectorize many computations. It is ≥ 3 times faster than BWA. BWA also continued to evolve, and BWA-mem introduced a new seed-chain-extend paradigm. In this approach, one first finds a seed for the alignment using super maximal exact matches (SMEM) based on the method of BWA, and a chain of seeds is formed based on the inter-seed distances. This often avoids useless and costly seed extension step. In the end, alignment is performed using the affine gap penalty dynamic program on the relevant chains. MrsFast is another notable tool that is cache-oblivious and indexed both reference and *reads* using kmer-based approach to provide read alignments.

RNA-seq (Wang, Gerstein, and Snyder, 2009) or transcriptome sequencing produces *reads* from transcribed (exonic regions) regions of the RNA. As explained in Figure 1.2 first double-stranded DNA is converted to a single strand of RNA. Then, to generate mRNA from RNA different coding regions (called *exon*) of the RNA are fused to form different types of mRNA (the process is called *alternative splicing*). In mRNA, the fusion point of two different exonic regions of the RNA is called as *splice junction*. If *reads*

from RNA-seq are aligned to the genomic reference sequence, a read can map to splice junction of mRNA which in the genomic region could be tens of thousands of bases apart and current methods of read-alignment will fail.

This divided the alignment problem into two different regimes namely spliced aligners and unspliced aligners. Spliced read aligners try to solve the problem of aligning to a target reference genome where some *reads* can span exons junctions. These tools are called splice read aligners since they have to find alignments to exon-exon junctions which maybe tens of thousand bases apart in the reference. On the other hand, they typically encounter very little multi-mapping since most *reads* have a single genomic locus of origin. Some of the notable spliced aligners are like TopHat (Trapnell, Pachter, and Salzberg, 2009), `STAR` (Dobin et al., 2013), Subread aligner (Liao, Smyth, and Shi, 2013b) which use novel seed and vote strategy and the recently published HISAT (Kim, Langmead, and Salzberg, 2015) which uses a hierarchical FM index to allow very fast mapping of both DNA-seq and RNA-seq *reads* with only small memory requirements.

TABLE 1.1: Short-Read Aligners/Mappers property. MAQ represents if mapping quality is used by tool or not

Tool	Indexing	Gapped	Paired-end	MAQ
BLAST	hash *reads*	No	No	Yes
MAQ	hash *reads*	Yes	Yes	Yes
SOAP	hash reference	Yes	Yes	Yes
Bowtie	FM-index	No	Yes	No
BWA	FM-index	Yes	Yes	No
BWA-mem	FM-index	Yes	Yes	No
pseudo-mapping	T-DBG	Yes	Yes	No
quasi-mapping	Suffix Array	Yes	Yes	No

Transcriptome aligners are same as unspliced aligners (i.e. they work on transcriptome level) but they also intelligently handle high multi-mapping rate in the alignments due to the presense of isoforms of the same exonic regions in mRNA. As can be seen in Figure 1.2, *exon1* (in blue) is present in all the mRNA and if a read is generated from the region of exon1 from any one mRNA it'll get aligned to each of them equally well. The modified problem of read-alignment for transcriptome aligners can be defined as follows:

Given: A set of sequences (called *reads* or RNA-seq *reads*) R, a set (called transcriptome) T of reference sequences (called transcripts) t, a distance function $d(u, v)$ which gives the distance between two sequence u and v, and ϵ (maximum edit distance) where $\epsilon \in \mathbb{Z}^+$.

$$R = \{r_i : r_i \in \Sigma^{l_i}, \Sigma \in \{A, C, G, T\}, i \in [1, n]\} \tag{1.4}$$

where l_i is the length of the *read* i and n is the total number of *reads*.

$$T = \{t_i : t_i \in \Sigma^{\ell_{t_i}}, \Sigma \in \{A, C, G, T\}, i \in [1, k]\} \tag{1.5}$$

where ℓ_{t_i} is the length of the transcript i and k is the total number of transcripts.

$$d : \Sigma^{|u|} \times \Sigma^{|v|} \to \mathbb{Z} \tag{1.6}$$

Read Alignment Problem for Transcriptome: Find $\forall r_i \in R$, a set of tuple $S = \{(s, p, t, c) : p \in \{[1, k] \cup \emptyset^{3}\}\}$ such that substring s of transcript t starting from a position p, can be generated from r_i using the transformation defined by c (also called *CIGAR string*) where $d(r_i, s) \leq \epsilon$ and c represents a collection of match, mismatch, insertion and deletion (indel).

Transcriptome aligners focus primarily, on a different problem from spliced-aligners (though some of the spliced aligners can be used as transcriptome aligner) since they have to align *reads* to the transcriptome instead of the genome. Specifically, they face a huge rate of multi-mapping in *reads* but they don't have to handle spliced events (since they assume a reference that consists of the already spliced (i.e. mature) transcripts). Transcriptome alignment can also be used after *de novo* transcriptome assembly and does not rely on the knowledge of the reference genome of the organism. Table 1.1 (Li and Homer, 2010) gives a gist of some of the tools and their constituent properties.

1.1.5 Salmon (Patro et al., 2017)

`Salmon` solved an important problem of quantification by using a novel concept of lightweight-alignemnt (Patro et al., 2017). The main motivation behind lightweight alignment was the realization that in some special biological application (in the case of `Salmon` it was the quantification of transcript abundances from RNA-seq *reads*), we do not require the actual alignment between the query sequence (i.e. c in our problem formulation) and the reference. Rather, simply knowing the transcripts (and the position in that transcript) where each match reasonably well is sufficient to solve the problem. `Salmon` attempts to find these mapping locations using BWA-mem type seed-chain-extend approach i.e. seeding through SMEM, chaining MEM and SMEM together and extending it to only exact matches.

The introduction of lightweight-alignment further divided the transcriptome aligners into two different regimes, namely aligners and mappers. Aligners (as discussed in Section 1.1.4) work by generating a CIGAR string which is a base-to-base correspondence

[3] Represents no match condition.

between two query strings. But, in comparison to alignment, mappers do not find CIGAR strings. Instead for each read r_i, they just report transcript t and the position p in t where r_i gets aligned "sufficiently well" (*Alignment vs. Mapping*)[4].

Using a different, alignment-free approach, a recent quantification tool `Kallisto` (Bray et al., 2016) makes use of Transcriptome-de-Bruijn graph(T-DBG) to compute called pseudo-alignments. `Kallisto` builds a de-Bruijn graph on the transcriptome and labels each unique k-mer region on the graph with its constituent transcripts. During the pseudo-alignment stage, a procedure is used to extract a subset of k-mers from the read and intersect them with the T-DBG to determine a "compatible" set of transcripts. In some sense, pseudo-alignment goes even further than mapping in that the pseudo-alignments themselves give no information about where (position) or how (orientation) a read maps to a transcript (though pseudo-alignments can be pre-processed, at some extra cost to approximate this extra information). The field of read-mapping is under constant development and we have developed a novel mapping technique, called quasi-mapping, using the suffix array, which is discussed in chapter 2.

1.2 Single Cell RNA-seq quantification

Bulk RNA-sequencing is an established method to perform genome-wide quantification of gene expressions (Mortazavi et al., 2008). However, bulk experiments average-out the expression pattern of an individual cell (or a cell type) across the full experiment – typically millions of cells, losing cell-level heterogeneity which is crucial, for example, to understand the Waddington landscape of an epigenomic analysis (Goldberg, Allis, and Bernstein, 2007). Shortly after bulk RNA-seq experiments became common, Tang et al. (2009) proposed single-cell RNA-seq (scRNA-seq) to perform a transcriptome-wide study of single cells. Even though it was restricted to investigating a few interesting cells, it later allowed exciting biological and medical insights such as characterizing cellular states and molecular circuitries (Consortium, 2012). Since then, many single-cell protocols have been proposed which improve many aspects of the technology; for example in Figure 1.3 (Svensson, Vento-Tormo, and Teichmann, 2018) show the exponential scaling of single-cell RNA-seq in the past decade, enabling the characterization of transcriptional profile for thousands or even millions of single cells.

In general, single-cell protocols follow similar library preparation steps. As shown in Figure 1.4 and discussed by (Ziegenhain et al., 2017), it requires the isolation and lysis of single cells, the conversion of their RNA into cDNA, and the amplification of cDNA to generate high-throughput sequencing libraries. However, along with the monetary cost, the effectiveness of single cell protocols is determined by a number of diverse variables like technical factors such as the rate of cell isolation (i.e. the fraction of cells in the population which is picked up), sensitivity to capture mRNA

[4]Since the problem is less well-studied than alignment and a reasonable determination of what constitutes "sufficiently well" is still a reasonable topic for discussion and debate

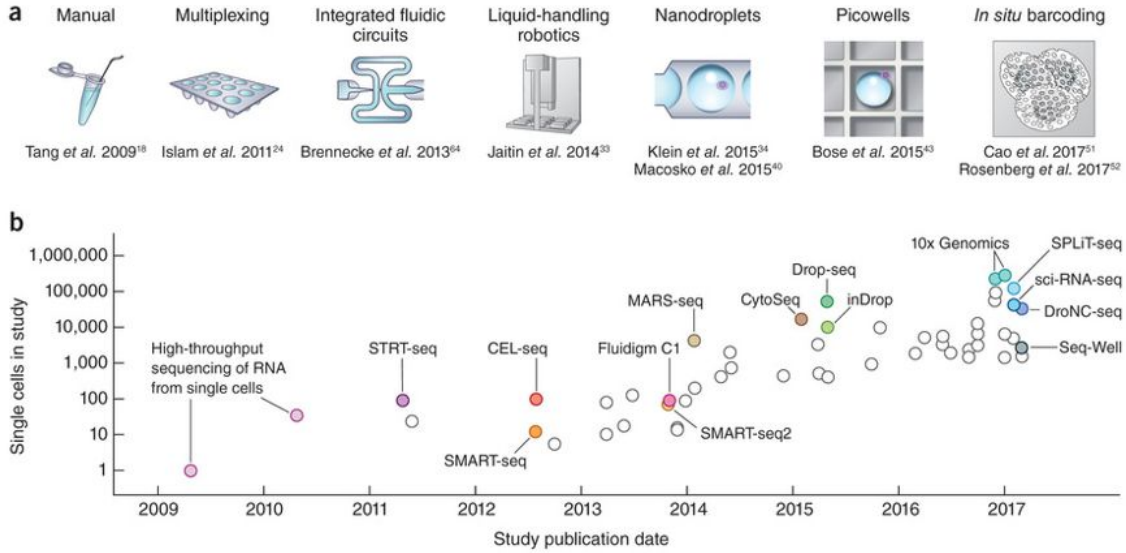


FIGURE 1.3: Scaling of scRNA-seq experiments (Svensson, Vento-Tormo, and Teichmann, 2018)

molecule in a cell and conversion to cDNA molecule, or the rate and the accuracy of the amplification procedure used by the single cell technology. Although the technical possibilities are vast, the main limiting factors for single-cell technologies include financial considerations and the larger amount of biological material required to start an experiment (Vieth et al., 2017; Ziegenhain et al., 2017). These factors limit the statistical power of an assessment and motivate more involved method development to process the data generated through these single-cell protocols.

1.2.1 Fluidigm based platforms (Islam et al., 2012; Hashimshony et al., 2012)

One of the earliest techniques to capture and isolate single cells was based on microliter volume plates (microwell plates) (Islam et al., 2012; Hashimshony et al., 2012). Later it was automated through the microfluidic platform from Fluidigm (C1 platform) (Islam et al., 2014), typically in very small volumes of a few nanolitres. The platform uses an integrated fluidic circuit (IFC) array where a batch of cells is loaded through pipetting. The microfluidics system in the platforms separates the cells into different chambers which are called as wells. The earlier C1 platform uses a 96 well plate but more recently an 800 well plate has also been available by Fluidigm. The C1 platform after capturing individual cells automatically washes and lysis, perform reverse transcription and PCR amplify the collected cDNA, which is extracted and sequenced through Illumina sequencing.

The C1 platform has provided a range of opportunities for many experiments but has

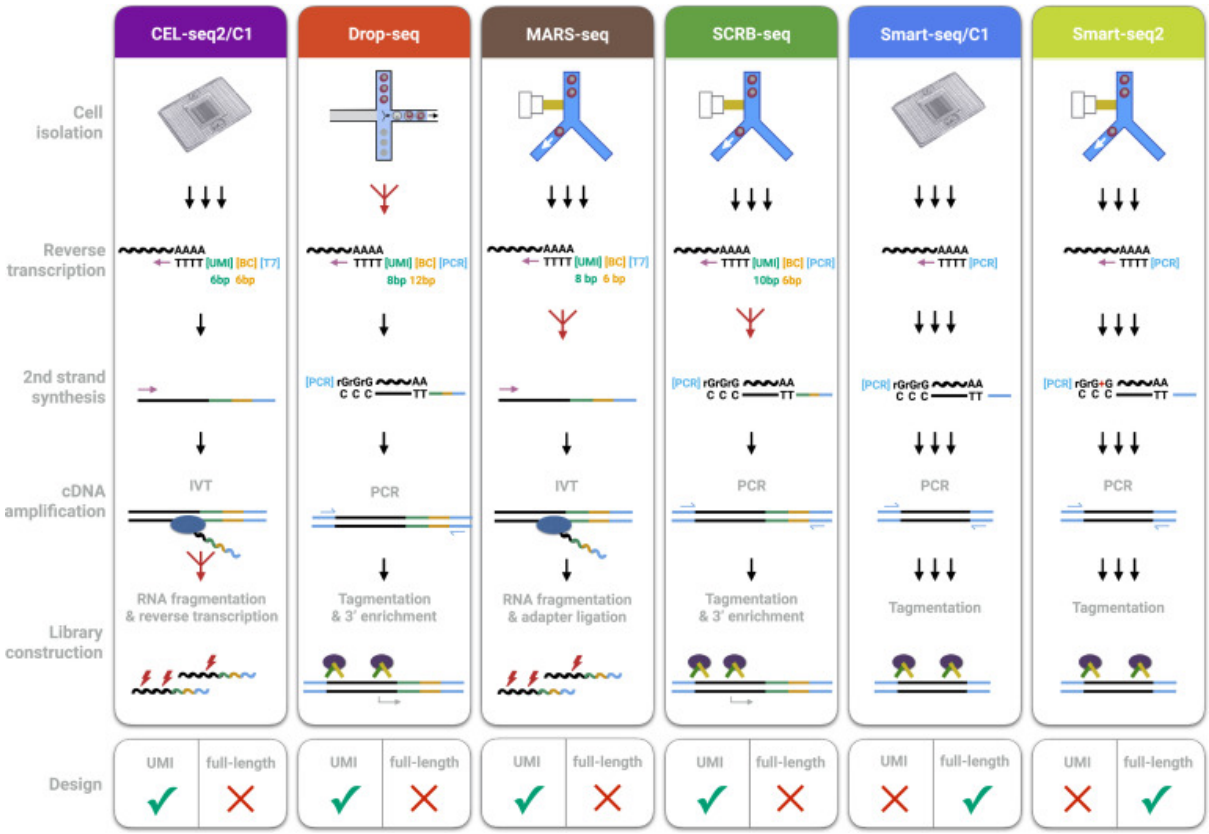


FIGURE 1.4: Schematic Overview of Library Preparation Steps (Ziegenhain et al., 2017)

its limitations. Some of the caveats associated with Fluidigm based experiments include relatively low throughput, in expectation higher per-cell cost, and the size of ICF chip as it captures cells of a particular size in a single run. In contrast to plate-based capture methods, which usually provide reads along the full length of a molecule, an alternative droplet-based capture method (Macosko et al., 2015; Klein et al., 2015; Zheng et al., 2017) was proposed which trades-off full-length sequencing with higher throughput resulting in lower per cell cost. In this study, we are going to focus primarily on droplet-based single-cell RNA-sequencing (dscRNA-seq).

1.2.2 Droplet based platforms (Macosko et al., 2015; Klein et al., 2015; Zheng et al., 2017)

Droplet-based platforms work by capturing cells in nanoliter-sized aqueous droplets by associating a different barcode with each cell's RNAs and sequencing them all together. Initially, droplet-based platforms were released as DIY systems but later automated and commercialized by 10x Genomics (Zheng et al., 2017) and their chromium platform. As discussed by Zappia (2019), all three platforms work in a similar fashion apart from how the beads are produced, when the droplets are broken and in some aspects of their chemistry. In Figure 1.5 we show how the molecular barcoding is performed in one of the dscRNA-seq platforms as proposed by (Macosko et al., 2015). In the drop-seq platform, individual cells are dissociated from complex tissue and along with a flow of beads, suspended in lysis buffer, are encapsulated into nanoliter-sized emulsion droplets. Inside the droplet, the cells are lysed and its mRNAs bind to the primers on the droplet's companion microparticle which is reverse-transcribed into cDNAs generating a set of beads. The droplets later break generating a library for all cells in parallel in one single tube which is amplified in pools for high-throughput Illumina sequencing.

On each bead, oligo-dT primers have a covalently bounded UMI (Unique Molecule Identifier) and a unique, bead-specific cellular barcode (CB). Association of each individual molecule with a cell-unique UMI sequence before PCR amplification makes the platform particularly effective as only a small section at the 3′ end of each molecule is sequenced. This has a monetary benefit of reducing the amount of cDNA that needs to be sequenced and therefore increases the number of cells that can be sequenced at a time, at a similar cost. However, the trade-off of the coverage with throughput comes at a cost of making this platform unsuitable for applications such as variant detection and de-novo assembly. Moreover, traditional pipelines for bulk RNA-seq quantification also cannot be directly used for dscRNA-seq quantification as datasets with CB/UMI need extra processing steps like CB whitelisting (the process of disambiguating droplets with real cells from empty/doublet containing droplets), UMI deduplication (the process of modeling PCR tree to deduplicate UMI sequences) which are non-trivial to model efficiently.

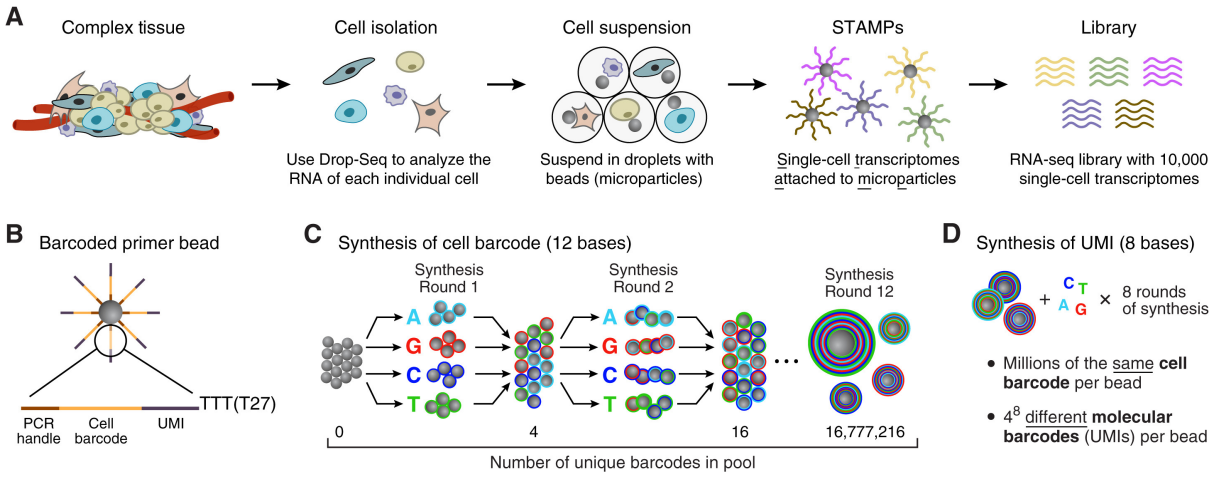


FIGURE 1.5: Molecular Barcoding of Cellular Transcriptomes in Droplets (Macosko et al., 2015)

A typical dscRNA-seq experiment contains a very small amount of RNA ($\sim$ 10–30 pg) of which less than 5 percent of which is mRNA (Zappia, 2019). The tiny sample size makes it difficult to process and necessitates a PCR amplification step, however, the rate of amplification of each molecule in each cell is not controllable and creates an irregularity in the expression of the transcript and gene abundances. Along with CB whitelisting, a basic model of PCR tree for deduplication has been previously applied by various protocols (Macosko et al., 2015; Zheng et al., 2017). In Chapter 4, we propose a unified pipeline called `alevin` where we describe an end-to-end quantification pipeline that takes as input sample-demultiplexed `FASTQ` files and outputs gene-level UMI counts for each cell in the library.

Chapter 2

RapMap (Srivastava et al., 2016b)

2.1 Background

The bioinformatics community has put tremendous effort into building a wide array of different tools to solve the read-alignment problem efficiently. These tools use many different strategies to quickly find potential alignment locations for reads; for example, Bowtie (Langmead et al., 2009), `Bowtie 2` (Langmead and Salzberg, 2012), BWA (Li and Durbin, 2009) and BWA-mem (Li, 2013) use variants of the FM-index, while tools like the Subread aligner (Liao, Smyth, and Shi, 2013b), Maq (Li, Ruan, and Durbin, 2008) and MrsFast (Hach et al., 2010) use k-mer-based indices to help align reads efficiently. Because read alignment is such a ubiquitous task, the goal of such tools is often to provide accurate results as quickly as possible. Indeed, recent alignment tools like `STAR` (Dobin et al., 2013) demonstrate that rapid alignment of sequenced reads is possible, and tools like HISAT (Kim, Langmead, and Salzberg, 2015) demonstrate that this speed can be achieved with only moderate memory usage. When reads are aligned to a collection of reference sequences that share a substantial amount of sub-sequence (near or exact repeats), a single read can have many potential alignments, and considering all such alignment can be crucial for downstream analysis (e.g. considering all alignment locations for a read within a transcriptome for the purpose of quantification (Li and Dewey, 2011), or when attempting to cluster *de novo* assembled contigs by shared multi-mapping reads (Davidson and Oshlack, 2014)). However, reporting multiple potential alignments for each read is a difficult task, and tends to substantially slow down even very efficient alignment tools.

Yet, in many cases, all of the information provided by the alignments is not necessary. For example, in the transcript analysis tasks mentioned above, simply the knowledge of the transcripts and positions to which a given read maps well is sufficient to answer the questions being posed. In support of such "analysis-oriented" computation, we propose a novel concept, called quasi-mapping, and an efficient algorithm implementing quasi-mapping (exposed in the software tool `RapMap`) to solve the problem of mapping sequenced reads to a target transcriptome. This algorithm is *considerably* faster than state-of-the-art aligners, and achieves its impressive speed by exploiting the structure of the transcriptome (without requiring an annotation), and eliding the computation

of full-alignments (e.g. `CIGAR` strings). Further, our algorithm produces mappings that meet or exceed the accuracy of existing popular aligners under different metrics of accuracy. Finally, we demonstrate how the mappings produced by `RapMap` can be used in the downstream analysis task of transcript-level quantification from RNA-seq data, by modifying the Sailfish (Patro, Mount, and Kingsford, 2014) tool to take advantage of quasi-mappings, as opposed to raw k-mer counts, for transcript quantification. We also demonstrate how quasi-mappings can be used to effectively cluster contigs from *de novo* assemblies. We show that the resulting clusterings are of comparable or superior accuracy to those produced by recent methods such as CORSET (Davidson and Oshlack, 2014), but that they can be computed *much* more quickly using quasi-mapping.

2.2 Materials and Methods

The quasi-mapping concept, implemented in the tool `RapMap`, is a new mapping technique to allow the rapid and accurate mapping of sequenced fragments (single or paired-end reads) to a target transcriptome. `RapMap` exploits a combination of data structures — a hash table, suffix array (SA), and efficient rank data structure. It takes into account the special structure present in transcriptomic references, as exposed by the suffix array, to enable ultra-fast and accurate determination of the likely loci of origin of a sequencing read. Rather than a standard alignment, quasi-mapping produces what we refer to as fragment *mapping* information. In particular, it provides, for each query (fragment), the reference sequences (transcripts), strand and position from which the query may have likely originated. In many cases, this mapping information is sufficient for downstream analysis. For example, tasks like transcript quantification, clustering of *de novo* assembled transcripts, and filtering of potential target transcripts can be accomplished with this mapping information. However, this method does not compute the base-to-base alignment between the query and reference. Thus, such mappings may not be appropriate in every situation in which alignments are currently used (e.g. variant detection).

We note here that the concept of quasi-mapping shares certain motivations with the notions of lightweight-alignment (Patro et al., 2017) and pseudo-alignment (Bray et al., 2016). Yet, all three concepts — and the algorithms and data structures used to implement them — are distinct and, in places, substantially different. Lightweight-alignment scores potential matches based on approximately consistent chains of super-maximal exact matches shared between the query and targets. Therefore, it typically requires some more computation than the other methods, but allows the reporting of a score with each returned mapping and a more flexible notion of matching. Pseudo-alignment, as implemented in `Kallisto`, refers only to the process of finding *compatible* targets for reads by determining approximately matching paths in a colored De Bruijn graph of a pre-specified order. Among compatible targets, extra information concerning the mapping (e.g. position and orientation) can be extracted *post-hoc*, but this requires extra processing, and the resulting mapping is no longer technically a pseudo-alignment.

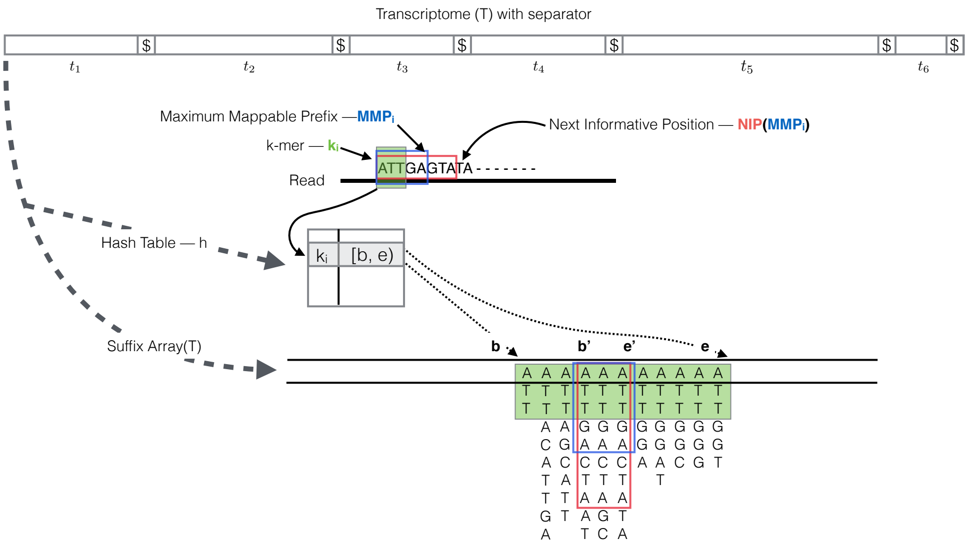


FIGURE 2.1: The transcriptome (consisting of transcripts $t_1, \ldots, t_6$) is converted into a \$-separated string, T, upon which a suffix array, $\texttt{SA}[T]$, and a hash table, h, are constructed. The mapping operation begins with a k-mer (here, $k = 3$) mapping to an interval $[b, e)$ in $\texttt{SA}[T]$. Given this interval and the read, MMP_i and $\texttt{NIP}(\text{MMP}_i)$ are calculated as described in section 2.2. The search for the next hashable k-mer begins k bases before $\texttt{NIP}(\text{MMP}_i)$.

Quasi-mapping seeks to find the *best* mappings (targets and positions) for each read, and does so (approximately) by finding minimal collections of dynamically-sized, right-maximal matching contexts between target and query positions. The algorithm for quasi-mapping that we describe below achieves this using a combination of a k-mer lookup table and a generalized suffix array. While each of these approaches provide some insight into the problems of alignment and mapping, they represent distinct concepts and exhibit unique characteristics in terms of speed and accuracy, as demonstrated below[1].

2.2.1 An algorithm for Quasi-mapping

The algorithm we use for quasi-mapping makes use of two main data structures, the generalized suffix array (Manber and Myers, 1993) $\texttt{SA}[T]$ of the transcriptome T, and a hash table h mapping each k-mer occurring in T to its suffix array interval (by default $k = 31$). Additionally, we must maintain the original text T upon which the suffix array was constructed, and the name and length of each of the original transcript sequences. T consists of a string in which all transcript sequences are joined together with a special separator character. Rather than designating a separate terminator $\$_i$ for each reference

[1]We do not compare against lightweight-alignment here, as no stand-alone implementation of this approach is currently available

sequence in the transcriptome, we make use of a single separator \$, and maintain an auxiliary rank data structure which allows us to map from an arbitrary position in the concatenated text to the index of the reference transcript in which it appears. We use the rank9b algorithm and data structure of Vigna (2008) to perform the rank operation quickly.

Quasi-mapping determines the mapping locations for a query read r through repeated application of (1) determining the next hash table k-mer that starts past the current query position, (2) computing the maximum mappable prefix (MMP) of the query beginning with this k-mer, and then (3) determining the next informative position (NIP) by performing a longest common prefix (LCP) query on two specifically chosen suffixes in the suffix array.

The algorithm begins by hashing the k-mers of r, from left-to-right (a symmetric procedure can be used for mapping the reverse-complement of a read), until some k-mer k_i — the k-mer starting at position i within the read — is present in h and maps to a valid suffix array interval. We denote this interval as $\texttt{I}(k_i) = [b, e)$. Because of the lexicographic order of the suffixes in the suffix array, we immediately know that this k-mer is a prefix of all of the suffixes appearing in the given interval. However, it may be possible to extend this match to some longer substring of the read beginning with k_i. In fact, the longest substring of the read that appears in the reference and is prefixed by k_i is exactly the maximum mappable prefix (MMP) (Dobin et al., 2013) of the suffix of the read beginning with k_i. We call this maximum mappable prefix MMP_i, and note that it can be found using a slight variant of the standard suffix array binary search (Manber and Myers, 1993) algorithm. For speed and simplicity, we implement the "simple accelerant" binary search variant of Gusfield (1997). Since we know that any substring that begins with k_i must reside in the interval $[b, e)$, we can restrict the MMP_i search to this region of the suffix array, which is typically very small.

After determining the length of MMP_i within the read, one could begin the search for the next mappable suffix array interval at the position following this MMP. However, though the current substring of the read will differ from all of the reference sequence suffixes at the base following MMP_i, the suffixes occurring at the lower and upper bounds of the suffix array interval corresponding to MMP_i may not differ from each other (See Figure 2.1). That is, if $\texttt{I}(\text{MMP}_i) = [b', e')$ is the suffix array interval corresponding to MMP_i, it is possible that $|\texttt{LCP}(\texttt{T}[\texttt{SA}[b']], \texttt{T}[\texttt{SA}[e'-1]]))| > |\text{MMP}_i|$. In this case, it is most likely that the read and the reference sequence bases following MMP_i disagree as the result of a sequencing error, not because the (long) MMP discovered between the read and reference is a spurious match. Thus, beginning the search for the next MMP at the subsequent base in the read may not be productive, since the matches for this substring of the query may not be informative — that is, such a search will likely return the same (relative) positions and set of transcripts. To avoid querying for such substrings, we define and make use of the notion of the next informative position (NIP). For a maximum mappable prefix MMP_i, with $\texttt{I}(\text{MMP}_i) = [b', e')$, we define $\texttt{NIP}(\text{MMP}_i) = |\texttt{LCP}(\texttt{T}[\texttt{SA}[b']], \texttt{T}[\texttt{SA}[e'-1]]))| + 1$. Intuitively, the next informative

position of prefix MMP_i is designed to return the next position in the query string where a suffix array search is likely to yield a set of transcripts different from those contained in $\texttt{I}(\text{MMP}_i)$. To compute the longest common prefix between two suffixes when searching for the NIP, we use the "direct min" algorithm of Ilie, Navarro, and Tinta (2010). We found this to be the fastest approach. Additionally, it doesn't require the maintenance of an LCP array or other auxiliary tables aside from the standard suffix array.

Given the definitions we have explained above, we can summarize the quasi-mapping procedure as follows (an illustration of the mapping procedure is provided in Figure 2.1). First, a read is scanned from left to right (a symmetric procedure can be used for mapping the reverse-complement of a read) until a k-mer k_i is encountered that appears in h. A lookup in h returns the suffix array interval $\texttt{I}(k_i)$ corresponding to the substring of the read consisting of this k-mer. Then, the procedure described above is used to compute MMP_i and $\ell = \texttt{NIP}(\text{MMP}_i)$. The search procedure then advances to position $i + \ell - k$ in the read, and again begins hashing the k-mers it encounters. This process of determining the MMP and NIP of each processed k-mer and advancing to the next informative position in the read continues until the next informative position exceeds position $l_r - k$ where l_r is the length of the read r. The result of applying this procedure to a read is a set $S = \{(q_0, o_0, [b_0, e_0)), (q_1, o_1, [b_1, e_1)), \dots\}$ of query positions, MMP orientations, and suffix array intervals, with one such triplet corresponding to each MMP.

The final set of mappings is determined by a consensus mechanism. Specifically, the algorithm reports the intersection of transcripts appearing in all hits — i.e. the set of transcripts that appear (in a consistent orientation) in every suffix array interval appearing in S. These transcripts, and the corresponding strand and location on each, are reported as *quasi-mappings* of this read. These mappings are reported in a `samtools`-compatible format in which the relevant information (e.g. target id, position, strand, pair status) is computed from the mapping. We note that alternative consensus mechanisms, both more and less stringent, are easy to enforce given the information contained in the hits (e.g. enforcing that the hits are co-linear with respect to both the query and reference). However, below, we consider this simple consensus mechanism.

Intuitively, `RapMap`'s combination of speed and accuracy result from the manner in which it exploits the nature of exactly repeated sequence that is prevalent in transcriptomes (either as a result of alternative splicing or paralogous genes). In addition to efficient search for MMPs and NIPs, the suffix array allows `RapMap` to encode exact matches between the query and many potential transcripts very efficiently (in the form of "hits"). This is because all reference locations for a given MMP appear in consecutive entries of the suffix array, and can be encoded efficiently by simply recording the suffix array interval corresponding to this MMP. By aggressively filtering the hits to determine the set of "best" matching transcripts and positions, `RapMap` is able to quickly discard small matches that are unlikely to correspond to a correct mapping. Similarly, the large collection of exact matches that appear in the reported mapping are very likely

to appear in the alignment mapping (were the actual alignments to be computed). In some sense, the success of the strategy adopted by `RapMap` further validates the claim of Liao, Smyth, and Shi (2013b) that the seed-and-vote paradigm can be considerably more efficient than the seed-and-extend paradigm, as `RapMap` adopts neither of these paradigms directly, but its approach is more similar to the former than the latter.

In the next section, we analyze how this algorithm for quasi-mapping, as described above, compares to other aligners in terms of speed and mapping accuracy.

2.3 Results

To test the practical performance of quasi-mapping, we compared `RapMap` against a number of existing tools, and analyzed both the speed and accuracy of these tools on synthetic and experimental data. Benchmarking was performed against the popular aligners `Bowtie 2` (Langmead and Salzberg, 2012) (v2.2.6) and `STAR` (Dobin et al., 2013) (v2.5.0c) and the recently-introduced pseudo-alignment procedure used in the quantification tool `Kallisto` (Bray et al., 2016) (v0.42.4). All experiments were scripted using Snakemake (Köster and Rahmann, 2012) and performed on a 64-bit linux server with 256GB of RAM and 4 x 6-core Intel Xeon E5-4607 v2 CPUs running at 2.60GHz. Wall-clock time was recorded using the `time` command.

In our testing we find that `Bowtie 2` generally performs well in terms of reporting the true read origin among its set of multi-mapping locations. However, it takes considerably longer and tends to return a larger set of multi-mapping locations than the other methods. In comparison to `Bowtie 2`, `STAR` is *substantially* faster but somewhat less accurate. `RapMap` achieves accuracy comparable or superior to `Bowtie 2`, while simultaneously being much faster than even `STAR`. `Kallisto` is similar to (slightly slower than) `RapMap` in terms of single-threaded speed, and exhibits accuracy very similar to that of `STAR`. For both `RapMap` and `Kallisto`, simply writing the output to disk tends to dominate the time required for large input files with significant multi-mapping (though we eliminate this overhead when benchmarking). This is due, in part, to the verbosity of the standard `SAM` format in which results are reported, and suggests that it may be worth developing a more efficient and succinct output format for mapping information.

2.3.1 Speed and accuracy on synthetic data

To test the accuracy of different mapping and alignment tools in a scenario where we know the true origin of each read, we generated data using the Flux Simulator (Griebel et al., 2012). This synthetic dataset was generated for the human transcriptome from an annotation taken from the ENSEMBL (Cunningham et al., 2015) database consisting of $86,090$ transcripts corresponding to protein-coding genes. The dataset consists of

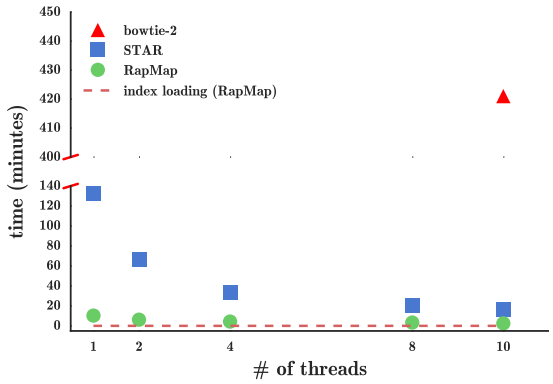


FIGURE 2.2: The time taken by `Bowtie 2`, `STAR` and `RapMap` to process the synthetic data using varying numbers of threads. `RapMap` processes the data substantially faster than the other tools, while providing results of comparable or better accuracy.

$\sim$ 48 million 76 base pair, paired-end reads. The detailed parameters used for the Flux Simulator can be found in Appendix A.2.

When benchmarking these methods, reads were aligned directly to the transcriptome, rather than to the genome. This was done because we wish to benchmark the tools in a manner that is applicable when the reference genome may not even be known (e.g. in *de novo* transcriptomics). The parameters of `STAR` (see Appendix A.1) were adjusted appropriately for this purpose (e.g. to dis-allow introns etc.). Similarly, `Bowtie 2` was also used to align reads directly to the target transcriptome; the parameters for `Bowtie 2` are given in Appendix A.1.

Mapping speed

We wish to measure, as directly as possible, just the time required by the mapping algorithms of the different tools. Thus, when benchmarking the runtime of different methods, we do not save the resulting alignments to disk. Further, to mitigate the effect of "outliers" (a small number of reads which map to a very large number of low-complexity reference positions), we bound the number of different transcripts to which a read can map to be 200.

Additionally, we have also benchmarked `Kallisto`, but have not included the results in Figure 2.2, as the software, unlike the other methods, does not allow multi-threaded execution if mappings are being reported. Thus, we ran `Kallisto` with a single thread, using the `-pseudobam` flag and redirecting output to `/dev/null` to avoid disk overhead. `Kallisto` requires 17.87m to map the 48M simulated reads, which included <1m of quantification time. By comparison, `RapMap` required 11.65m to complete with a single thread.

TABLE 2.1: Accuracy of aligners/mappers under different metrics

	Bowtie 2	Kallisto	RapMap	STAR
reads aligned	47579567	44804857	47613536	44711604
recall	97.41	91.60	97.49	91.35
precision	98.31	97.72	98.48	97.02
F1-score	97.86	94.56	97.98	94.10
FDR	1.69	2.28	1.52	2.98
hits per read	5.98	5.30	4.30	3.80

Finally, we note Kallisto, STAR and RapMap require 2-3× the memory of Bowtie 2, but all of the methods tested here exhibit reasonable memory usage. The synthetic set of 48 million reads can be mapped to an index of the entire human transcriptome on a typical laptop with 8 GB of RAM.

As Figure 2.2 illustrates, RapMap out-performs both Bowtie 2 and STAR in terms of speed by a substantial margin, and finishes mapping the reads with a single thread faster than STAR and Bowtie 2 with 10 threads. We consider varying the number of threads used by RapMap and STAR to demonstrate how performance scales with the number of threads provided. On this dataset, RapMap quickly approaches peak performance after using only a few threads. We believe that this is not due to limits on the scalability of RapMap, but rather because the process is so quick that, for a dataset of this size, simply reading the index constitutes a large (and growing) fraction of the total runtime (dotted line) as the number of threads is increased. Thus, we believe that the difference in runtime between RapMap and the other methods may be even larger for datasets consisting of a very large number of reads, where the disk can reach peak efficiency and the multi-threaded input parser (we use the parser from the Jellyfish (Marçais and Kingsford, 2011) library) can provide input to RapMap quickly enough to make use of a larger number of threads. Since running Bowtie 2 with each potential number of threads on this dataset is very time-consuming, we only consider Bowtie 2's runtime using 10 threads.

Mapping accuracy

Since the Flux Simulator records the true origin of each read, we make use of this information as ground truth data to assess the accuracy of different methods. However, since a single read may have multiple, equally-good alignments with respect to the transcriptome, care must be taken in defining accuracy-related terms appropriately. A read is said to be correctly mapped by a method (a true positive) if the set of transcripts reported by the mapper for this read contains the true transcript. A read is said to be incorrectly mapped by a method (a false positive) if it is mapped to some set of 1 or more transcripts, none of which are the true transcript of origin. Finally, a read is considered to be incorrectly un-mapped by a method (a false negative) if the method reports no

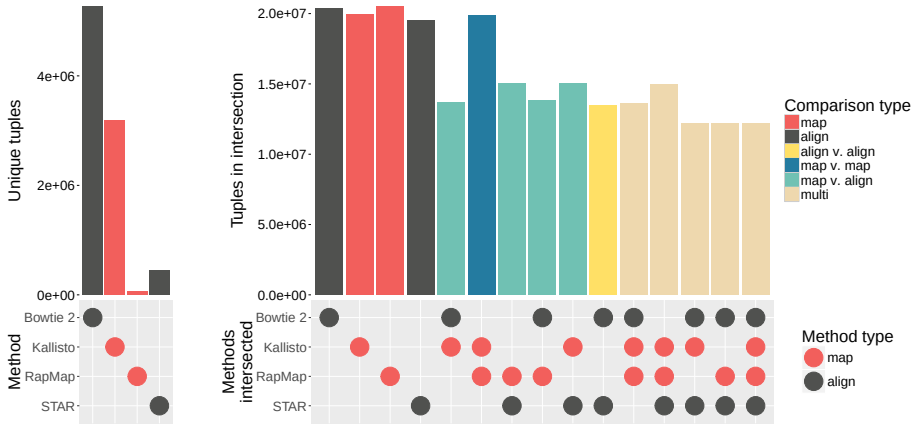


FIGURE 2.3: Mapping agreement between subsets of `Bowtie 2`, `STAR`, `Kallisto` and `RapMap`.

mappings, but the transcript of origin is in the reference. Given these definitions, we report precision, recall, F1-Score and false discovery rate (FDR) in Table 2.1 using the standard definitions of these metrics. Additionally, we report the average number of "hits-per-read" (hpr) returned by each of the methods. Ideally, we want a method to return the smallest set of mappings that contains the true read origin. However, under the chosen definition of a true positive mapping, the number of reported mappings is not taken into account, and a result is considered a true positive so long as it contains the actual transcript of origin. The hpr metric allows one to assess how many *extra* mappings, on average, are reported by a particular method.

As expected, `Bowtie 2`— perhaps the most common method of directly mapping reads to transcriptomes — performs very well in terms of precision and recall. However, we find that `RapMap` yields very similar (in fact, slightly better) precision and recall. `STAR` and `Kallisto` obtain similar precision to `Bowtie 2` and `RapMap`, but have lower recall. `STAR` and `Kallisto` perform similarly in general, though `Kallisto` achieves a lower (better) FDR than `STAR`. Taking the F1-score as a summary statistic, we observe that all methods perform reasonably well, and that, in general, alignment-based methods do not seem to be more accurate than mapping-based methods. We also observe that `RapMap` yields very accurate mapping results that match or exceed those of `Bowtie 2`.

Additionally, we tested the impact of noisy reads (i.e. reads not generated from the indexed reference) on the accuracy of the different mappers and aligners. To create these background reads, we use a model inspired by (Gilbert et al., 2004), in which reads are sampled from nascent, un-spliced transcripts. The details of this experiment are included in Appendix A.3.

2.3.2 Speed and concordance on experimental data

We also explore the concordance of `RapMap` with different mapping and alignment approaches using experimental data from the study of Cho et al. (2014) (NCBI GEO accession `SRR1293902`). The sample consists of ~ 26 million 75 base-pair, paired-end reads sequenced on an Illumina HiSeq.

Since we do not know the true origin of each read, we have instead examined the agreement between the different tools (see Figure 2.3). Intuitively, two tools agree on the mapping locations of a read if they align / map this read to the same subset of the reference transcriptome (i.e. the same set of transcripts). More formally, we define the elements of our universe, $\mathcal{U}$, to be tuples consisting of a read identifier and the set of transcripts returned by a particular tool. For example, if, for read r_i, tool A returns alignments to transcripts $\{t_1, t_2, t_3\}$ then $e_{Ai} = (r_i, \{t_1, t_2, t_3\}) \in \mathcal{U}$. Similarly, if tool B maps read r_i to transcripts $\{t_2, t_3, t_4\}$ then $e_{Bi} = (r_i, \{t_2, t_3, t_4\}) \in \mathcal{U}$. Here, tools A and B do not agree on the mapping of read r_i. Given a universe $\mathcal{U}$ thusly-defined, we can employ the normal notions of set intersection and difference to explore how different subsets of methods agree on the mapping locations of the sequenced reads. These concordance results are presented in Figure 2.3, which uses a bar plot to show the size of each set of potential intersections between the results of the tools we consider. In Figure 2.3 the dot matrix below the bar plot identifies the tools whose results are intersected to produce the corresponding bar. Tools producing mappings and alignments are denoted with black and red dots and bars respectively. The left bar plot shows the size of the unique tuples produced by each tool (alignments / mappings that do not match with any other tool). The right bar plot shows the total number of tuples produced by each tool, and well as the concordance among all different subsets of tools.

Under this measure of agreement, `RapMap` and `Kallisto` appear to agree on the exact same transcript assignments for the largest number of reads. Further, `RapMap` and `Kallisto` have the largest pairwise agreements with the aligners (`STAR` and `Bowtie 2`) — that is, the traditional aligners exactly agree more often with these tools than with each other. It is important to note that one possible reason we see (seemingly) low agreement between `Bowtie 2` and other methods is because the transcript alignment sets reported by `Bowtie 2` are generally larger (i.e. contain more transcripts) than those returned by other methods, and thus fail to qualify under our notion of agreement. This occurs, partially, because `RapMap` and `Kallisto` (and to some extent `STAR`) do not tend to return sub-optimal multi-mapping locations. However, unlike `Bowtie 1`, which provided an option to return only the best "stratum" of alignments, there is no way to require that `Bowtie 2` return only the best multi-mapping locations for a read. We observe similar behavior for `Bowtie 2` (i.e. that it returns a larger set of mapping locations) in the synthetic tests as well, where the average number of hits per read is higher than for the other methods (see Table 2.1). In terms of runtime, `RapMap`, `STAR` and `Bowtie 2` take 3, 26, and 1020 minutes respectively to align the reads from this experiment using 4 threads. We also observed a similar trend in terms of the average

number of hits per read here as we did in the synthetic dataset. The average number of hits per read on this data were 4.56, 4.68, 4.21, 7.97 for `RapMap`, `Kallisto`, `STAR` and `Bowtie 2` respectively.

2.4 Application of quasi-mapping for transcript quantification

While mapping cannot act as a stand-in for full alignments in all contexts, one problem where similar approaches have already proven very useful is transcript abundance estimation. Recent work (Patro, Mount, and Kingsford, 2014; Zhang and Wang, 2014; Bray et al., 2016; Patro et al., 2017) has demonstrated that full alignments are not necessary to obtain accurate quantification results. Rather, simply knowing the transcripts and positions where reads may have reasonably originated is sufficient to produce accurate estimates of transcript abundance. Thus, we have chosen to apply quasi-mapping to transcript-level quantification as an example application, and have implemented our modifications as an update to the Sailfish (Patro, Mount, and Kingsford, 2014) software, which we refer to as quasi-Sailfish. These changes are present in the Sailfish software from version `0.7` forward. Here, we compare this updated method to the transcript-level quantification tools RSEM (Li et al., 2010), Tigar2 (Nariai et al., 2014) and `Kallisto` (Bray et al., 2016), the last of which is based on the pseudo-alignment concept mentioned above.

2.4.1 Transcript quantification

In an RNA-seq experiment, the underlying transcriptome consists of M transcripts and their respective counts. The transcriptome can be represented as a set $\mathcal{X} = \{(t_1, \ldots, t_M), (c_1, \ldots, c_M)\}$, where t_i denotes the nucleotide sequence of transcript i and c_i denotes the number of copies of t_i in the sample. The length of transcript t_i is denoted by l_i. Under ideal, uniform, sampling conditions (i.e. without considering various types of experimental bias), the probability of drawing a fragment from a transcript t_i is proportional to its nucleotide fraction (Li et al., 2010) denoted by $\eta_i = \frac{c_i l_i}{\sum_{j=1}^{M} c_j l_j}$.

If we normalize the η_i for each transcript by its length l_i, we obtain a measure of the relative abundance of each transcript called the transcript fraction (Li et al., 2010), which is given by $\tau_i = \frac{\frac{\eta_i}{l_i}}{\sum_{j=1} \frac{\eta_i}{l_i}}$.

When performing transcript-level quantification, $\boldsymbol{\eta}$ and $\boldsymbol{\tau}$ are generally the quantities we are interested in inferring. Since they are directly related, knowing one allows us to directly compute the other. Below, we describe our approach to approximating the estimated number of reads originating from each transcript, from which we estimate $\boldsymbol{\tau}$ and, subsequently transcripts per million (TPM).

TABLE 2.2: Performance evaluation of different tools along with quasi enabled sailfish (q-Sailfish) with other tools on synthetic data generated by Flux simulator / RSEM simulator.

	`Kallisto`	RSEM	q-Sailfish	Tigar 2
Proportionality corr.	0.74/0.91	0.78/0.93	0.75/0.91	0.77/0.93
Spearman corr.	0.69/0.91	0.73/0.93	0.70/0.91	0.72/0.93
TPEF	0.77/0.53	0.96/0.49	0.60/0.53	0.59/0.50
TPME	-0.24/0.00	-0.37/-0.01	-0.10/0.00	-0.09/0.00
MARD	0.36/0.29	0.29/0.25	0.31/0.29	0.26/0.23
wMARD	4.68/1.00	5.23/0.88	4.45/1.01	4.35/0.94

2.4.2 Quasi-mapping-based Sailfish

Using the quasi-mapping procedure provided by `RapMap` as a library, we have updated the Sailfish (Patro, Mount, and Kingsford, 2014) software to make use of quasi-mapping, as opposed to individual k-mer counting, for transcript-level quantification. In the updated version of Sailfish, the `index` command builds the quasi-index over the reference transcriptome as described in Section 2.2. Given the index and a set of sequenced reads, the `quant` command quasi-maps the reads and uses the resulting mapping information to estimate transcript abundances.

To reduce the memory usage and computational requirements of the inference procedure, quasi-Sailfish reduces the mapping information to a set of equivalence classes over sequenced fragments. These equivalence classes are similar to those used in Nicolae et al. (2011), except that the position of each fragment within a transcript is not considered when defining the equivalence relation. Specifically, any fragments that map to exactly the same set of transcripts are placed into the same equivalence class. Following the notation of Patro et al. (2017), the equivalence classes are denoted as $\boldsymbol{\mathcal{C}} = \{\mathcal{C}^1, \mathcal{C}^2, \dots\}$, and the count of fragments associated with equivalence class $\mathcal{C}^j$ is given by d^j. Associated with each equivalence class $\mathcal{C}^j$ is an ordered collection of transcript identifiers $\mathbf{t}^j = (t_{j1}, t_{j2}, \dots)$ which is simply the collection of transcripts to which all equivalent fragments in this class map. We call $\mathbf{t}^j$ the *label* of class $\mathcal{C}^j$.

Inferring transcript abundances

The equivalence classes $\boldsymbol{\mathcal{C}}$ and their associated counts and labels are used to estimate the number of fragments originating from each transcript. The estimated count vector is denoted by $\boldsymbol{\alpha}$, and α_i is the estimated number of reads originating from transcript t_i. In quasi-Sailfish, we use the variational Bayesian expectation maximization (VBEM) algorithm to infer the parameters (the estimated number of reads originating from each transcript) that maximize a variational objective. Specifically, we maximize a simplified version of the variational objective of Nariai et al. (2013).

The VBEM update rule can be written as a simple iterative update in terms of the equivalence classes, their counts, and the prior (α_0). The iterative update rule for the VBEM is:

$$\alpha_i^{u+1} = \alpha_0 + \sum_{\mathcal{C}^j \in \mathcal{C}} d_j \left(\frac{e^{\gamma_i^u} \frac{1}{\hat{l}_i}}{\sum_{t_k \in \mathbf{t}^j} e^{\gamma_k^u} \frac{1}{\hat{l}_k}} \right), \tag{2.1}$$

where

$$\gamma_i^u = \Psi(\alpha_0 + \alpha_i^u) - \Psi(\sum_k \alpha_0 + \alpha_k^u) \tag{2.2}$$

and $\Psi(\cdot)$ is the digamma function. Here, $\hat{l}_i$ is the *effective* length of transcript t_i, computed as in Li et al. (2010). To determine the final estimated counts — $\boldsymbol{\alpha}$ — Equation (2.1) is iterated until convergence. The estimated counts are considered to have converged when no transcript has estimated counts differing by more than one percent between successive iterations.

Given $\boldsymbol{\alpha}$, we compute the TPM for transcript i as

$$TPM_i = 10^6 \frac{\frac{\alpha_i}{\hat{l}_i}}{\sum_j \frac{\alpha_j}{\hat{l}_j}}. \tag{2.3}$$

Sailfish outputs, for each transcript, its name, length, effective length, TPM and the estimated number of reads originating from it.

2.4.3 Quantification performance comparison

We compared the accuracy of quasi-Sailfish (Sailfish v0.9.0; q-Sailfish in Table 2.2) to the transcript-level quantification tools RSEM (Li et al., 2010) (v1.2.22), Tigar 2 (Nariai et al., 2014) (v2.1), and `Kallisto` (Bray et al., 2016) (v0.42.4) using 6 different accuracy metrics and data from two different simulation pipelines. One of the simulated datasets was generated with the Flux Simulator (Griebel et al., 2012), and is the same dataset used in Section 2.3 to assess mapping accuracy and performance on synthetic data. The other dataset was generated using the RSEM simulator via the same methodology adopted by Bray et al. (2016). That is, RSEM was run on sample `NA12716_7` of the Geuvadis RNA-seq data (Lappalainen et al., 2013) to learn model parameters and estimate true expression. The learned model was then used to generate the simulated dataset, which consists of 30 million 75 bp paired-end reads.

We measure the accuracy of each method based on the estimated versus true number of reads originating from each transcript, and consider 6 different metrics of accuracy;

proportionality correlation (Lovell et al., 2015), Spearman correlation, the true positive error fraction (TPEF), the true positive median error (TPME), the mean absolute relative difference (MARD) and the weighted mean absolute relative difference (wMARD). Detailed definitions for the last four metrics are provided in Appendix A.5.

Each of these metrics captures a different notion of accuracy, and all are reported to provide a more comprehensive perspective on quantifier accuracy. The first two metrics — proportionality and Spearman correlation — provide a global notion of how well the estimated and true counts agree, but are fairly coarse measures. The TPEF assesses the fraction of transcripts where the estimate is different from the true count by more than some nominal fraction (here 10%). Unlike TPEF, the TPME metric takes into account the direction of the mis-estimate (i.e. is it an over or under-estimate of the true value). However, both metrics are assessed only on truly-expressed transcripts, and so provide no insight into the tendency of a quantifier to produce false positives.

The absolute relative difference (ARD) metric has the benefit of being defined on all transcripts as opposed to only those which are truly expressed and ranges from 0 (lowest) to 2 (highest). Since the values of this metric are tightly bounded, the aggregate metric, MARD, is not dominated by high expression transcripts. Unfortunately, it therefore has limited ability to capture the magnitude of mis-estimation. The wMARD metric attempts to account for the magnitude of mis-estimation, while still trying to ensure that the measure is not completely dominated by high expression transcripts. This is done by scaling each ARD_i value by the logarithm of the expression.

Table 2.2 shows the performance of all 4 quantifiers, under all 6 metrics, on both datasets. While all methods seem to perform reasonably well, some patterns emerge. RSEM seems to perform very well in terms of the correlation metrics, but less well in terms of the TPEF, TPME, and wMARD metrics (specifically in the Flux Simulator-generated dataset). This is likely a result of the lower mapping rate obtained on this data by RSEM's very strict `Bowtie 2` parameters. Tigar 2 generally performs very well under a broad range of metrics, and produces highly-accurate results. However, it is *by far* the slowest method considered here, and requires over a day to complete on the Flux simulator data and almost 7 hours to complete on the RSEM-sim data given 16 threads (and not including the time required for `Bowtie 2` alignment of the reads). Finally, both quasi-Sailfish and `Kallisto` perform well in general under multiple different metrics, with quasi-Sailfish tending to produce somewhat more accurate estimates. Both of these methods also completed in a matter of minutes on both datasets.

One additional pattern that emerges is that the RSEM-sim data appears to present a much simpler inference problem compared to the Flux Simulator data. One reason for this may be that the RSEM-sim data is very "clean" — yielding concordant mapping rates well over 99%, even under RSEM's strict `Bowtie 2` mapping parameters. As such, all methods tend to perform well on this data, and there is comparatively little deviation between the methods under most metrics.

For completeness, we also provide (in Appendix A.4) the results, under all of these metrics, where the true and predicted abundances are considered in terms of TPM rather than number of reads. We find that the results are generally similar, with the exception that TIGAR 2 performs considerably worse under the TPM measure.

2.5 Application of quasi-mapping for clustering *de novo* assemblies

TABLE 2.3: Performance of CORSET, CD-HIT and `RapMap` enabled clustering (R-CL) on yeast and human data

	Human			Yeast		
	CORSET	CD-HIT	R-CL	CORSET	CD-HIT	R-CL
precision	0.96	0.96	0.95	0.36	0.41	0.36
recall	0.56	0.37	0.60	0.63	0.36	0.71
time (min)	957	268	8	23	5	2

Estimating gene-expression from RNA-seq reads is an especially challenging task when no reference genome is present. Typically, this problem is solved by performing *de novo* assembly of the RNA-seq reads, and subsequently mapping these reads to the resulting contigs to estimate expression. Due to sequencing errors and artifacts, and genetic variation and repeats, *de novo* assemblers often fragment individual isoforms into separate assembled contigs. Davidson and Oshlack (2014) argue that better differential expression results can be obtained in *de novo* assemblies if contigs are first clustered into groups. They present a tool, CORSET, to perform this clustering, and compare their approach to existing tools such as CD-HIT (Fu et al., 2012). CD-HIT compares the sequences (contigs) directly, and clusters them by sequence similarity. CORSET, alternatively, aligns reads to contigs (allowing multimapping) and defines a distance between each pair of contigs based on the number of multimapping reads shared between them, and the changes in estimated expression inferred for these contigs under different conditions. Hierarchical agglomerative clustering is then performed on these distances to obtain a clustering of contigs.

Here, we show how `RapMap` can be used for the same task, by taking an approach similar to that of CORSET. First, we map the RNA-seq reads to the target contigs and simultaneously construct equivalence classes over the mapped fragments as in Section 2.4. We construct a weighted, undirected graph from these equivalence classes as follows. Given a set of contigs $\mathbf{c}$ and the equivalence classes $\mathcal{C}$, we construct $G = (V, E)$ such that $V = \mathbf{c}$, and $E = \{\{u, v\} \mid \exists j : u, v \in \mathbf{t}^j\}$. We define the weight of edge $\{u, v\}$ as $w(u, v) = \frac{R_{u,v}}{\min(R_u, R_v)}$. Here R_u is the total number of reads belonging to all equivalences classes in which contig u appears in the label. R_v is defined analogously. $R_{u,v}$ is the total

sum of reads in all equivalence classes for which contigs u and v appear in the label. Given the undirected graph G, we use the *Markov Cluster Algorithm*, as implemented in MCL (Van Dongen, 2000), to cluster the graph.

To benchmark the time and accuracy of our clustering scheme compared to CD-HIT and CORSET, we used two datasets from the CORSET paper (Davidson and Oshlack, 2014). The first dataset consists of 231 million human reads in total, across two conditions, each with three replicates (as originally described by Trapnell et al. (2013)). The second dataset, from yeast, was originally published by Nookaew et al. (2012) and consists 36 million reads, grown in two different conditions with three replicates each. For both of these datasets, we consider clustering the contigs of the corresponding *de novo* assemblies, which were generated using Trinity (Grabherr et al., 2011).

To measure accuracy, we consider the precision and recall induced by a clustering with respect to the true genes from which each contig originates. Assembled contigs were mapped to annotated transcripts using BLAT (Kent, 2002), and labeled with their gene of origin. A pair of contigs from the same cluster is regarded as true positive (tp) if they are from the same gene in the ground truth set. Similarly, a pair is a false positive (fp) if they are not from same gene but are clustered together. A pair is a false negative (fn) if they are from same gene but not predicted to be in the same cluster and all the remaining pairs are true negatives (tn). With these definitions of tp, fp, tn and fn we can define precision and recall in standard manner. As shown in Table 2.3, when considering both precision and recall, `RapMap` (quasi-mapping) enabled clustering performs substantially better than CD-HIT and similar to CORSET. `RapMap` enabled clustering takes 8 minutes and 2 minutes to cluster the human and yeast datasets respectively — which is substantially faster than the other tools. To generate the timing results above, CD-HIT was run with 25 threads. The time recorded for CORSET consists of both the time required to align the reads using `Bowtie 2` (using 25 threads) and the time required to perform the actual clustering, which is single threaded. The time recorded for `RapMap` enabled clustering consists of the time required to quasi-map the reads, build the equivalence classes and construct the graph (using 25 threads), plus the time required to cluster the graph with MCL (using a single thread). Overall, on these datasets, `RapMap`-enabled clustering appears to provide comparable or better clusterings than existing methods, and produces these clusterings much more quickly.

2.6 Conclusion

In this chapter we have argued for the usefulness of our novel approach, quasi-mapping, for mapping RNA-seq reads. More generally, we suspect that read *mapping*, wherein sequencing reads are assigned to reference locations, but base-to-base alignments are not computed, is a broadly useful tool. The speed of traditional aligners like `Bowtie 2` and `STAR` is limited by the fact that they must produce optimal alignments for each location to which a read is reported to align.

In addition to showing the speed and accuracy of quasi-mapping directly, we apply it to two problems in transcriptome analysis. First, we have updated the Sailfish software to make use of the quasi-mapping information produced by `RapMap`, rather than direct k-mer counts, for purposes of transcript-level abundance estimation. This update improves both the speed and accuracy of Sailfish, and also reduces the complexity of its codebase. We demonstrate, on synthetic data generated via two different simulators, that the resulting quantification estimates have accuracy comparable to state-of-the-art tools. We also demonstrate the application of `RapMap` to the problem of clustering *de novo* assembled contigs, a task that has been shown to improve expression quantification and downstream differential expression analysis (Davidson and Oshlack, 2014). `RapMap` can produce clusterings of comparable or superior accuracy to those of existing tools, and can do so much more quickly.

However, `RapMap` is a stand-alone mapping program, and need not be used only for the applications we describe here. We expect that quasi-mapping will prove a useful and rapid alternative to alignment for tasks ranging from filtering large read sets (e.g. to check for contaminants or the presence or absence specific targets) to more mundane tasks like quality control and, perhaps, even to related tasks like metagenomic and metatranscriptomic classification and abundance estimation.

We hope that the quasi-mapping concept, and the availability of `RapMap` and the efficient and accurate mapping algorithms it exposes, will encourage the community to explore replacing alignment with mapping in the numerous scenarios where traditional alignment information is un-necessary for downstream analysis.

Chapter 3

Selective Alignment

3.1 Background

Since its introduction in 2008 (Lister et al., 2008; Nagalakshmi et al., 2008; Mortazavi et al., 2008), transcriptome profiling via RNA-seq has become a popular and widely-used technique to profile gene- and transcript-level expression, and to identify and assemble novel transcripts. Expression estimation is often done with the goal of subsequently performing differential expression analysis on the gene abundance profiles. In response to improvements in RNA-seq quality and read lengths, as well as significant improvements in the available quantification methods, it has also become increasingly common to perform quantification and differential testing at the transcript level. Recently, very fast computational methods (Patro, Mount, and Kingsford, 2014; Bray et al., 2016; Patro et al., 2017; Ju et al., 2017) for transcript abundance estimation have been developed which obtain their speed, in part, by forgoing the traditional step of aligning the reads to the reference genome or transcriptome. These methods have gained popularity due to their markedly smaller computational requirements and their simplicity of use compared to more traditional quantification "pipelines" that require alignment of the sequencing reads to the genome or transcriptome, followed by the subsequent processing of the resulting BAM file to obtain quantification estimates.

In various assessments on simulated data (Kanitz et al., 2015; Germain et al., 2016; Zhang et al., 2017), these lightweight methods have compared favorably to well-tested but much slower methods for abundance estimation, like RSEM (Li and Dewey, 2011), coupled with alignment methods such as Bowtie2 (Langmead and Salzberg, 2012). However, assessments based primarily (or entirely) upon simulated data often fail to capture important aspects of real experiments, and similar performance among methods on such simulated datasets does not necessarily generalize to experimental data. Popular methods for transcript quantification (Zhang and Wang, 2014; Ju et al., 2017; Patro et al., 2017; Bray et al., 2016; Patro, Mount, and Kingsford, 2014; Li and Dewey, 2011; Vuong et al., 2018; Hensman et al., 2015; Glaus, Honkela, and Rattray, 2012) differ in many aspects, ranging from how they handle read mapping and alignment, to the optimization algorithms they employ, to differences in their generative models or which biases they attempt to model and correct. These differences are often

obscured when analyzing simulated data, since aspects of experimental data that can lead to substantial divergence in quantification estimates are not always properly recapitulated in simulations.

We focus on the effect of read mapping on the resulting transcript quantification estimates. To compare the effect of different alignment and mapping methods on RNA-seq transcript quantification and related downstream analysis, we have picked tools from three different categories of mapping strategies: (1) unspliced alignment of RNA-seq reads directly to the transcriptome, (2) spliced alignment of RNA-seq reads to the annotated genome (with subsequent projection to the transcriptome), and (3) (unspliced) lightweight mapping (quasi-mapping) of RNA-seq reads directly to the transcriptome. While numerous different lightweight mapping approaches exist (Patro, Mount, and Kingsford, 2014; Zhang and Wang, 2014; Bray et al., 2016; Srivastava et al., 2016b; Ju et al., 2017), and the degree to which they diverge from alignment-based methods can differ, a key feature shared by such approaches is that they do not validate predicted fragment mappings via an alignment score, which precludes them from discerning loci where the best mappings would not admit a reasonable-scoring alignment (i.e. spurious mappings). Furthermore, the focus on speed means that such methods tend not to explore sub-optimal mapping loci, despite the fact that such loci might admit the best alignment scores and therefore be the most likely origin for a fragment. We show that differences in how reads are aligned or mapped can lead to considerable differences in the predicted abundances. Specifically, we find that lightweight mapping approaches, which are generally highly concordant with traditional alignment approaches in simulated data, can lead to quite different abundance estimates from alignment-based methods in experimental data. These differences happen across a large number of samples, but the magnitude of the differences can vary substantially from sample to sample. We also find that these differences appear even when exactly the same optimization procedure is used to infer transcript abundances. Instead, these differences are a result of the different mapping and alignment approaches returning distinct, and sometimes even disjoint, mapping loci for certain reads.

Due to the absence of a ground truth in experimental data, it is difficult to categorically specify which approaches produce more accurate estimates. However, by investigating the divergence we observe among the quantifications produced by different methods, and the differences in read mapping that lead to this divergence in quantifications, we uncover some primary failure cases of different alignment and mapping strategies. This leads us to compare and combine the results of different alignment strategies, and allows us to curate a set of oracle alignments for experimental samples. Comparing various approaches to the oracle provides further evidence for a hypothesis, raised in Sarkar et al. (2018) and Vuong et al. (2018), that lightweight mapping approaches may suffer from spurious mappings leading to a decrease in the resulting quantification accuracy compared to alignment-based approaches. We also demonstrate that, even among alignment-based approaches, non-trivial differences arise between quantifications based upon mapping to the transcriptome (using Bowtie2 (Langmead and

Salzberg, 2012)) and quantifications based upon mapping to the genome and subsequently projecting these alignments into transcriptomic coordinates (using STAR (Dobin et al., 2013)). Both of these alignment-based approaches sometimes disagree with the oracle, but do so for different subsets of fragments and to a varying degree among different samples.

Finally, we introduce an improved mapping algorithm, selective alignment (SA), that is designed to remain fast, while simultaneously eliminating many of the mapping errors made by lightweight approaches. SA is integrated into the `Salmon` (Patro et al., 2017) transcript quantification tool. Our proposed method increases both the sensitivity and specificity of fast read mapping. It relies upon alignment scoring to help differentiate between mapping loci that would otherwise be indistinguishable due to, for example, similar exact matches along the reference. Our approach also determines when even the best mapping for a read exhibits insufficient evidence that the read truly originated from the locus in question, allowing it to avoid spurious mappings. We also attempt to address one of the failure modes of direct alignment against the transcriptome, compared to spliced alignment to the genome: when a sequenced fragment originates from an unannotated genomic locus bearing sequence-similarity to an annotated transcript, it can be falsely mapped to the annotated transcript since the relevant genomic sequence is not available to the method. We describe a procedure that makes use of MashMap (Jain et al., 2018) to identify and extract such sequence-similar *decoy* regions from the genome. The normal `Salmon` index is then augmented with these decoy sequences, which are handled in a special manner during mapping and alignment scoring, leading to a reduction in such cases of false mappings. We benchmark two variants of this approach; one in which we extract a small collection of decoy sequences via the procedure mentioned above (designated as `SA`), and one in which we align against the transcriptome and whole genome simultaneously, allowing us to detect fragments that better map to a non-transcriptomic target. The latter approach, which essentially treats the whole genome as *decoy* sequence, is denoted as `SAF`. We benchmark these approaches on both simulated data and a broad collection of experimental RNA-seq samples, and demonstrate that they lead to improved concordance with the abundance estimates obtained via quantification following traditional alignment.

3.2 Materials and Methods

3.2.1 Selective alignment

Selective alignment is based on the pufferfish indexing data structure first described in (Almodaresi et al., 2018). Moreover, the index is augmented with the relevant decoy sequence (either restricted decoy sequence as described in Appendix B.2 or the entire genome) which is marked during indexing and handled in a special manner during alignment scoring.

The mapping approach works in 5 distinct phases (for paired-end reads). First, exact matches between the read and transcriptome are collected. Second, the set of transcripts to be considered for further processing are extracted. Third, exact match chaining and chain scoring, using the algorithm of Li (2018), is used to determine the relevant putative mapping loci for the read. Fourth (for paired-end reads), the mappings for the first and second read of the pair are matched to determine the mapping loci for the whole fragment. Finally, the computed mapping loci are scored using extension alignment scoring (Li, 2018; Suzuki and Kasahara, 2018) before, after and between the exact matches belonging to the highest-scoring chain for each mapping. In this final step, information about the decoy sequences is used to determine which mappings are considered valid and which are not (details are provided below).

In the first phase of the mapping algorithm, uni-MEMs (Liu et al., 2016) are collected between the sequenced fragment (with each read end treated separately) and the index. The uni-MEMs are found via k-mer lookup, and then are extended maximally until the end of a unitig is encountered, the end of the read is encountered, or a mismatch is encountered. If uni-MEM extension terminates because it reaches the end of a unitig or because of a mismatch, search advances in the read and subsequent k-mers are queried in the index to collect other uni-MEMs shared between the read and the index. This process is repeated until the end of the read, and results in a collection of uni-MEMs — matches between the read and the index that can be efficiently decoded into the implied matches between the read and the reference.

In the second phase, uni-MEMs are projected to their corresponded reference loci, and the exact matches are collated by reference and orientation. Let M denote the number of matches for the transcript, orientation pair with the maximum number of matches for the current read. An optional (user-determined) filtering policy is applied, whereby any transcript and orientation pair that does not have at least τM matches is discarded from further consideration. The value of τ is a user-specified fraction, set as 0.65 by default. This optional filtering policy, termed as "filtering before chaining", is disabled by default, but can be enabled via the command-line option `--hitFilterPolicy BEFORE`. Note that it was not enabled in our experiments and the default was used.

Next, matches along each transcript, orientation pair are sorted and compacted. This compaction is necessary since it is possible to have matches that are directly adjacent on both the read and reference, but which were not extracted as a single exact match during the first phase because the underlying uni-MEM terminated. This compaction phase eliminates such fragmentation due to uni-MEM termination, and reduces the number of exact matches that must be considered by the chaining algorithm. Candidate mapping locations are determined by applying the chaining algorithm of minimap2 (Li, 2018) to the exact matches for each transcript passing the previous filter. If multiple equally-good positions for a read along a transcript, in terms of their chaining score, are discovered, they are all propagated downstream in the mapping procedure until mappings for paired-end reads are merged. Likewise, if a read is determined to map to

a transcript in both the forward and reverse-complement orientation, then all equally-best mapping loci in both orientations are propagated downstream in the mapping procedure. Let S be the best chaining score obtained for any mapping of the current read. An optional filter is applied where any mapping with a chaining score less than $\tau' S$ is discarded from further consideration. By default, this filter, termed as "fitering after chaining", is enabled by default and τ' is set as 0.65.

For paired-end reads, the pairs are merged by determining, for each transcript, the locations of the read ends that respect the expected mapping constraints (e.g. that the leftmost position of the reverse complement read is to the right of the leftmost position of the forward-strand read, and the distances between the reads is less than the maximum allowed insert size). If passed the appropriate flag `--allowDovetails`, then dovetailed (*Bowtie2 user manual*) mappings are allowed, but they are prioritized below any non-dovetailed mapping.

All putative mappings are scored using the ksw2 (Li, 2018; Suzuki and Kasahara, 2018) library for alignment extension. We note that we compute only the optimal alignment score, and not the details of the alignment itself (i.e. the CIGAR string), which improves the speed of this mapping validation. To avoid redundant computation of the same alignment problem (which is quite prevalent when mapping directly to the transcriptome, as many alignments to alternatively-spliced transcripts will be identical), SA maintains a per-read alignment cache. This alignment cache is a hash table where the key is a hash of the reference transcriptome substring where the read is predicted to align, and the value is the previous alignment score computed for such a substring. Thus, if multiple transcripts would produce identical alignments for the same read, because the read maps to identical regions of these transcripts, SA is able to avoid this redundant work.

Finally, all of the relevant alignments are grouped by their associated alignments scores. Any alignments that fall below the (user-provided) threshold (default of 0.65 of the maximum obtainable alignment score) for a minimum valid alignment score are discarded. During alignment scoring, the score of the best alignment for a given fragment to any decoy sequence as well as to any non-decoy sequence is computed and stored. If the best alignment score to a decoy sequence is strictly greater than the best alignment score to a non-decoy sequence, then all of the fragment's mappings are considered invalid and the fragment is not considered for quantification. Otherwise, any alignments to decoy sequences are filtered out, and the remaining alignments to valid transcripts are further processed by `Salmon` using range-factorized equivalence classes (Zakeri et al., 2017), which allows the relevant information about the scores for the different alignments of the read to be appropriately summarized and used for quantification.

3.2.2 Analysis details

A note on the difference between SA and SAF This manuscript introduces the idea of selective alignment over a reference sequence indexed using the pufferfish (Almodaresi et al., 2018) data structure, implemented in the `Salmon` program. The index takes as input a set of decoy sequences, which are not part of the reference transcriptome and are, therefore, not quantified. In our analyses, we considered the performance of the selective alignment algorithm when paired with two different sets of decoys as input. In the first approach, referred to as SA in the manuscript, the index is built on the transcriptome and regions of the genome that have high sequence similarity with the transcriptome. In the second approach, referred to as SAF, the complete genome is included in the pufferfish index as a set of decoys. Hence, the index for SA contains a smaller portion of the genome, whereas the SAF index contains the full genome. Also, note that selective alignment replaces the quasi-mapping algorithm previously used in the `Salmon` program.

A note on on orphan and dovetail mappings We attempted to normalize for some mapping-related differences between methods that have little to do with the ability of the aligner to appropriately find the correct loci for a read, and instead have to do with constraints placed on what constitutes a valid mapping. Specifically, when projecting to the transcriptome, STAR disallows orphan mappings (cases where one end of a fragment aligns to a transcript but the other end does not). Likewise it is recommended practice in existing alignment-based quantification tools (**rsem**; Hensman et al., 2015; Glaus, Honkela, and Rattray, 2012), when using Bowtie2, to discard discordant and orphaned alignments. Thus, in our analyses, we disallowed orphaned mappings so that, in paired-end datasets, the pair is discarded if only one end of a fragment is mapped, or if the fragment ends only map to distinct transcripts. To be consistent with the default behavior of Bowtie2, the configurations of quasi-mapping and SA were also set to disallow dovetailed mappings (mappings where the first mapped base of the reverse complement strand read is upstream of the first mapped base of the forward strand read). While Bowtie2's scoring function (when performing global alignment) does not allow insertions or deletions to occur at the beginning or end of the read, we attempted to minimize the effect of this structural constraint on alignments by setting the `--gbar` parameter to 1.

A note on genomic alignment, as used in this manuscript. We explored differences that arise between quantification based on alignment of the sequencing reads to the genome and the transcriptome. We considered genomic alignment here to be the process of alignment to the genome — with the benefit of a known annotation — with subsequent projection to the transcriptome. That is, genomic alignment is characterized based on running STAR (with appropriate parameters) to align the reads to the genome, and then making use of the transcriptomically-projected alignments output by STAR via the `--quantMode TranscriptomeSAM` flag (as would be used in e.g. a STAR(Dobin

et al., 2013)/RSEM(**rsem**)-based quantification pipeline). Such an approach is necessarily concerned only with how well STAR is able to align the sequenced reads to the annotated transcriptome of the organism being assayed, and our assessment is concerned only with the accuracy of quantification of known and annotated isoforms. Importantly, spliced alignment of RNA-seq reads to the genome can be a useful tool in tackling a broader range of problems and in a larger set of cases than can unspliced alignment to a known transcriptome. For example, spliced alignment of sequencing reads to the genome can be done in the absence of an annotation of known isoforms, and can be used to help identify novel exons, isoforms, or transcribed regions of the genome, while unspliced alignment to a pre-specified set of transcripts does not admit this type of analysis. Further, alignment directly to the genome can easily cope with events like intron retention, which are more difficult to account for when using methods that align reads to the transcriptome.

A note on the influence of short transcripts on quantification. The human GENCODE v29 reference includes transcripts as short as 8bp, which is much shorter than a single sequencing read or the typical fragment length in most RNA-seq experiments. While RNA-seq might not be the appropriate method to quantify these transcripts, depending on the alignment method, they may have mapped reads and obtain non-zero expression values. In our analyses, we observed that lightweight mapping methods that do not perform end-to-end alignment tend to assign reads to shorter transcripts when there is an exact match. This effect has been explored in some detail by Wu et al. (2018). In such a scenario, it is hard to judge the true origin of the read, and while this may lead to some differences between mapping and alignment-based methods, we showed that the differences in quantification estimates for short transcripts account for only a very small fraction of the overall differences between methods. Since it is difficult to judge how these shorter transcripts, and the reads aligning to them, should be handled, we simply highlighted this issue and refrained from suggesting a particular strategy or attempting to determine which method performed better or worse on transcripts shorter than 300bp.

3.3 Results

3.3.1 Comparison between alignment and mapping algorithms

For benchmarking, we used quasi-mapping and SA (with either the similar-sequence decoy regions or the whole genome), both available in the `Salmon` (Patro et al., 2017) program, where quasi-mapping is a representative for lightweight mapping methods and SA is our proposed method that performs sensitive lightweight mapping followed by an efficient alignment-scoring procedure. For unspliced read alignment directly to the transcriptome, we used Bowtie2 (Langmead and Salzberg, 2012), which is an accurate and popular tool for unspliced alignment. Similarly, we used STAR (Dobin

et al., 2013) as representative of methods that perform spliced read alignment against the genome. We chose STAR, in particular, since it has the ability to project the aligned reads to transcriptomic coordinates, which allowed us to use a consistent quantification method, and also because it is part of the popular STAR (Dobin et al., 2013)/RSEM (Li and Dewey, 2011) transcript abundance estimation pipeline.

We used `Salmon` as the main quantification engine in our analyses since it supports quantification from quasi-mapping, SA, and via the output of traditional aligners. To the best of our knowledge, `Salmon` is the only quantification tool that has support for both lightweight mapping approaches and quantification using traditional alignments. We used `Salmon` in alignment mode to process the output from Bowtie2 and STAR. In tests on the initial simulated data, we also included RSEM. To remove variability in the quantification methods that is ancillary to our focus on mapping and alignment, we used the `-useEM` flag in `Salmon` for comparison against the EM-based algorithm of RSEM. Likewise, to eliminate variability due to the target set of transcripts being quantified, we passed the `-keepDuplicates` option to `Salmon` when indexing for subsequent mapping using quasi-mapping or SA[1].

Where mentioned, the "strict" and "RSEM" versions of Bowtie2 and STAR refer to these tools being run with the flags recommended in the RSEM manual (*RSEM manual*), which disallow insertions, deletions and soft-clipping in the resulting alignments. The difference between them is that the "strict" versions are quantified using `Salmon` and the "RSEM" versions using the RSEM expression calculation method. Throughout the text, we refer to the pipelines by the following shorthand (more details about the methods are given in Section 3.2.2, and Table B.1 and the full command line options provided to each tool are given in Appendix B.3):

- Bowtie2 – Alignment with Bowtie2 to the target transcriptome and allowing alignments with indels, followed by quantification using `Salmon` in alignment mode.
- Bowtie2_strict – Alignment with Bowtie2 to the target transcriptome and disallowing alignments with indels (i.e. using the same parameters as those used by RSEM), followed by quantification using `Salmon` in alignment mode.
- Bowtie2_RSEM – Alignment with Bowtie2 to the target transcriptome and disallowing alignments with indels, followed by quantification using RSEM.
- STAR – Alignment with STAR to the target genome (aided with the GTF annotation of the transcriptome) and projected to the transcriptome allowing alignments

[1]Though we performed indexing here with `-keepDuplicates` and quantification with `-useEM`, this is done only to eliminate controllable sources of variability between methods so as to isolate, as much as possible, the effect of differences in mapping. We generally recommend that duplicate transcripts are discarded during indexing, and that the offline phase of quantification is performed using the variational Bayesian EM.

with indels and soft clipping, followed by quantification using `Salmon` in alignment mode.

- STAR_strict – Alignment with STAR to the target genome (aided with the GTF annotation of the transcriptome) and projected to the transcriptome and disallowing alignments with indels or soft clipping, followed by quantification using `Salmon` in alignment mode.
- STAR_RSEM – Alignment with STAR to the target genome (aided with the GTF annotation of the transcriptome) and projected to the transcriptome and disallowing alignments with indels or soft clipping, followed by quantification using RSEM.
- quasi – quasi-mapping directly to the target transcriptome, coupled with quantification using `Salmon` in non-alignment mode.
- SA – Selective alignment directly to the target transcriptome and a set of decoy sequences (high similarity with the transcriptome), coupled with quantification using `Salmon` in non-alignment mode. (Details in Appendix B.2 and Section 3.2.1.)
- SAF – Selective alignment (full) directly to the target transcriptome and the genome, treated as decoy sequences, coupled with quantification using `Salmon` in non-alignment mode. (Details in Section 3.2.1.)

3.3.2 Performance on typical simulations

We used a Polyester(Frazee et al., 2015)-simulated dataset to show the performance of various methods on synthetic data. The distribution of transcript expression for this simulation was learned from an experimental (human) sample (`SRR1033204`, quantified using Bowtie2 with `Salmon`). We computed the Spearman correlation of quantification estimates from all the pipelines when compared against a known ground truth (in terms of read count). To simulate technical variation, we ran each simulation 10 times using the same input abundance distribution, but varying the random seed used by Polyester.

We observed that, though there are differences in correlation, all pipelines had somewhat similar overall performance on this simulated dataset, with the exception of STAR, which exhibited the lowest correlation. On this data, quasi-mapping performed better than aligning to the genome (and then projecting to the transcriptome) but marginally worse than doing traditional alignment against the transcriptome (both with and without the strict parameters for Bowtie2). We observed that SA performed better than aligning to the transcriptome using Bowtie2, except when quantified using RSEM, though the difference in this scenario is quite small. Finally, SAF performed very similarly to SA in these simulations, though SA, using the smaller decoy set, performed marginally better given that, in reality, all of the reads truly derive from annotated transcripts (with only very minor modifications due to simulated sequencing error).

Overall, the analysis on this synthetic dataset gives an impression that quantifications resulting from the different mapping approaches exhibit similar accuracy, and that all approaches quantify transcript abundances relatively well. While this is true for these simulated data, we show below that this observation does not generalize to experimental data. We posit that this is because, though great advancements have been made in improving the realism of simulated RNA-seq data, these simulations still fail to capture some of the complexities of experimental data. We describe below one particular way in which the realism of the simulated data can be increased by accounting for variations between the sequenced reads and the transcriptome used for quantification.

Method	Truth
Bowtie2	0.940 ± 0.001
Bowtie2_strict	0.941 ± 0.001
Bowtie2_RSEM	0.949 ± 0.000
SA	0.945 ± 0.000
SAF	0.944 ± 0.000
quasi	0.937 ± 0.000
STAR	0.915 ± 0.001
STAR_strict	0.912 ± 0.001
STAR_RSEM	0.919 ± 0.000

TABLE 3.1: **Spearman correlation against ground truth for data simulated using Polyester.** Note that the counts were based on a real sample from human.

3.3.3 Performance on simulations from a variant mouse transcriptome

An observation we made from the previous simulation was that disallowing indels using the RSEM parameters (used for strict and RSEM versions) for Bowtie2 and STAR did not adversely affect accuracy compared to using the default parameters of each method. We hypothesized that this is because the reads are simulated exactly from the reference transcriptome that is being used for alignment and quantification, and only sequencing errors (which are taken to consist entirely of substitution errors) are introduced by the simulator. Yet, in experimental data, the sample being quantified likely exhibits variation with respect to the reference against which the reads are aligned. Some of these variants will be single nucleotide polymorphisms (SNPs), while others will be indels and yet others may be larger structural variants. Thus, restricting alignments to disallow indels seems undesirable, unless one is quantifying against a personalized reference that is known to contain the variants present in the sample, which can potentially improve the accuracy of transcript quantification (Munger et al., 2014; Robert and Watson, 2015).

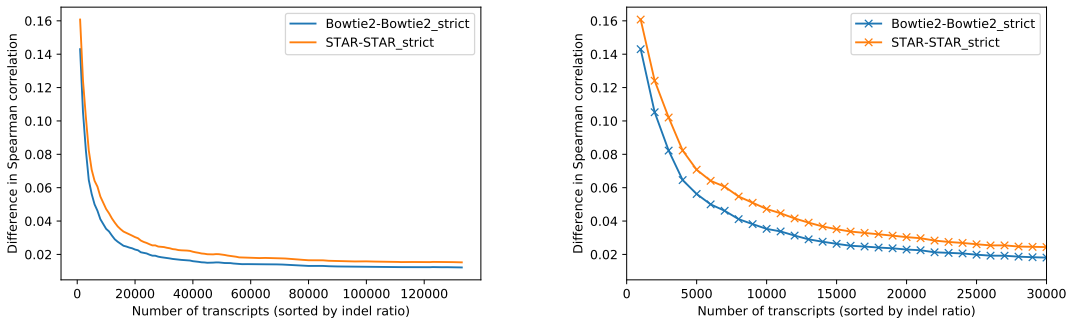
Method	PWK	GRCm38.91
Bowtie2	0.939 ± 0.001	0.934 ± 0.001
Bowtie2_strict	0.939 ± 0.001	0.923 ± 0.001
Bowtie2_RSEM	0.942 ± 0.001	0.925 ± 0.001
SA	0.941 ± 0.001	0.936 ± 0.001
SAF	0.939 ± 0.001	0.935 ± 0.001
quasi	0.940 ± 0.001	0.926 ± 0.000
STAR	0.935 ± 0.001	0.929 ± 0.001
STAR_strict	0.934 ± 0.000	0.914 ± 0.001
STAR_RSEM	0.937 ± 0.001	0.916 ± 0.001

TABLE 3.2: **Spearman correlation against ground truth for data simulated using Polyester.** Note that the reads were simulated using the reference containing the mouse PWK strain's variants.

To test the hypothesis that disallowing indels in the alignments will adversely affect quantification accuracy when simulating from a reference transcriptome containing realistic variants compared to the reference, we performed the following experiment. We obtained VCF files from the Sanger Mouse Genomes website[2] describing the variants present in the PWK mouse strain. Using g2gtools (Vincent and Kwangbom "KB" Choi, 2017), we generated a copy of the GRCm38.91 transcriptome containing the variants (including indels) present in the PWK strain and simulated reads from this transcriptome. The results presented in the first column of Table 3.2 show that when reads are aligned against the PWK strain's reference and indels are disallowed, the quantification estimates are as accurate as those derived from alignments allowing indels, as expected. However, when we aligned the reads back to the original mouse reference transcriptome (version GRCm38.91), we observed that, indeed, Bowtie2 performed better than Bowtie2_strict (second column of Table 3.2), and that, generally, disallowing indels in alignments has a negative effect on quantification accuracy.

To further analyze the influence of indels on quantification, we aligned the transcript sequences from the PWK strain and the original reference using edlib (Šošić and Šikić, 2017) and counted the total length of indels in each transcript compared to the unaltered transcript's original length (we refer to this quantity as the indel ratio). We then sorted the transcripts in descending order by their indel ratios, and evaluated at each cumulative subset, the difference in correlation with the truth between the quantifications using the alignment method and its "strict" variant. We evaluated this quantity increasing the cumulative subsets by 1000 transcripts at each step. We observed that the difference between methods is highest in transcripts that have a larger indel ratio (Figure 3.1). Hence, the impact of disallowing indels in the alignment can be considerable for reads that originate from transcripts that differ from the reference

[2] ftp://ftp-mouse.sanger.ac.uk/REL-1410-SNPs_Indels/



(A) (B)

FIGURE 3.1: **Impact on quantification accuracy when disallowing indels.** (a) Difference in correlation with the truth between both alignment methods and their "strict" variants, where indels are disallowed, on all mouse transcripts sorted by their indel ratios. (b) The same plot restricted to the $30,000$ transcripts with the largest indel ratio.

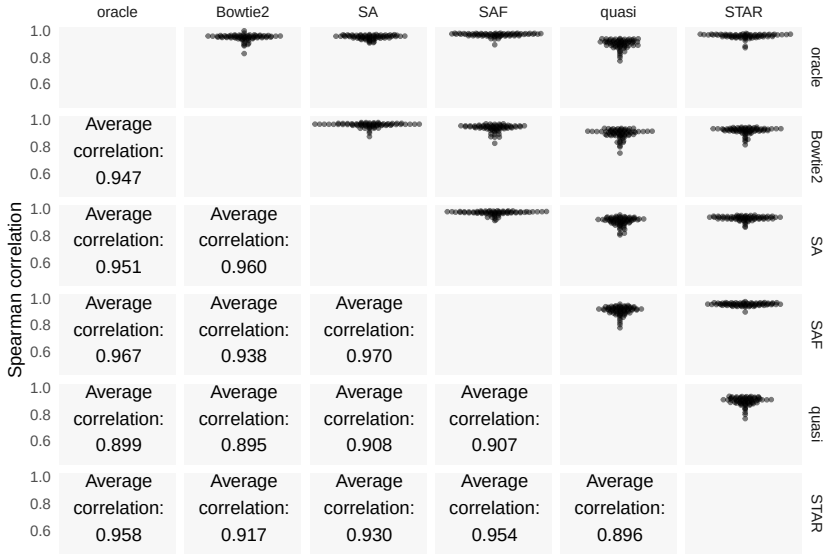
due to the presence of indels, and this can eventually lead to such transcripts being substantially misquantified.

Due to both the theoretical concerns and the practical evidence shown here, we proceeded in representing the alignment-based methods by using Bowtie2 and STAR in our comparisons, only in configurations that allow indels to occur in the alignments, and excluded from our analyses the "strict" and "RSEM" versions of the pipelines.

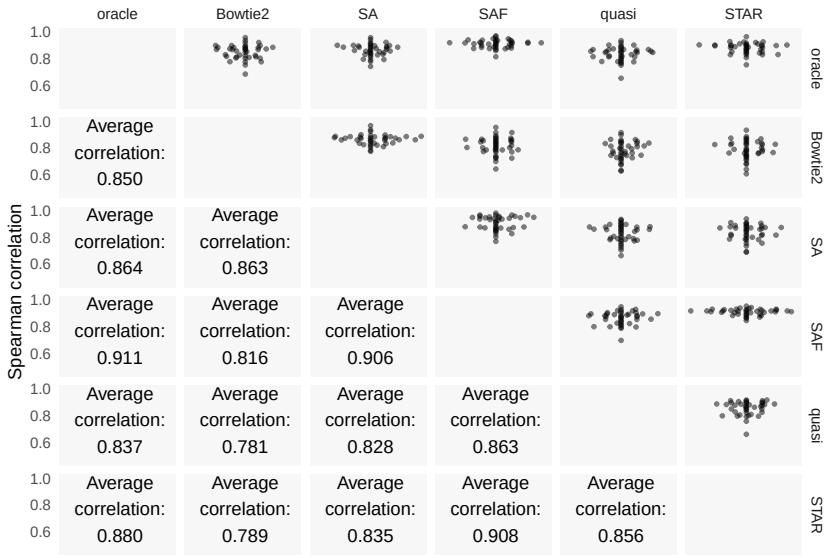
3.3.4 Randomly sampled experiments from NCBI database

It is crucial to evaluate the performance of the various tools on data from real experiments, that can be vastly more complex than even state-of-the-art simulations and can include processes, both known and unknown, that affect the underlying data in complicated ways. To analyze the accuracy of existing tools and study the impact of the artifacts (like the above) on experimental datasets, we randomly selected 200 human RNA-seq experiments from the NCBI database for further investigation. We then filtered the selected samples to include only paired-end libraries having a minimum read length of 75bp. After applying these filters, we were left with a set of 109 samples (69 bulk and 40 full-length single-cell). Before further processing, we applied adapter and (light) quality trimming using TrimGalore (Krueger, 2015; Martin, 2011)[3]. We also observed that the overall mapping rates across samples tended to be similar between all methods (Figure B.1), though Bowtie2 tends to exhibit the highest sensitivity (i.e. aligns the most reads) on average. Subsequently, we quantified all 109 samples using each of the remaining pipelines.

[3]The effect of trimming on the overall results was relatively minimal (result not shown).



(A) Bulk



(B) Single-cell

FIGURE 3.2: **Performance of each method on real bulk and single-cell datasets.** The top half of the matrix shows swarm plots of the pairwise correlations of the TPM values predicted by different approaches on the experimental samples. The bottom half shows the average Spearman correlation across the 109 bulk and single-cell samples.

Since no ground truth transcript abundances were available for these 109 experimental datasets, it became more difficult to analyze the accuracy of the different pipelines. However, we explored, manually, some of the cases where differences in mappings

and alignments led to divergence of quantification estimates between methods. Between Bowtie2, quasi-mapping, and STAR, the mappings seemed to fall into one of two major categories. In one case, Bowtie2 seemed to be appropriately reporting a more comprehensive set of best-scoring mappings than STAR and quasi-mapping. In the other case, the resulting sequencing fragment seemed to clearly arise from some unannotated region of the genome — either from intronic or intergenic sequence — and it was spuriously assigned by Bowtie2 and quasi-mapping to some set of annotated transcripts (though not always the same set). This led to the following observation: when the fragment truly originates from the annotated transcriptome, Bowtie2 appears the most sensitive and accurate method in aligning the read to the appropriate subset of transcripts, as is also supported by the variant transcriptome simulations from Section 3.3.3. However, this same sensitivity can sometimes lead Bowtie2 to spuriously align reads to the annotated transcriptome when they are better explained by some other (unannotated) genomic locus. In this latter case, STAR tends to report the correct alignment for the read, and, appropriately refrains from reporting alignments to annotated transcripts. These complications, in which reads are sequenced from underlying fragments that either overlap or are sequence-similar to annotated transcripts, is yet another factor that leads to divergent behavior between different mapping and alignment approaches, but which is not commonly considered in simulation.

These observations led us to combine information from both Bowtie2 and STAR to derive an *oracle* method, that avoids the obvious shortcomings, as listed above, of either of the constituent methods. To derive the oracle alignments in each sample, we used the following approach. First, we aligned the reads for the sample using both Bowtie2 and STAR, and for STAR we retained both the genomic and transcriptomic BAM files (i.e. we considered all of the alignments that STAR was able to produce to the genome, as well as those that it was able to successfully project to the transcriptome). Subsequently, we examined the reads that were aligned to the transcriptome using Bowtie2, and were aligned to the genome using STAR, but which STAR did not project to the transcriptome. For each such read, we examined the best-scoring transcriptomic alignment records produced by Bowtie2 as well as the best-scoring genomic alignment records produced by STAR. We compared the quality of these alignments between the tools by first parsing the extended CIGAR string (the `MD` tag), and assigning a score to each reported alignment. In our scoring scheme, we assigned 1 to every matched base while penalizing soft-clips, SNPs and indels by assigning a score of 0. We reported the score of an alignment as the sum of the number of properly matched bases along the ends of the read. If the transcriptomic alignment of Bowtie2 was of equal or higher quality to the genomic alignment, then we retained the transcriptomic alignment. Otherwise, we marked the fragment's alignment records for removal. We then processed the original Bowtie2 BAM file for the sample, removing alignments for all fragments that have been marked for removal. The result was a filtered version of the Bowtie2 BAM file in which spurious transcriptomic alignments have been removed. We quantified the sample by providing `Salmon` with this filtered BAM file, and refer to the resulting quantification estimates as the *oracle* estimates for this sample.

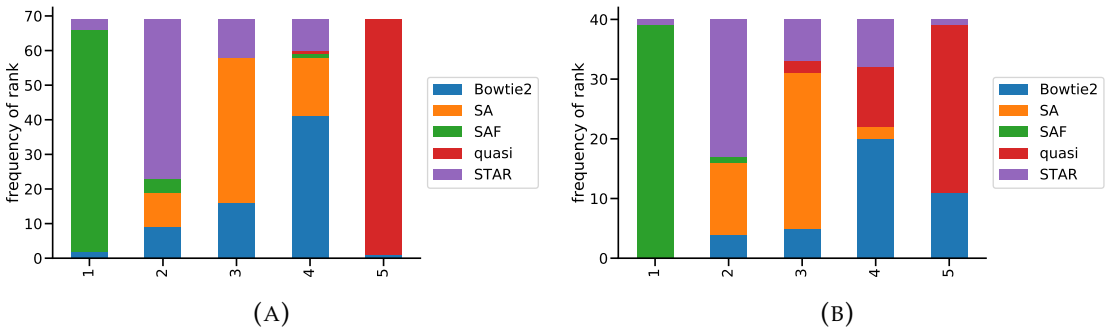


FIGURE 3.3: **Rank of each method in terms of correlation with oracle.** Histogram of the ranks across the 69 bulk samples (a) and the 40 full-length single-cell samples (b), of different methods in terms of the Spearman correlation of the method's abundance with the oracle. Here, the most correlated method is assigned rank 1, while the least correlated method is assigned rank 5.

While other complex alignment scenarios may occur, these oracle estimates represent quantification based on the set of alignments that avoid the obvious shortcomings of the different approaches being considered. Specifically, being based on alignment rather than lightweight-mapping, all alignments benefit from the improved sensitivity of Bowtie2's search procedure and are guaranteed to support a matching of the read to the reference of at least the required quality. Further, since these alignments are derived from Bowtie2, they likely correspond to a correct and comprehensive set of transcripts when the fragment does, in fact, originate from the annotated transcriptome. Finally, in the case where the fragment does not originate from an annotated transcript, and is instead the product of transcription from an unannotated locus, novel splicing, or intron retention, the corresponding alignment records have been removed using information from STAR's alignment to the genome, so that the fragment is not spuriously allocated to annotated transcripts. Thus, we treated the oracle quantifications as a proxy for the true abundances in the experimental samples.

In terms of Spearman correlation between all methods, we observed the highest average pairwise concordance between the oracle and SAF, in both the bulk and single-cell samples (Figure 3.2). SAF had the highest rank in order of its correlation with the oracle across the bulk and single-cell samples (histogram of the frequencies of these ranks in Figure 3.3), and had the closest mapping rate compared to the oracle (Figure B.1). Among the alignment-based methods, STAR displayed higher correlation with the oracle — especially in full-length single-cell samples — than did Bowtie2; a reversal of the trend that was observed with respect to the ground truth on the simulated data in Sections 3.3.2 and 3.3.3. Further, the overall correlations were lower, and the differences were larger, in the full-length single-cell samples than in the bulk samples. Bowtie2 tended to correlate highly with SA, while STAR tended to correlate highly with SAF, though SA and SAF were, themselves, highly correlated. Also, Bowtie2

and STAR both had a higher correlation with the oracle than they did with each other. This was somewhat expected since the oracle was created by considering the alignments of both Bowtie2 and STAR. However, this also suggested that the manners in which these approaches diverged from the oracle were largely distinct (i.e. they made different types of mistakes in alignment). The quantifications from lightweight mapping exhibited the lowest overall correlation with the oracle. These results were also indicative of the types of divergence between simulated and experimental datasets that we expected to observe. The trend was similar when comparing TPM values after discarding transcripts shorter than 300bp (Tables B.2 and B.3) and when comparing read counts predicted by each method, instead of TPM, as shown in Figure B.2. For the purpose of analyzing the performance of a lightweight mapping algorithm other than quasi-mapping, we also compared the estimates from `Kallisto` (Bray et al., 2016) against the other methods on these 109 experimental datasets. It displayed the lowest overall correlations with the alignment-based approaches (Figure B.3). This may be due, in part, to the fact that it altered both the quantification and mapping methodology, and because there were no options to control for structural constraints on the reported mappings (i.e. orphaned and dovetailed mappings). Thus, for consistency, we excluded it from the other analyses in the manuscript.

Finally, while one would not typically regard any of the average correlations in Figure 3.2 as poor, it is important to properly frame these differences as representing the aggregate Spearman correlation, across the samples, each quantifying $206,694$ transcripts (wherein most features are properly quantified as having an abundance of 0). A correlation coefficient is a single coarse metric, and as demonstrated in Section 3.3.3, even substantial quantification differences across thousands of transcripts can nevertheless result in small differences in the summary correlation. To explore some of the transcripts with large differences in quantification across methods, we performed differential transcript expression analysis across methods on the 109 samples, using limma-trend (Law et al., 2014). The counts per million (CPM) for the top 100, 500, and 1000 transcripts is shown in Figure B.4. This highlighted the divergence of the methods from each other, in terms of quantification and revealed clusters of transcripts that were differentially expressed under each method. Further, as described in Section 3.3.6, such differences can lead to considerable changes in which genes are found to be differentially expressed.

3.3.5 Simulation fails to capture complex patterns of real experiments, even when seeded from experimental abundances

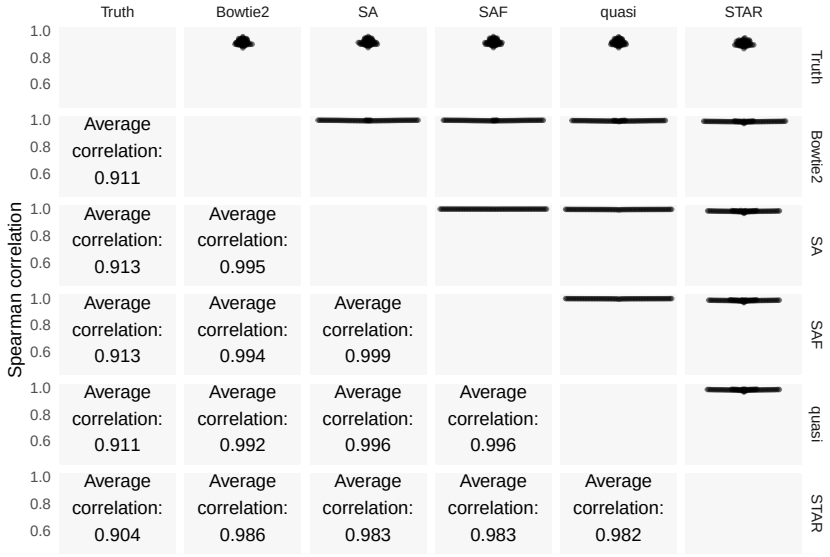
In principle, if the specific transcript expression profile was the primary source of quantification difficulty among the different approaches, we should be able to reproduce the types of divergence we observed between different methods in the experimental data (i.e. Figure 3.2) in simulation by simply creating simulations where the transcript expression profile is seeded with the estimated abundance results obtained from the

experimental samples (using e.g. the Bowtie2-derived quantifications). To test this hypothesis, we used Polyester (Frazee et al., 2015) to simulate 109 synthetic experiments where the expression profiles in each simulated sample were matched to those of the corresponding experimental sample's transcript abundances generated by the Bowtie2-based pipeline.

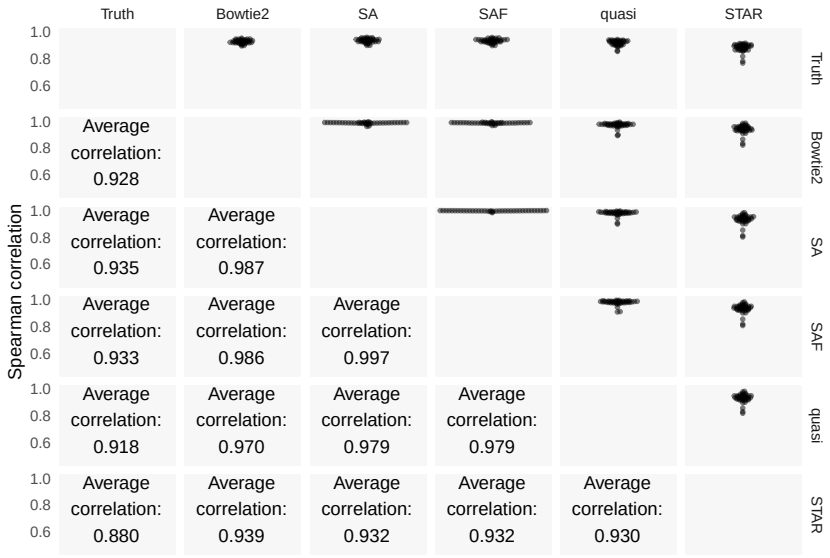
We quantified transcript abundance in all of these simulated samples using the same methods we considered in the experimental data. For transcript counts from simulated data, the correlation with the truth, and between methods, is shown in Figure 3.4. Shown in Figure B.5 is the correlation values calculated using the predicted TPMs instead of read counts. Clearly, the correlations among methods was markedly higher in the simulated data than in the experimental data (Figure 3.2), and multiple methods even showed a correlation ≥ 0.99. The variability between the samples was also considerably lower than what we observed in experimental data. This further suggested that comparison of correlations on simulated data is likely to be only a starting point in assessing different methods, as many salient differences that arise in experimental data disappear when the comparisons are performed on simulated data.

The variation in correlation across the simulated samples was inconsistent with the hypothesis that the distribution of transcript expressions alone is sufficient to simulate quantification scenarios as complicated as those observed in experimental data. This suggests that there are important aspects, apart from the underlying transcript expression profiles and simulation of read errors, that the simulation failed to capture. Though we do not know all such features, some important biological features, like structural variations (SV), SNP variants and small indel variants, which are sample-dependent rather than reference-dependent, are missing. Furthermore, transcription and sequencing in experimental RNA-seq samples are not limited to the fully-spliced, annotated transcript sequences present in a reference database, even for organisms as well-characterized as human and mouse. Reads derived from unannotated genomic regions that bear resemblance to annotated transcripts, or which only partially overlap the annotated features, can then be spuriously aligned to the annotated features, leading to inaccurate quantification of their abundances. Since such effects can vary from sample-to-sample, they do not affect the estimated expression of the annotated features in a uniform way, and can therefore affect subsequent analyses, such as differential expression testing.

We realize that sample-dependent features are difficult to simulate, and that all of the major features or processes affecting a sample may not even be known, but not including such effects diminishes the realism of the simulated data, and this lack of complexity can be observed in the divergence of the performance of different quantification pipelines compared to how they performed in experimental samples. Identifying other factors present in experimental datasets but lacking in simulations, and determining how to faithfully simulate these factors, seems an important area for future research.



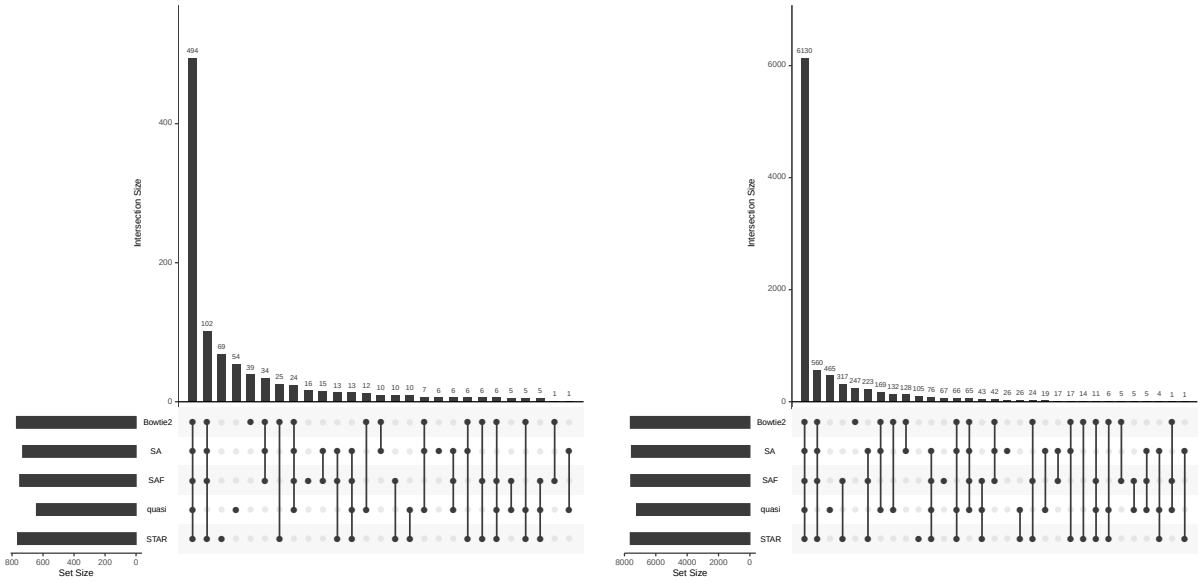
(A) Bulk



(B) Single-cell

FIGURE 3.4: **Performance of each method on bulk datasets simulated using Polyester.** The top half of the matrix shows swarm plots of pairwise correlations of counts predicted by the different approaches with each other and with the true read counts on the simulated samples generated using the experimentally-derived abundances from Bowtie2 and `Salmon`. The bottom half shows the average Spearman correlations between the different methods across the 109 simulated datasets.

3.3.6 Quantification differences can affect differential gene expression analysis



(A) (B)

FIGURE 3.5: **Differentially expressed genes predicted using each method in 2 datasets.** Comparison of sets of differentially expressed genes, and their overlaps, computed using each method. The analysis was done on 2 datasets containing multiple replicates of control and infected samples, from human ALS motor neurons (a) and HSV-1 infected cells (b). In each plot, the combination matrix at the bottom shows the intersections between the sets and the bar above it encodes the size of the intersection.

One of the most common downstream uses of transcript and gene abundance estimates is differential gene expression analysis. Errors in the transcript quantification phase can lead to incorrect detection of differentially expressed genes across conditions. Therefore, quantification is a crucial step for accurate differential gene expression (DGE) estimation and other downstream analyses. To show the influence of quantification on DGE, we performed a case study on two recently published datasets, where sequencing was done for RNA profiling of differences between the healthy and diseased samples. The first dataset is comprised of human ALS motor neurons and studies the effect of the SOD1A4V mutation (2 patient-derived samples) versus (3) isogenic controls (PRJNA236453 (Kiskinis et al., 2014)). The second dataset contained 3 replicates each of uninfected and herpesvirus (HSV-1) infected samples (PRJNA406943 (Shi et al., 2018)). Hence, the design of this study will highlight how misquantifications, possibly arising from incorrect alignments, can impact DGE analysis.

We aligned and quantified reads from all samples using the SA, SAF, quasi-mapping, STAR and Bowtie2 pipelines. The transcript-level counts were summed to the gene level

using tximport (Soneson, Love, and Robinson, 2016) and differential expression analysis was performed using DESeq2 (Love, Huber, and Anders, 2014). Genes were called as differentially expressed between the conditions, for each tool, if they had an adjusted p-value ≤ 0.01 (i.e. an FDR (false discovery rate) of 0.01 was used). The overlaps of the resulting gene sets were computed. While we had focused on transcript-level analysis in the paper until now, here we looked at differences in gene-level differential expression. This demonstrated that the quantification issues caused by lightweight mapping or misalignment of reads could be of relevance even when one is performing gene-level analyses.

The results, visualized using UpSetR (Conway, Lex, and Gehlenborg, 2017) and presented in Figure 3.5, show that lightweight mapping tended to miss the largest number of genes discovered as DE by all other approaches (i.e. the second-largest set in both examples, after the consensus set containing genes found by all approaches, was the set of genes found by all approaches except for lightweight mapping). The mean adjusted p-value of these genes under quasi-mapping was 0.034 and 0.069 in the two datasets, showing that lightweight mapping tended to deviate by a large margin from the other methods. Further, lightweight mapping also tended to discover a considerable number of distinct genes that were called as differentially expressed by this approach but not by any of the other approaches (alignment-based or selective-alignment-based), and which may have represented false positives. In each sample, the number of genes with at least one transcript shorter than 300bp constituted less than 10% of the total number of genes called differentially expressed only under the lightweight mapping based quantification, so this effect was unlikely to be driving these differences. In all cases, despite having a large overlap in DE calls with the alignment-based methods, SA produced quantifications that yielded the fewest isolated DE calls. A similar trend was observed when using an FDR of 0.05, as shown in Figure B.6(a,b) and when including `Kallisto` as a lightweight mapping method as well, as in Figure B.6(c,d). These results suggested that, when the sequenced reads tend to vary more from the reference, as might be the case in many diseased cells, lightweight mapping methods can lead to misquantifications that can eventually lead to false positives and false negatives in downstream differential gene expression studies.

3.3.7 Quantification differences can affect differential transcript expression analysis

Errors in quantification can impact differential expression analysis not just at the gene level, but at the transcript level as well. To show this impact, we selected a dataset that has recently been used to study the impact of the zika virus on the transcriptome in human neural progenitor cells (PRJNA313294 (Tang et al., 2016)). Note that, for consistency, we have used the same design as previous analyses and included both the paired-end and single-end datasets. Hence, there are 2 control and 2 infected samples under each protocol. As with the gene level analysis, the alignment and

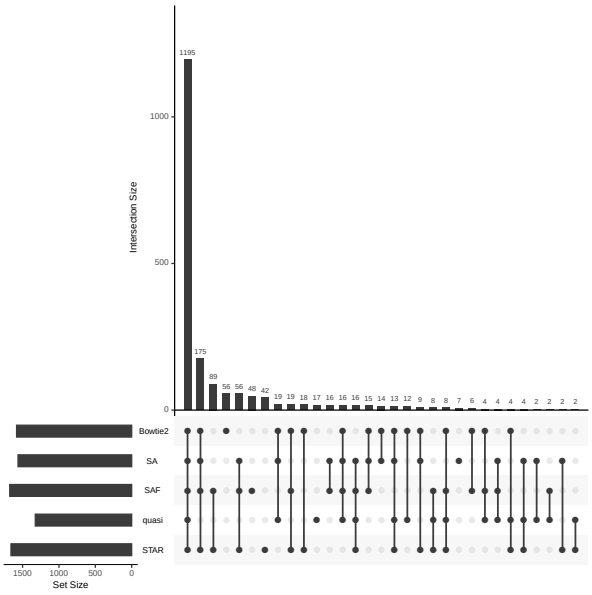


FIGURE 3.6: **Differentially expressed transcripts predicted using each method.** Comparison of sets of differentially expressed transcripts, and their overlaps, computed using each method. The analysis was done on a dataset containing multiple replicates of control and zika infected samples. In each plot, the combination matrix at the bottom shows the intersections between the sets and the bar above it encodes the size of the intersection.

quantification for all samples was done using the SA, SAF, quasi-mapping, STAR and Bowtie2 pipelines. Transcript differential expression analysis was done using sleuth (Pimentel et al., 2017) and transcripts were called as differentially expressed at an FDR of 0.01.

The results are presented in Figure 3.6 and show, as with the gene level analysis, that lightweight mapping tended to be the biggest outlier in terms of missing transcripts called as DE under all other methods. Here, we also observed a considerable set of transcripts (89) discovered only by SAF and STAR, and not by any other approach. For transcripts called as differentially expressed by the other methods but missed by quasi-mapping, the mean adjusted p-value was 0.027. At an FDR of 0.05, this increased to 0.09, highlighting the difference between the quantification estimates from lightweight mapping and alignment-based methods (results visualized in Figure B.7(a)). The results presented in Figure B.7(b) show the distribution of the DE transcripts if we included `Kallisto` as a mapping and quantification method in this analysis. As before, the lightweight mapping methods, quasi-mapping and `Kallisto`, tended to deviate from the alignment based methods. We also observed that the number of DE transcripts are the lowest under these methods.

3.4 Discussion & Conclusion

We compared and benchmarked the effects of using different alignment and mapping strategies for RNA-seq quantification, and discussed the caveats implied by different approaches. We observed that methods that perform traditional alignment of the reads against the transcriptome can produce results that are sometimes markedly different from the results produced by lightweight mapping methods. We also observed that performing spliced alignment to the genome and then projecting these alignments to transcriptome can also produce divergent results compared to directly aligning to the transcriptome.

At the same time, we proposed and benchmarked a new hybrid alignment method, SA, which provides an efficient alternative to lightweight mapping that produces results much closer to what is obtained by performing traditional alignment. This approach overcomes the shortcomings of lightweight mapping both in terms of sensitivity and specificity, as it is able to determine appropriate alignments when lightweight approaches return either suboptimal mappings or no mapping, and it is also able to better distinguish the optimal alignment loci among a set of otherwise similar sequences. Some key differences that lead to the improved accuracy of SA are an increase in mapping sensitivity (i.e. more initial mapping loci are explored), a more comprehensive and systematic mechanism for scoring potential mapping loci (making use of the match chaining algorithm of Li (2018)), and an actual alignment scoring phase that provides precise information about the quality of each retained mapping, allowing filtering out of spurious mappings that should not be reported. Moreover, the SA approach can take as input a set of decoy sequences, enabling it to avoid some of the spurious transcriptome mappings reported by Bowtie2, when, in reality, the read aligns better to an unannotated genomic locus than to the annotated transcriptome.

The results of benchmarking the different approaches on multiple simulated and experimental datasets lead to a number of conclusions. First, despite the fact that major strides have been made in improving the realism of simulated RNA-seq data, there still remain numerous ways in which simulated data fail to recapitulate the intricacies and challenges of experimental data. One of these is the fact that simulations are almost always carried out on precisely the same transcriptome that is used for quantification, while, in experimental samples, individual variation exists between the sample being assayed and the transcriptome being used for quantification. Another effect not commonly captured in simulation, but prevalent in real data, is the sequencing of reads from unannotated, alternatively-spliced transcripts, from transcripts with retained introns, from otherwise unannotated genomic loci sharing sequence-similarity with annotated transcripts, and from contamination with the sample that may share sequence similarity, to some extent, with the target transcriptome. These effects, along with others that we have not fully characterized in this manuscript, make alignment and quantification in experimental samples much more challenging than in simulated data. Hence, we observed that when quantifying across a broad sample of experimental datasets, the

quantification results obtained using different mapping and alignment approaches can demonstrate considerable variation. Together, these results suggest that quantification based purely on lightweight mapping approaches can fail to achieve the accuracy that is obtainable by the same inference algorithms when using traditional alignments, and that these errors in quantification can also affect downstream analyses, even at the gene level (as discussed in Section 3.3.6). It also suggests that there is practical room for improvement, even in the most accurate existing alignment approaches, at least for the purpose of quantifying the abundance of annotated transcripts.

While it has been previously reported (Yi et al., 2018) that pseudoalignment to the transcriptome results in comparable quantification accuracy to alignment to the genome, the analyses performed in this manuscript suggest that alignment to the transcriptome, lightweight mapping to the transcriptome, and alignment to the genome yield quantification results that are sometimes markedly different. There are a few reasons that the analyses carried out in this paper lead to a different conclusion on this question. First, the focus here is much more on experimental as opposed to simulated data. While we found that differences between lightweight mapping and alignment do exist in simulation, the magnitude of their effect on quantification is generally much smaller than is observed in experimental data. Second, while lightweight mapping to the transcriptome and alignment to the genome do yield different quantification results, we also considered traditional alignment to the transcriptome, expanding upon the different common approaches that are taken when aligning reads prior to transcript quantification. Finally, Yi et al. (Yi et al., 2018) preprocess both alignments and pseudoalignments into equivalence-class counts (the count of fragments deemed compatible with different subsets of transcripts). Then, from these reduced statistics, abundance estimation is performed. This transformation discards factors that contribute to conditional fragment assignment probabilities like alignment scores (where applicable), fragment lengths, fragment positions, etc. In the analysis presented here, we accounted for such conditional fragment probabilities in the online phase of transcript quantification, and incorporated them (approximately) into the sufficient statistics via the use of range-factorized equivalence classes (Zakeri et al., 2017). Discarding such conditional probabilities could potentially diminish true differences that exist in the underlying mappings that may, depending on the complexity of the quantification model, have an effect on quantification estimates. All of these factors may account for the sometimes considerable differences in quantification accuracy observed downstream of different lightweight mapping and alignment procedures. While we focused on quantification and differential expression, the observations made in this manuscript about the sensitivity and accuracy of different alignment approaches may extend to other downstream analyses as well, such as trans-acting expression quantitative trait locus (eQTL) detection (Saha and Battle, 2018).

Considering only the results on simulated data, one might prefer quantification based on alignment or lightweight mapping of sequencing reads directly to the transcriptome, rather than performing alignment to the genome followed by projection to the

transcriptome. One would also observe only small differences between lightweight mapping and alignment to the transcriptome. However, our analyses in experimental data suggested that the increased complexity in real RNA-seq experiments leads to more divergent behavior. In both the bulk and full-length single-cell samples analyzed, SAF yielded the highest overall correlation with the oracle, despite the fact that the oracle is derived from a combination of the Bowtie2 and STAR alignment results. Among the methods based on traditional alignment, alignment to the genome (using STAR, and projecting the resulting alignments to the transcriptome), seemed to display the best concordance, on average, with the quantifications resulting from oracle alignments. SA yielded similar but slightly better accuracy than alignment to the transcriptome using Bowtie2. This is likely, in part, because it is accounting for the sequence-similar decoys that can lead alignment to only the target transcriptome astray. The main benefit of SAF is that it aligns to a reference index that contains both the fully-spliced transcript sequences as well as the entire underlying genome (as potential decoy sequence). This allows SAF to obtain the type of sensitivity that is exhibited by approaches like Bowtie2 and SA when the read truly arises from the annotated transcriptome, but also allows it, like STAR, to avoid spuriously aligning a read to an annotated transcript when it is better explained by some other genomic locus. In the experimental data, both alignment-based approaches and selective-alignment methodologies performed better than quasi-mapping, though the manner in which these methods differ from quasi-mapping, and from each other, were not identical.

When trying to choose an approach, a choice can be made by the user performing the analysis based on any time-accuracy tradeoff they wish to make. In terms of speed, we observed that quasi-mapping is the fastest approach, followed by SA and SAF and then STAR. Bowtie2 was considerably slower than all three of these approaches. However, in terms of accuracy, we found that SAF yielded the best results, followed by alignment to the genome (with subsequent transcriptomic projection) using STAR and SA (using carefully selected decoy sequences). Bowtie2 generally performed similarly to SA, but without the benefit of decoy sequences, seemed to admit more spurious mappings. Finally, lightweight mapping of sequencing reads to the transcriptome showed the lowest overall consistency with quantifications derived from the oracle alignments. The analyses carried out in this manuscript suggest that, with respect to accurate quantification of annotated transcripts, alignment scoring is an important component, but the various pre-existing alignment approaches excelled in different cases. SA takes steps toward addressing the shortcomings of existing alignment based approaches without making large compromises on speed. This is done by indexing parts of the genome that are sequence-similar to the transcriptome or, as in the case of SAF, the entire genome in addition to the annotated transcriptome, hence exhibiting the sensitivity of Bowtie2 in transcriptomic alignment, while avoiding the spurious alignment of reads that do not truly originate from some annotated transcript, like STAR. This approach seemed to provide the highest overall accuracy, at least for the purposes of quantifying an annotated set of transcripts.

Chapter 4

Alevin (Srivastava et al., 2019a)

4.1 Background

There has been a steady increase in the throughput of single-cell RNA-seq (scRNA-seq) experiments, with droplet-based protocols (`dscRNA-seq`)(Macosko et al., 2015; Klein et al., 2015; Zheng et al., 2017) facilitating experiments assaying tens of thousands of cells in parallel. The three most widely-used `dscRNA-seq` protocols: drop-seq(Macosko et al., 2015), inDrop(Klein et al., 2015), and 10x-chromium(Zheng et al., 2017), use two separate barcodes that require appropriate processing for accurate quantification estimation. First, cellular barcodes (CBs) are used to tag each cell with a unique barcode, which enables pooling of cells for sequencing and their subsequent separation *in silico*. Thus, data processing requires the identification of the true CBs corresponding to distinct cells, and grouping the reads accordingly. Second, identification of PCR duplicates is aided by Unique Molecular Identifiers (UMIs), which tag each unique molecule prior to amplification. Since the mRNA capture rate is only around $5 - 10\%$ (Svensson et al., 2017), many rounds of PCR are typically performed prior to sequencing (Macosko et al., 2015). Appropriately accounting for the barcode information is therefore crucial for accurate estimation of gene expression. Only a minor fraction of the possible CBs present will ultimately tag a cell, and likewise, only a minor fraction of UMIs will tag unique molecules from the same gene. Thus, in each case, the aim is to identify the barcodes used. Unfortunately, both CBs and UMIs are subject to errors that occur during sequencing and amplification (Smith, Heger, and Sudbery, 2017; Macosko et al., 2015), which makes the accurate deconvolution of this information *in silico* a non-trivial task. This task is made more difficult by the amplification of background RNA from empty droplets (ambient CBs) or damaged cells.

Various methods have been proposed to correctly process `dscRNA-seq` barcodes in an error-aware manner ("whitelisting") (Zhao et al., 2017; Tambe and Pachter, 2019; Zheng et al., 2017; Petukhov, V and Guo, J and Baryawno, N and Severe, N and Scadden, DT and Samsonova, MG and Kharchenko, PV, 2018; Smith, Heger, and Sudbery, 2017), to correct sequencing errors in CBs and UMIs (Petukhov, V and Guo, J and Baryawno, N and Severe, N and Scadden, DT and Samsonova, MG and Kharchenko, PV, 2018; Smith, Heger, and Sudbery, 2017), to deduplicate UMI tags inferred to be duplicates (Smith,

Heger, and Sudbery, 2017), and to obtain cell-level gene quantification estimates(Tian et al., 2018).

TABLE 4.1: The percentage of reads multi-mapping in bulk datasets from human and mouse. We use these datasets for various analyses throughout the chapter. Note that this percentage varies depending on the read length as well as the overall quality of the dataset.

Species	Accession number	Read length	Percentage
Human	SRR1303990(Poldrack et al., 2015)	101	7.4
Human	SRR1373442(Dvinge et al., 2014)	49	9.2
Human	SRR1644186(Bouquet et al., 2016)	100	9.2
Human	SRR5074291(Shen et al., 2017)	150	7.7
Mouse	ERR435943(Schmitt et al., 2014)	75	23.0
Mouse	SRR3532922(Saito et al., 2016)	125	10.6
Mouse	SRR6753775(Fratta et al., 2018)	150	5.6
Mouse	SRR327047(Grant et al., 2011)	120	5.2

Here, we describe an end-to-end quantification pipeline that takes as input sample-demultiplexed `FASTQ` files and outputs gene-level UMI counts for each cell in the library. We call this unified pipeline `alevin`, and it overcomes two main shortcomings of traditional pipelines. First, existing techniques for UMI deduplication discard reads that map to more than one gene. In bulk RNA-seq datasets (with paired-end reads and full-length transcript coverage), the proportion of gene ambiguous reads is generally small (Table 4.1). Yet, in tagged-end scRNA-seq, this set of gene-ambiguous reads is generally larger, and commonly accounts for $\sim 14 - 23\%$ of the input data (Table 4.2). This is a result of both the fact that `dscRNA-seq` protocols, by construction, display a very strong 3′ bias and that these protocols yield effectively single-end reads (only one of the sequenced reads contains sequence from the underlying transcript), resulting in a reduced ability to resolve multimapping using a pair of reads from a longer fragment. We show that discarding the multimapping reads can negatively bias the gene-level counts predicted by various methods. Second, existing quantification pipelines combine independent processing algorithms and tools for each step, usually communicating results between pipeline stages via intermediate files on disk, which significantly increases the processing time and memory requirements for the complete analysis. We show that `alevin` makes use of more reads than other pipelines, that this leads to more accurate quantification of genes, and that `alevin` does this ~ 8 times faster and with a lower memory requirement, when compared to existing best practice pipelines for `dscRNA-seq` analysis.

TABLE 4.2: Percentage of reads multi-mapping across various scRNA-seq samples, using the `alevin` mappings.

Sample	Percentage
Human PBMC 4k	14.2
Human PBMC 8k	14.1
Mouse Neurons 900	21.8
Mouse Neurons 2k	22.7
Mouse Neurons 9k	17.2

4.2 Materials and Methods

4.2.1 Initial whitelisting and barcode correction

After standard quality control procedures, the first step of existing single-cell RNA-seq processing pipelines (Macosko et al., 2015; Klein et al., 2015; Zheng et al., 2017) is to extract cell barcode and UMI sequences, and to add this information to the header of the sequenced read or save it in temporary files. This approach, while versatile, can create many intermediate files on disk for further processing, which can be time and space-consuming.

`Alevin` begins with sample-demultiplexed `FASTQ` files. It quickly iterates over the file containing the barcode reads, and tallies the frequency of all observed barcodes (regardless of putative errors). We denote the collection of all observed barcodes as $\mathcal{B}$. Whitelisting involves determining which of these barcodes may have derived from a valid cell. When the data has been previously processed by another pipeline, a whitelist may already be available for `alevin` to use. When a whitelist is not available, `alevin` uses a two-step procedure for calculating one. An initial draft whitelist is produced using the procedure explained below, to select CBs for initial quantification. This list is refined after per-cell level quantification estimates are available (see section 4.2.5) to produce a final whitelist.

To generate a putative whitelist, we follow the approach taken by other `dscRNA-seq` pipelines by analyzing the cumulative distribution of barcode frequencies, and finding the knee in this curve (Macosko et al., 2015; Klein et al., 2015). Those barcodes occurring after the knee constitute the whitelist, denoted $\mathcal{W}$. We use a Gaussian kernel to estimate the probability density function for the barcode frequency and select the local minimum corresponding to the "knee". In the case of a user-provided whitelist, the provided $\mathcal{W}$ is used as the fixed final whitelist.

Next, we consider those barcodes in $\mathcal{E} = \mathcal{B} \setminus \mathcal{W}$ to determine, for each non-whitelisted barcode, whether a) its corresponding reads should be assigned to some barcode in $\mathcal{W}$; or b) this barcode represents some other type of noise or error (e.g., ambient RNA, lysed cell, etc.) and its associated reads should be discarded. The approach of `alevin` is to determine, for each barcode $h_j \in \mathcal{E}$, the set of whitelisted barcodes with which h_j could

be associated. We call these the putative labels of h_j — denoted as $\ell(h_j)$. Following the criteria used by previous pipelines (Macosko et al., 2015), we consider a whitelisted barcode w_i to be a putative label for some erroneous barcode h_j if h_j can be obtained from w_i by a substitution, by a single insertion (and clipping of the terminal base) or by a single deletion (and the addition of a valid nucleotide to the end of h_j). Rather than applying traditional algorithms for computing the all-versus-all edit-distances directly, and then filtering for such occurrences, we exploit the fact that barcodes are relatively short. Therefore, we can explicitly iterate over all of the valid $w_i \in \mathcal{W}$ and enumerate all erroneous barcodes for which this might be a putative label. Let $Q(w_i, H)$ be the set of barcodes from $\mathcal{E}$ that adhere to the conditions defined above; then, for each $h_j \in Q(w_i, H)$, we append w_i as putative label for the erroneous barcode h_j.

Once all whitelisted barcodes have been processed, each element in $\mathcal{E}$ will have zero or more putative labels. If an erroneous barcode has more than one putative label, we prioritize substitutions over insertions and deletions. If this does not yield a single label, ties are broken randomly. If no candidate is discovered for an erroneous barcode, then this barcode is considered "noise", and its associated reads are simply discarded. Note that, although adopted from existing methods, the `alevin` initial whitelisting process is designed to output a larger number of CBs.

4.2.2 Mapping reads and UMI de-duplication

After labeling each barcode, either as noise, or as belonging to some whitelisted barcode, `alevin` maps the sequenced reads to the target transcriptome (Srivastava et al., 2016b; Sarkar et al., 2018). Reads mapping to a given transcript (or multimapping to a set of transcripts) are categorized hierarchically, first based on the label of their corresponding cellular barcode, and then based on their unique molecular identifier (UMI). At this point, it is then possible to deduplicate reads based on their mapping and UMI information.

The process of read deduplication involves the identification of duplicate reads based on their UMIs and alignment positions. Most amplification occurs prior to fragmentation in library construction for 10x Chromium protocols (*10x-Genomics Single-Cell 3′-V2 Kit* 2018). Because of this, the alignment position of a given read is not straightforward to interpret with respect to deduplication, as the same initial unique molecule may yield reads with different alignment coordinates[2]. UMIs can also contain sequence errors. Thus, achieving the correct deduplication requires proper consideration of the available positional information and possible errors.

Our approach for handling sequencing errors, and PCR errors in the UMIs is motivated by "directional" approach introduced in UMI-tools (Smith, Heger, and Sudbery, 2017).

[2]We note that whether the majority of amplification occurs pre- or post-fragmentation can be protocol specific and can suggest different strategies for UMI deduplication. Here, we are primarily concerned with the 10X Chromium protocols, dominated by pre-fragmentation amplification. However, the method we propose for UMI deduplication can be applied to other protocols as well.

Let $\mathcal{U}_i$ be the set of UMIs observed for gene i. A specific UMI $u_n \in \mathcal{U}_i$, observed c_n times in gene i, is considered to have arisen by PCR or sequence error if there exists $u_m \in \mathcal{U}_i$ such that $d(u_n, u_m) = 1$ and $c_m > 2c_n + 1$, where $d(\cdot, \cdot)$ is, the Hamming distance. Using this information, only UMIs that could not have arisen as an error under this model are retained. However, this approach may over-collapse UMIs if there exists evidence that similar UMIs (i.e. UMIs at a Hamming distance of 1 or less) may have arisen from different transcripts, and hence, distinct molecules. Moreover, this approach first discards reads that multimap to more than one read, causing it to lose a substantial amount of information before even beginning the UMI deduplication process.

As previously proposed to address the problem of cell-clustering (Ntranos et al., 2016), an equivalence class (Turro et al., 2011; Mezlini et al., 2013; Patro, Mount, and Kingsford, 2014; Bray et al., 2016; Patro et al., 2017; Zhang and Wang, 2014; Ju et al., 2017) encodes some positional information, by means of encoding the set of transcripts to which a fragment is mapped. Specifically, these equivalence classes can encode constraints about which UMIs may have arisen from the same molecule and which UMIs — even if mapping to the same gene — must have derived from distinct pre-PCR molecules. This can be used to avoid over-collapsing UMI tags that are likely to result from different molecules by considering UMIs as distinct for each equivalence class. However, in its simplest form, this deduplication method is prone to reporting a considerably higher number of distinct UMIs than likely exist. This is because reads from different positions along a single transcript, and tagged with the same UMI, can give rise to different equivalence classes, so that membership in a different equivalence class is not, alone, sufficient evidence that a read must have derived from a distinct (pre-PCR) molecule. This deters us from directly using such a UMI collapsing strategy for deriving gene-level counts, though it may be helpful for other types of analyses.

Given the shortcomings of both approaches to UMI deduplication, we propose, instead, a novel UMI resolution algorithm that takes into account transcript-level evidence when it exists, while simultaneously avoiding the problem of under-collapsing that can occur if equivalence classes are treated independently for the purposes of UMI deduplication.

4.2.3 UMI Resolution Algorithm

A potential drawback of gene-level deduplication is that it discards transcript-level evidence. In this case, such evidence is encoded in the equivalence classes. Thus, gene-level deduplication provides a conservative approach and assumes that it is highly unlikely for molecules that are distinct transcripts of the same gene to be tagged with a similar UMI (within an edit distance of 1 from another UMI from the same gene). However, entirely discarding transcript-level information will mask true UMI collisions to some degree, even when there is direct evidence that similar UMIs must have arisen from distinct transcripts. For example, if similar UMIs appear in transcript-disjoint equivalence classes (even if all of the transcripts labeling both classes belong to the

same gene), then they *cannot* have arisen from the same pre-PCR molecule. Accounting for such cases is especially true when using an error-aware deduplication approach, and as sequencing depth increases.

To perform UMI deduplication, `alevin` begins by constructing a **p**arsimonious **U**MI **g**raph (PUG), $G = (V, E)$, for each cell, where each $v_i = (u, T_i)$ is a tuple consisting of UMI sequence u and a set of transcripts $T_i = \{t_{i_1}, t_{i_2}, \ldots, t_{i_m}\}$. There is a count associated with each vertex such that $c(v_i) = c_i$ is the number of times this UMI, equivalence class pair is observed. G contains two types of edges; directed and bi-directed. There exists a directed edge between every pair of vertices (v_i, v_j) for which $c_i > 2c_j - 1$, $|T_i \cap T_j| > 0$, and $d(\text{umi}(v_i), \text{umi}(v_j)) = 1$. For every pair of vertices for which there is no directed edge, there exists a bi-directed edge if $d(\text{umi}(v_k), \text{umi}(v_\ell)) \leq 1$ and $|T_k \cap T_\ell| > 0$. Once the edges of this PUG have been formed, we no longer need to consider the counts of the individual UMI, equivalence class pairs.

Before proceeding further, we introduce the notion of *monochromatic arborescences* in terms of this graph G. We can refer to the transcript labels of each node as the potential colors of the node. Since our graph is directed, an arborescence would be a rooted tree in the graph, where each node within the arborescence has exactly one directed path reaching it from a determined root node. Given these definitions, a monochromatic arborescence is one where the set of colors of the nodes within the arborescence have a non-null intersection and hence, the arborescence can be labeled using a single color. Then, for a given connected component in the graph, we can find different sets of monochromatic arborescences and, for our graph, each one represents a single pre-PCR molecule.

However, motivated by the principle of parsimony, we wish to explain the observed vertices (i.e., UMI, equivalence class pairs) via the minimum possible number of pre-PCR molecules that are consistent with the observed data. Hence, we pose this problem in the following manner. Given a graph G, we seek a *minimum cardinality covering by monochromatic arborescences*. In other words, we wish to cover G by a collection of vertex-disjoint arborescences, where each arborescence is labeled consistently by a set of transcripts, which are the pre-PCR molecule types from which its reads and UMIs are posited to have arisen. Further, we wish to cover all vertices in G using the minimum possible number of arborescences. Here, the graph G defines which UMI, read pairs can potentially be explained in terms of others (i.e. which vertices may have arisen from the same molecule by virtue of different fragmentation positions or which vertices may have given rise to other through PCR duplication with error). The decision version of this problem is NP-complete below, and so `alevin` employs a greedy algorithm in practice to obtain a valid, though not necessarily minimum, covering of G. We note that while numerous covering and packing problems related to arborescences have appeared in the literature (Bernáth and Pap (2011) and references therein), to the best of our knowledge, the following problem formulation is new.

Theorem 1. *Minimum cardinality covering by monochromatic arborescences is NP-complete.*

Proof: *Consider a reduction from dominating set. Let (G, k) be an instance of the dominating set problem where $G = (V, E)$ is an undirected graph. Then we can construct a new graph $G' = (V, E')$ such that G' has a minimum cardinality covering by $\leq k$ monochromatic arborescences if and only if G has a minimum dominating set of size $\leq k$. The color of an arborescence is chosen from among the intersection of the set of labels for each node it covers and, hence, is non-null. Construct G' as follows. Convert each edge in G to a bi-directed edge in G' and label each node with the union of its own label and the labels of all nodes to which it is directly connected in G. In other words, $T_i = \{i\} \cup \{j \mid \{i, j\} \in E\}$.*

$\rightarrow$ If G has a minimum dominating set of size k, then G' has a minimum cardinality covering by k monochromatic arborescences. Every node in the original graph G has to be connected to at least one node in the dominating set. Due to the manner in which node labels are assigned in G', this means that every node in G' can be covered by an arborescence starting from a dominating set node; this arborescence is colored by the label assigned to that node. Since there are k nodes in the dominating set, there will be k monochromatic arborescences in G', and since the k nodes in G dominate V, the arborescences will cover all of V.

$\leftarrow$ If G' has a covering of k monochromatic arborescences, then G has a dominating set of size k. An arborescence is assigned a color, let's say ℓ_i, from the intersection of the labels of the nodes it covers. Hence the node with label ℓ_i in G' has to be one of the nodes covered by this arborescence. That node connects to all the nodes in this arborescence, otherwise they would not have shared this label. Let these nodes be selected as the dominating set of G. Hence, if there are k arborescences, there are k such nodes that are part of the dominating set, and because the arborescences cover all of G', the selected nodes, likewise, dominate G. □

The algorithm employed by `alevin` works as follows. First, we note that weakly-connected components of G can be processed independently, and so we describe here the procedure used to resolve UMIs within a single weakly-connected component — this is repeated for all such components. Let $C = (V_C, E_C)$ denote our current component. We perform a breadth-first search starting from each vertex $v_i \in V_C$ and considering each transcript t_{i_j} (the j^{th} transcript in the equivalence class labeling vertex v_i). We compute the size (cardinality) of the largest arborescence that can be created starting from this node and using this label to cover the visited vertices. Let $v_{i'}, t_{i'_{j'}}$ be the vertex, transcript pair generating the largest arborescence and let $a(v_{i'}, t_{i'_{j'}})$ be the corresponding arborescence. We now remove all of the vertices in $a(v_{i'}, t_{i'_{j'}})$, and all of their incident edges, from C, and we repeat the same procedure on the remaining graph. This process is iterated until all vertices of C have been removed. This procedure is guaranteed to select some positive order arborescence (i.e. an arborescence containing at least one node) in each iteration, and hence is guaranteed to terminate after at most a linear number of iterations in the order of C.

After computing a covering, each arborescence is labeled with a particular transcript. However, the selected transcript may not be the unique transcript capable of producing this particular arborescence starting from the chosen root note. We can compute, for each arborescence, the *set* of possible transcript labels that could have colored it (i.e.

those in the intersection of the equivalence class labels for all of the vertices in the arborescence). If the cardinality of this set is 1, then only a single transcript is capable of explaining all of the UMIs associated with this arborescence. If the cardinality of this set is > 1, then we need to determine if all transcripts capable of covering this arborescence belong to the same gene, or whether transcripts from multiple genes may, in fact, be capable of explaining the associated UMIs. In the former case, the count of pre-PCR molecules (i.e. distinct, deduplicated UMIs) associated with this uniquely-selected gene is incremented by 1. In the latter case, the molecule associated with the arborescence is considered to potentially arise from any of the genes with which it could be labeled. Subsequently, an EM algorithm is used to distribute the counts between the genes. Note that other pipelines simply discard these gene-ambiguous reads and that both manners in which `alevin` attempts to resolve such reads (i.e. either by being selected via the parsimony condition or probabilistic allocated by the EM algorithm) are novel in the context of scRNA-seq quantification. The EM procedure we adopt to resolve ambiguous arborescences proceeds in the same manner as the EM algorithm used for transcript estimation in bulk RNA-seq data (Patro et al., 2017), with the exception that we assume the probability of generating a fragment is directly proportional to the estimated abundance, rather than the abundance divided by the effective length (i.e. we assume that, in the tagged-end protocols used, there is no length effect in the fragment generation process).

4.2.4 Tier Assignment

The `alevin` program also outputs a tier matrix, of the same dimensions as the cell-gene count matrix. Within a cell, each gene is assigned one of fours tiers. The first tier (assigned 0) is the set of genes that have no read evidence in this cell and are, therefore, predicted to be unexpressed (whether truly absent, or the effect of some dropout process). The rest of the tiers (1,2, and 3) are assigned based on a graph induced by the transcript equivalence classes as follows:

1. All equivalence classes of size 1 are filtered out. The genes associated with the transcripts from these classes are assigned to tier 1.
2. For the remaining equivalence classes, of size > 1 gene, a graph G is constructed. The nodes in G are transcripts and two nodes share an edge if their corresponding transcripts belong to a single equivalence class.
3. All the connected components in G are listed and the transcript labels on the nodes mapped to their corresponding genes. If any component contains a node whose gene has previously been assigned to tier 1, that gene and all other genes in this connected component are assigned to tier 2. Hence, tier 2 contains genes whose quantification is impacted by the EM algorithm (after the UMI deduplication).
4. Genes associated with the remaining nodes in the graph are assigned tier 3. These are genes that have no unique evidence, and do not share reads (or, in

fact, paths in the equivalence class graph) with another gene that has unique evidence. Hence, the EM algorithm will distribute reads between these genes in an essentially uniform manner, and their estimates are uninformative. Their abundance signifies that some genes (at least 1) in this ambiguous family are expressed, but exactly which and their distribution of abundances cannot be determined.

`Alevin`, optionally (using the `-numCellBootstraps` flag), also outputs bootstrap variance estimates for genes within each cell. These variance estimates could conceivably be used by downstream tools for dimensionality reduction, differential expression testing, or other tasks.

4.2.5 Final whitelisting (Optional)

Many existing tools for whitelisting CBs, such as `Cell-Ranger` (Zheng et al., 2017) and Sircel(Tambe and Pachter, 2019) perform whitelisting only once. As discussed above, both tools rely on the assumption that the number of times a CB is observed is sufficient to identify the *correct* CBs, i.e. those originating from droplets containing a cell. However, as observed by Petukhov, V and Guo, J and Baryawno, N and Severe, N and Scadden, DT and Samsonova, MG and Kharchenko, PV (2018), there is considerable variation in sequencing depth per-cell, and some droplets may contain damaged or low-quality cells. Thus, true CBs may fall below a simple knee-like threshold. Similarly, erroneous CBs may lie above the threshold. Petukhov, V and Guo, J and Baryawno, N and Severe, N and Scadden, DT and Samsonova, MG and Kharchenko, PV (2018) proposed that instead of selecting a single threshold, one should treat whitelisting as a classification problem and segregate CBs into three regions; high-quality, low-quality and uncertain / ambiguous. Here, high-quality refers to the CBs which are deemed to be definitely correct, and low-quality are the CBs which are deemed to most likely not arise from valid cells. A classifier can then be trained on the high and low-quality CBs to classify the barcodes in the ambiguous region as either high or low-quality. We adopt this approach in `alevin`, using our knee method's cutoff to determine the ambiguous region. Specifically, we divide everything above the knee threshold into two equal regions; high-quality valid barcodes (upper-half), denoted by $\mathcal{H}$, and ambiguous barcodes (lower-half), denoted by $\mathcal{L}$. Since the initial whitelisting procedure is very liberal in selecting a threshold, most of the recoverable, low-confidence CBs tend to reside in the ambiguous region, and to learn the low-quality region, we take $n_l = \max(0.2 \cdot |\mathcal{H}|, 1000)$ barcodes just below the knee threshold.

In the implementation of Petukhov, V and Guo, J and Baryawno, N and Severe, N and Scadden, DT and Samsonova, MG and Kharchenko, PV (2018), a kernel density estimation classifier was trained using features which described the number of reads per UMI, UMIs per gene, the fraction of intergenic reads, non-aligned reads, the fraction of lowly expressed genes and the fraction of UMIs on lowly expressed genes. In addition, a maximum allowable mitochondrial read content was set for a CB to be classified as

"high-quality". Whilst these features enabled the authors to build a classifier which efficiently separated "high-quality" cells from "low-quality" cells, we believe it may be possible to improve this set of features. Specifically, most of these features would be expected to correlate with the number of reads or UMIs per CB. Thus, the classifier is biased towards attributes associated with higher read depth, when in fact one wants it to learn the feature attributes associated with high-quality cells. We therefore used a slightly different set of features, listed below, which we believe may better capture the differences between high and low-quality cells. While these features work in general, they may not be suitable for all analyses and will have to be tweaked accordingly. We chose to use a naïve Bayes classifier to perform classification, since we observed no clear difference between multiple ML methods (not shown), and the naïve Bayes classifier yields classification probabilities which are easy to interpret. Our final set of whitelisted CBs are those classified as high-confidence.

1. Fraction of reads mapped
2. Fraction of mitochondrial reads (Optionally activated by `--mRNA` flag.)
3. Fraction of rRNA reads (Optionally activated by `--rRNA` flag.)
4. Duplication rate
5. Mean gene counts post de-duplication
6. The maximum correlation of gene-level quantification estimates with the high-quality CBs. (Optionally activated by `--useCorrelation` flag.)

4.3 Results

4.3.1 `Alevin` overview

There are several steps in the `alevin` pipeline that are streamlined to work without the overhead of writing to disk, as highlighted in Figure 4.1 (details in Materials and Methods). The first step is to identify the CBs that represent properly-captured and tagged cells ("whitelisting"). `Alevin` uses a two-step whitelisting procedure, where the second step takes place at the end of the pipeline. An initial whitelist is produced by finding the "knee" in the cumulative distribution of CB frequencies(Macosko et al., 2015; Zheng et al., 2017). For each non-whitelisted CB, `alevin` tries to correct it to a whitelisted CB either by a substitution, or a single insertion or deletion. If no such barcode exists in the set of whitelisted barcodes, the barcode and its associated reads are discarded. The next step is mapping reads from the tagged with the whitelisted CBs to a target transcriptome(Srivastava et al., 2016b; Sarkar et al., 2018), followed by UMI deduplication.

The process of deduplication requires identifying duplicate reads based on their UMIs and alignment positions along the transcriptome. `Alevin` uses a novel algorithm for

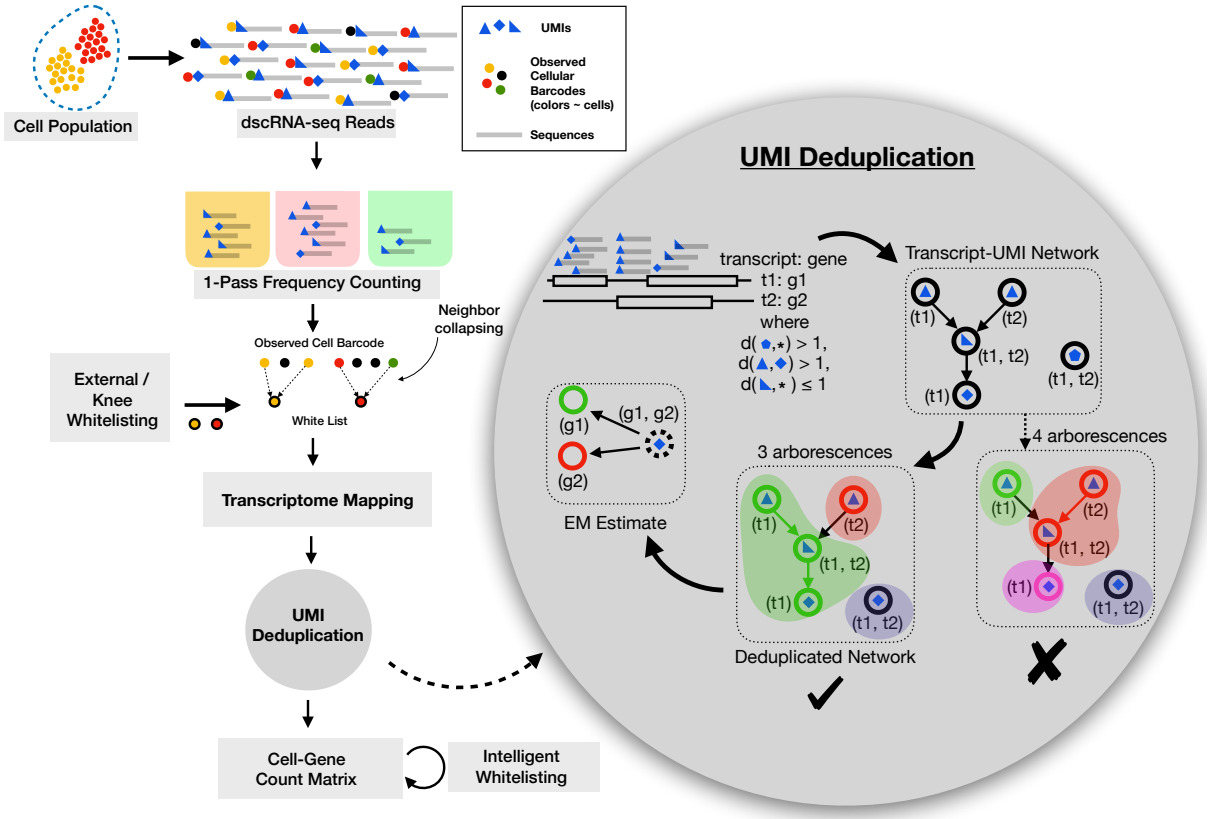


FIGURE 4.1: Overview of the `alevin` pipeline. The input to the pipeline are sample-demultiplexed `FASTQ` files and there are several steps, outlined here, that are required to process this data and obtain per cell gene-level quantification estimates. The first step is cell-barcode (CB) whitelisting using their frequencies. Barcodes neighboring whitelisted barcodes are then associated with (collapsed into) their whitelisted counterparts. Reads from whitelisted CBs are mapped to the transcriptome and the UMI-transcript equivalence classes are generated. Each equivalence class contains a set of transcripts, the UMIs that are associated with the reads that map to each class and the read count for each UMI. This information is used to construct a parsimonious UMI graph (PUG) where each node represents a UMI-transcript equivalence class and nodes are connected based on the associated read counts. The UMI deduplication algorithm then attempts to find a minimal set of transcripts that cover the graph (where each consistently-labeled connected component — each monochromatic arborescence — is associated with a distinct pre-PCR molecule). In this way, each node is assigned a transcript label, and in turn, an associated gene label. Reads associated with arborescences that could be consistently labeled by multiple genes are divided amongst these possible loci probabilistically based on an expectation-maximization algorithm. Finally, optionally, and if not provided with high quality CB whitelist externally, an intelligent whitelisting procedure finalizes a list of high quality CBs using a naïve Bayes classifier to differentiate between high and low-quality cells.

deduplication that begins by constructing parsimonious UMI graphs, that we refer to as a PUGs, using information from the UMI sequences, the UMI counts and the transcript equivalence classes (Turro et al., 2011). This PUG is constructed such that each UMI-transcript equivalence class pair is represented by a node and there exists an edge from a node to any node that could have arisen from an amplified molecule due to sampling the underlying transcript (a single pre-PCR molecule) at a different position, or via a PCR or a sequencing error being introduced into the UMI. When the direction of "duplication" during PCR is clear, a directed edge is added, otherwise a bi-directed edge is placed. An optimal covering of this graph, using the transcripts associated with each node, will give the minimum number of UMIs, along with their counts, required to explain the set of mapped reads. Hence, we have mapped the deduplication problem to that of finding a minimum cardinality covering of a given graph by monochromatic arborescences. Since the decision version of this problem is NP-complete, we propose a greedy algorithm to obtain a minimum cardinality covering of this graph (proof and algorithm detailed under Materials and Methods). Each covering, and the associated UMI, is assigned a set of transcript labels of size ≥ 1. After this UMI resolution phase, the remaining ambiguous reads with more than 1 transcript label, are assigned based on an expectation-maximization method (Patro et al., 2017).

Finally, having obtained per-cell gene expression estimates, CB whitelisting is finalized using a naïve Bayes classifier to differentiate between high and low-quality cells utilizing a set of features derived from the expression estimates and other diagnostic features (Petukhov, V and Guo, J and Baryawno, N and Severe, N and Scadden, DT and Samsonova, MG and Kharchenko, PV, 2018). In addition to the gene-by-cell count matrix, `alevin` also provides information about the reliability of the abundance estimate computed for each gene in each cell in the form of a *tier* matrix (and, optionally, the summarized variance of bootstrap estimates), which succinctly encodes the quality of the evidence used to derive the corresponding count.

4.3.2 Impact of discarding multimapping reads

Before proceeding with a more detailed analysis of the `alevin` pipeline, it is important to highlight scenarios where existing pipelines would fail using simple examples. These also lead to a better understanding of the `alevin` UMI deduplication algorithm that intelligently utilizes transcript-level information to obtain accurate gene-level estimates. Since current deduplication methods do not have a mechanism to detect UMIs that map between multiple transcripts of the same gene, they can, in certain cases, incorrectly detect PCR duplicates and hence, under-estimate the total UMI counts. Some obvious cases can be resolved by considering the read-to-transcript mapping, instead of the read-to-gene mapping, as done in `alevin` and shown in the left panel in Figure 4.2. The first row (top to bottom) demonstrates a case when we observe the same UMI (U1) being used to tag transcripts from two separate genes (G1 and G2). Here, all methods are able to correctly assess that these instances of U1 are not PCR duplicates. In the center row, we observe the same UMI deriving from two (sequence-distinct) transcripts

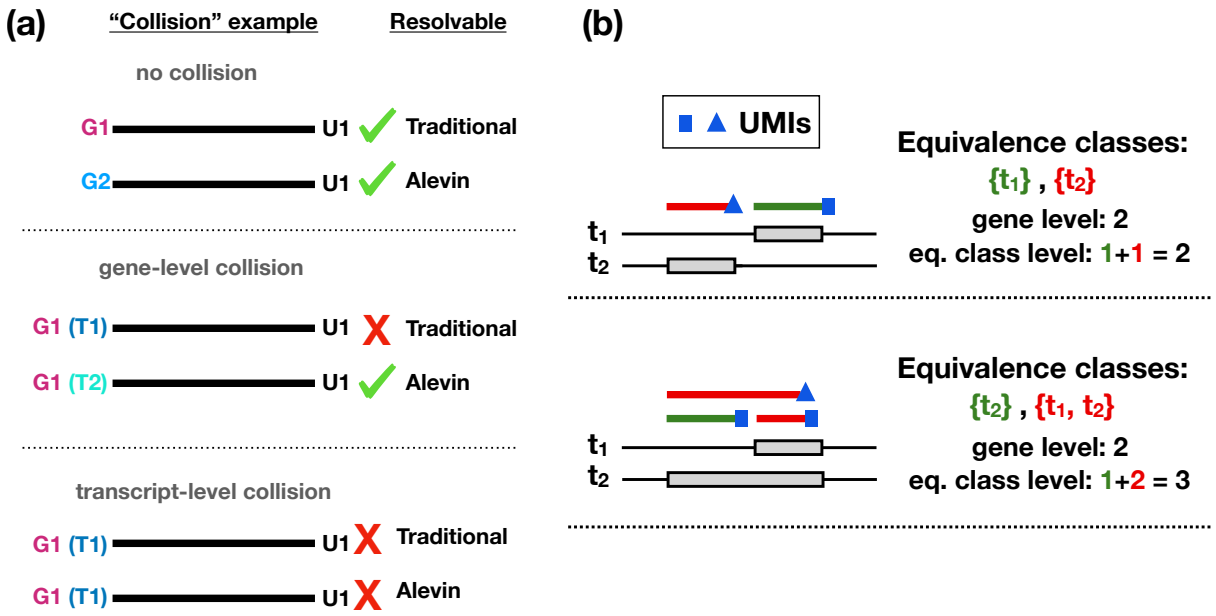


FIGURE 4.2: (a) This figure illustrates examples of various classes of UMI collisions, and which method(s) would be able to correctly resolve the origin of the multimapping reads in each scenario. These cases are shown top-to-bottom in order of their likelihood. (b) A simulated example demonstrates how treating equivalence classes individually during UMI deduplication can lead to under-collapsing of UMIs compared to gene-level methods (especially in protocols where the majority of cDNA amplification occurs prior to fragmentation). In the first row, both methods report correctly a single UMI. In the second row, there are two fragmented molecules aligned against two transcripts from the same gene. The `alevin` deduplication algorithm will attempt to choose the minimum number of transcripts required to explain the read mappings and hence correctly detect the UMI counts. The equivalence class method will over-estimate the gene count.

of the same gene. Here, purely gene-level methods fail to resolve this collision, while `alevin`'s strategy can. Finally, in the bottom row, we observe a UMI collision within a single transcript. That is two different copies (molecules) of the same transcript have been tagged with the same UMI. This resolution cannot be resolved by any of the methods. Though possible, the situation presented in the third row is *highly*-unlikely, especially given current sequencing depths.

A second scenario is highlighted in the right panel of Figure 4.2 where using the transcript level equivalence classes lead to over-counting UMIs (discussed further in Materials and Methods). In these simulated examples, different types of transcripts, and corresponding expression patterns are shown. Reads are randomly sampled from the 3'-end of the annotated-transcript(s) according to a realistic fragment length distribution, where exon overlap induces the corresponding equivalence classes of each fragment. The top simulation shows 1 (pre-PCR) molecule expressed for each transcript, identifiable by a unique-id (UMI), shown in blue. Due to the disjoint equivalence classes, both methods will correctly assign the gene count. In the bottom simulation, both molecules originate from the second transcript. However, since the equivalence classes are different, the two fragments sharing a UMI will not be collapsed. Specifically, as the rate of splicing (and hence the number of equivalence classes) increases, so too does the number of distinct UMIs reported. In this case, the `alevin` UMI deduplication algorithm will correctly detect the number of transcripts in order to greedily assign the minimum number of transcripts required to explain the given UMI and mapping information.

To show that the UMI deduplication algorithm from `alevin` does, indeed, perform better, we calculate the ratio of the number of reads mapping to each gene and the final count of UMIs as predicted by `alevin` and `Cell-Ranger` for that gene. When a read maps ambiguously, the count is divided uniformly between the genes. Hence, if a read maps to two genes, the count for each is incremented by 0.5 to get the initial number of reads mapping to these genes. Note that the mappings are also different under each pipeline and that some reads may be inherently ambiguous under one or both mappings. These reads cannot be accurately assigned but, while `Cell-Ranger` discards them, `alevin` assigns them to a gene via the PUG-resolution algorithm, or,in the case that parsimony fails to distinguish a single best gene, proportionally to multiple genes according to the other uniquely-mapping reads of the experiment. We divide the genes into 20 bins, based on the number of k-mers shared across genes. We expect the above calculated ratio to remain fairly consistent across these 20 bins, irrespective of the sequence properties of the genes in them. However, we observe in Figure 4.3, that the predictions from `Cell-Ranger` are biased for the genes with low sequence uniqueness. This is because a large number of reads from these genes will multimap across genes and will, therefore, be discarded. Hence, simply discarding multimapping reads seems to bias the count estimates for all genes but strongly impacts counts for genes that are expected to have a larger number of multimapping reads due to their high sequence similarity.

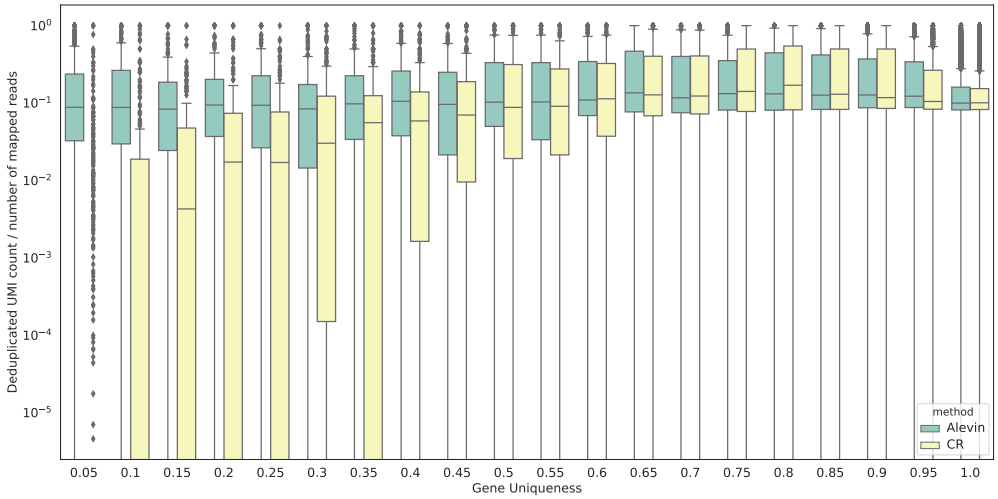


FIGURE 4.3: The ratio of the final number of deduplicated UMIs against the number of initial reads for both `alevin` and Cell Ranger (on the human PBMC 4k dataset) stratified by gene-level sequence uniqueness. The genes are divided into 20 equal sized bins and the x-axis represents the maximum gene uniqueness in each bin. The plotted ratio for genes that have high sequence similarity with other genes is strongly biased when using `Cell-Ranger`. This is because `Cell-Ranger` will discard a majority (or all) of the reads originating from these genes since they will most likely map to multiple positions across various genes. `Alevin`, on the other hand, will attempt to accurately assign these reads to their gene of origin. This plot also demonstrates that `alevin` does not over-count UMIs, which would be the case if deduplication was done at the level of equivalence classes.

4.3.3 Accuracy analysis on real datasets

TABLE 4.3: Number of final whitelisted cellular barcodes output by `alevin` and `Cell-Ranger`.

Dataset	`Cell-Ranger`	`Alevin`	No.of reads
Human PBMC 4k	4346	4341	379,462,522
Human PBMC 8k	8379	8291	784,064,148
Mouse Neurons 900	933	1291	52,805,264
Mouse Neurons 2k	2009	1881	147,010,995
Mouse Neurons 9k	9116	8519	383,366,284

To assess the performance of `alevin`, both in terms of accuracy in quantification and resource consumption, we ran it on 10x Chromium datasets from human and mouse. We compare our results against the `Cell-Ranger` pipeline(Zheng et al., 2017), the `dropEst` pipeline(Petukhov, V and Guo, J and Baryawno, N and Severe, N and Scadden, DT and Samsonova, MG and Kharchenko, PV, 2018) [1]. , and a custom pipeline, with an external list of whitelisted CBs, using STAR(Dobin et al., 2013), featureCounts (Liao, Smyth, and Shi, 2013a), and UMI-tools (Smith, Heger, and Sudbery, 2017), which we refer to as the *naïve* pipeline. The exact parameters for running each tool are provided under Materials and Methods. Note that we run `alevin` with the `-keepDuplicates` flag during indexing, which ensures that even when multiple sequence-identical transcripts exist in the annotation, they are not discarded. This is to allow for fair comparison against the other tools, since they do not discard such transcripts, and the existence of such transcripts will impact the number of multimapping reads. However, we do not generally recommended using this flag when running `alevin`. We observe that the number of final whitelisted cells predicted by `alevin` are in close proximity to the count of cells predicted by `Cell-Ranger` (and `dropEst`, since they use the same whitelise), but there are non-trivial differences (Table 4.3). Comparison on data using the Drop-seq(Macosko et al., 2015) protocol is also detailed below. Comparisons against the recently released version 3.0.0 of `Cell-Ranger` are also provided (Additional file 1: Figure S1), along with results from another run of `alevin` using different parameters. Where mentioned, the results are stratified by gene uniqueness which is the proportion of k-mers, of size 31, that are not shared between two or more genes. We note that varying the k-mer size changes the stratification of the genes but does not impact the overall correlation and performance of the methods. We show this for the mouse neuronal 900 dataset (Additional file 1: Figure S2). We calculated this for each gene in the human (GENCODE release 27, GRCh38.p10) and mouse (GENCODE release M16, GRCm38.p5) transcriptomes. Note that this was not calculated using the canonicalized k-mers from the genes. This is because the scRNA-seq protocols are stranded and a read, therefore, can not multi-map

[1] Note that we were not able to run the dropEstr Bayesian correction method and the results presented are after running just the `dropEst` pipeline (*Pipeline for initial analysis of droplet-based single-cell RNA-seq data* 2018).

between two genes if the reverse complement of one of them is shared with the other's forward sequence.

4.3.4 Accuracy of estimates against bulk data

TABLE 4.4: Average Spearman correlation of gene-level estimates from each method for the single cell datasets against bulk data from the same cell types (4 for human, 3 for mouse).

Dataset	`Alevin`	`Cell-Ranger`	*naïve*	`dropEst`
Human PBMC 4k	0.813	0.780	0.747	0.783
Human PBMC 8k	0.810	0.772	0.740	0.776
Mouse Neurons 900	0.812	0.773	0.761	0.779
Mouse Neurons 2k	0.822	0.781	0.767	0.784
Mouse Neurons 9k	0.831	0.796	0.776	0.803

To test the accuracy of the quantification estimates, we aggregate the estimates from each of the single-cell quantification tools (summing across all cells) and calculate the correlation with estimates predicted by RSEM(Li and Dewey, 2011) (paired with Bowite2(Langmead and Salzberg, 2012) alignments) using bulk datasets from the same cell types. While the differences between single-cell and bulk sequencing protocols and techniques are significant, we believe that, in the absence of established benchmarks, the correlation between them is a reasonable indicator of the accuracy of each quantification method. Estimates from `alevin`, when summed across all cells, have a higher Spearman rank correlation than the `Cell-Ranger`, `dropEst`, and *naïve* pipelines (Table 4.4). Specifically, we posit that the methods demonstrate a strong and persistent bias against groups of two or more genes that exhibit high sequence similarity. That is, the more sequence-similar a gene is to another gene, the less likely these pipelines are able to assign reads to it — in the extreme case, some genes essentially become *invisible* due to the *in silico* biases of these approaches (a similar effect was reported by Robert and Watson (Robert and Watson, 2015) in bulk RNA-seq data when simple read-counting approaches are used for quantification, where they highlight that many such genes are relevant to human disease).

To further explore this hypothesis, we stratified the accuracy of the different methods by the uniqueness of the underlying genes (Figure 4.4a, Table 4.5). The bar plots at the top of each subfigure represent the tiers of the genes as assigned by `alevin`. Tier 1 is the set of genes where all the reads are uniquely mapping. Tier 2 is genes that have ambiguously mapping reads, but connected to unique read evidence as well, that can be used by the EM to resolve the multimapping reads. Tier 3 is the genes that have no unique evidence and the read counts are, therefore, distributed between these genes according to an uninformative prior. In agreement with the hypothesized relationship, we observed that the higher accuracy of `alevin` is particularly large for genes with a lower proportion of unique k-mers, that tend to belong to tier 2 or

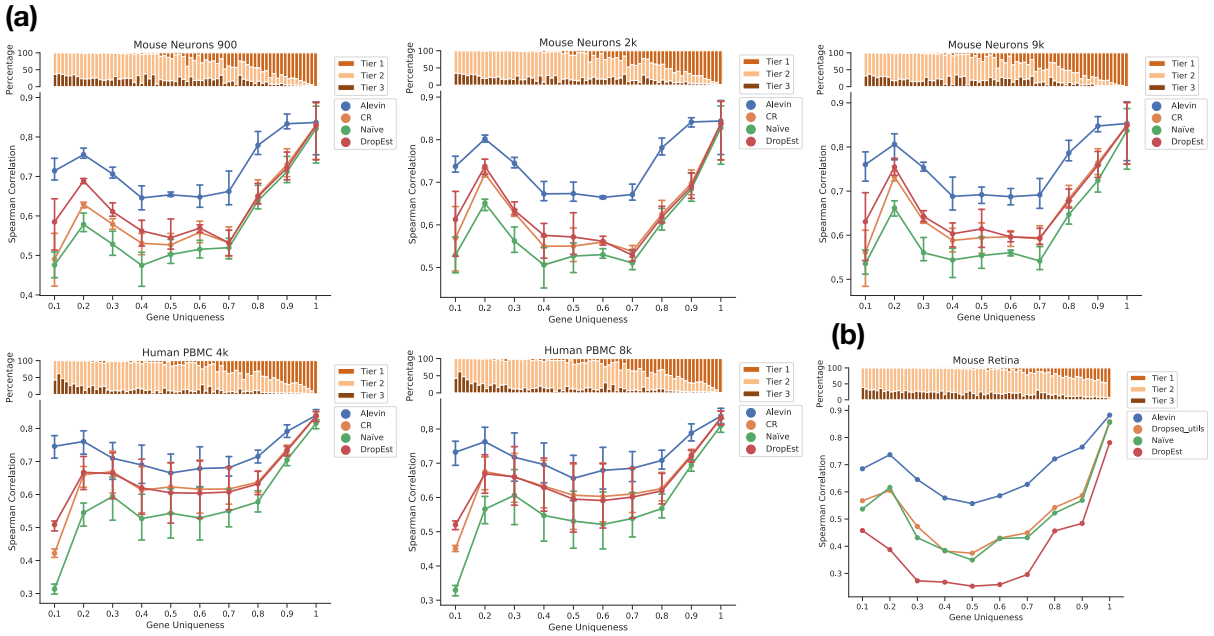


FIGURE 4.4: (a) The Spearman correlation between quantification estimates (summed across all cells) from different scRNA-seq methods against bulk data from the mouse neuronal and human PBMC datasets, stratified by gene sequence uniqueness. The bar plot on the top of each figure shows the percentage of genes in each bin that have unique read evidence. Tier 1 is the set of genes with only uniquely mapping reads. Tier 2 is genes that have ambiguously mapping reads, but are connected to unique read evidence that can be used to resolve the multimapping reads. Tier 3 is genes that are completely ambiguous. Note that all methods perform very similarly on genes from tier 1, but the performance of `alevin` is much better for the other tiers. (b) Comparison of various methods used to process dropseq data from mouse retina with 4k cells. The Spearman correlation is calculated against bulk quantification estimates predicted using Bowtie2 and RSEM on data from the same cell type.

TABLE 4.5: Number of genes in each bin, when stratified by gene uniqueness.

Bin number	Human	Mouse
1	3155	4786
2	894	1089
3	853	945
4	822	1061
5	962	1104
6	1174	1318
7	1565	1476
8	2546	1877
9	4695	2960
10	41622	36763

3. On genes from tier 1, all the methods to perform similarly. Thus, the approach of `Cell-Ranger`, `dropEst`, and *naïve*, which discard reads mapping to multiple genes, results in systematic inaccuracies in genes which are insufficiently unique (i.e. which share a high degree of sequence homology with some other gene).

This bias could impact the expression estimates of important marker genes, such as the genes for the hemoglobin alpha and beta proteins in the mouse neurons(Han et al., 2018; Richter et al., 2009). Due to their lower uniqueness ratio, `Cell-Ranger` appears to exhibit a bias against such genes, and their expression, as predicted by `alevin`, is systematically higher (Figure 4.5). Anecdotally, we also noticed that, in the human PBMC data, `alevin` sometimes predicts the expression of even relatively sequence-unique genes, like YIPF6, that we expect to be expressed in a subpopulation of these cells (monocytes) (Nakaya et al., 2011), but which exhibit almost no expression as predicted by `Cell-Ranger` (Figure 4.6). Because the bias against sequence-ambiguous genes is fundamental and sequence-specific, it cannot be easily remedied with more data, but instead requires the development of fundamentally novel algorithms, like `alevin`, that account for, rather than discard, reads mapping to such genes. Hence, `alevin` not only quantifies a greater proportion of the sequenced data than existing methods, but also does so more accurately and in a less-biased manner.

4.3.5 Accuracy of estimates using combined genomes

To further assess the accuracy of quantification estimates, in the absence of any established read-level simulation protocol, we performed an experiment aimed to introduce controlled gene-level multimapping to analyze its effect on the different methods. We quantified the mouse neuronal 900 sequencing dataset using both `Cell-Ranger` and `alevin` and each quantification was performed under two separate references: the mouse genome, and the combined human and mouse genome. Noting that the reads in this experiment originate from mouse, we desire that the quantifications returned

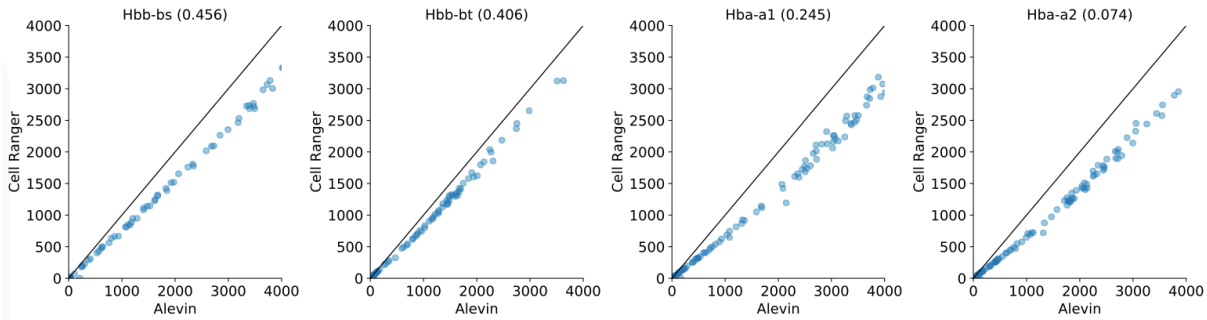


FIGURE 4.5: Expression of the Hba and Hbb genes as predicted by `alevin` and `Cell-Ranger` in mouse neuronal cells. The title of each plot is the name of the gene and its k-mer uniqueness ratio. Note that `Cell-Ranger` systematically underestimates the expression of these genes compared to `alevin`. This bias is greater for the Hba genes, which have a lower uniqueness ratio, and therefore, a greater number of multi-mapping reads.

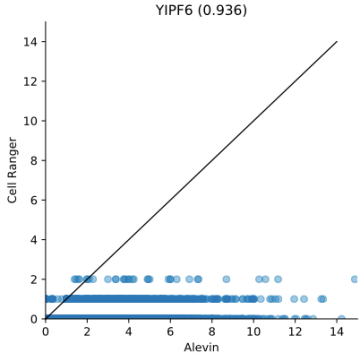


FIGURE 4.6: Expression of the YIPF6 gene (which has a high uniqueness ratio) as predicted by `alevin` and `Cell-Ranger` in the PBMC8k data.

by a method deviate as little as possible under the two different reference configurations. Under ideal conditions, for example, the gene counts under both references should be the same. However, combining the mouse and human references increases the gene sequence ambiguity, due to the presence of homologous genes, resulting in misestimation.

We show in Figure 4.7a that the distance under the two references is higher for the `Cell-Ranger` estimates than that for the `alevin` estimates. Due to the increased homology among genes between the references, the ratio of reads mapping to multiple genes increases, resulting in more information being discarded by `Cell-Ranger`. The total number of UMIs accounted for by `Cell-Ranger` decreases by $\sim 20,000$, in comparison the number of distinct UMIs predicted by `alevin` decreased by $\sim 1,500$, which one might attribute to changes in the underlying PUGs as a result of mapping $\sim 0.01\%$ more reads. The number of human genes expressed (non-zero UMI count) under the joint reference is 624 for `Cell-Ranger` and 600 for `alevin`, out of a total of $58,288$ genes. Note that in both cases, these genes account for $< 0.05\%$ of the total UMI count predicted by each method.

To provide a statistical analysis of the differences observed for the methods under the two different reference sequences, we performed the following test. We sample, randomly, 1000 sets of 100 cells from the entire experiment, and for each sample, we compute the sum of absolute difference between the predictions of each tool under both references. We compare the resulting distribution of differences for `Cell-Ranger` with that of `alevin` and find that the differences in `alevin`'s quantifications are smaller than those of `Cell-Ranger` ($p < 0.001$, Mann Whitney Wilcoxon test). These distributions are plotted in Additional file 1: Figure S3.

We also show in Figure 4.7b that, for the genes that have sequence similarity in the joint reference but are unique in the mouse genome, `Cell-Ranger` expression estimates vary much more than those from `alevin`.

4.3.6 Time and memory efficiency

The time and memory requirements for `alevin` are significantly less than those for the existing pipelines (Figure 4.8), where all methods were run using 16 threads. `dropEst` is excluded from the figure since it consumes the BAM file output by `Cell-Ranger` and is not a complete end-to-end pipeline. For the smallest dataset (900 mouse neuronal cells), `alevin` was ~ 5 times faster than *naïve* and ~ 21 times faster than `Cell-Ranger`. This difference increases further as the size of the dataset increases, since the performance of `alevin` scales better than the other tools. Hence, where `alevin` took only 70 minutes to process the human PBMC 8k dataset, `Cell-Ranger` took 22 hours and *naïve* took 11 hours. On this dataset, `dropEst` took ~ 2 hours, after `Cell-Ranger` was used to process and align the reads. In terms of memory, `alevin` used only ~ 13GB on the human PBMC 8k cell dataset, whereas *naïve* took ~ 20GB and `dropEst` took ~ 32GB. For the mouse neuronal 9k cell dataset, `alevin` used ~ 14GB, *naïve* ~ 18GB,

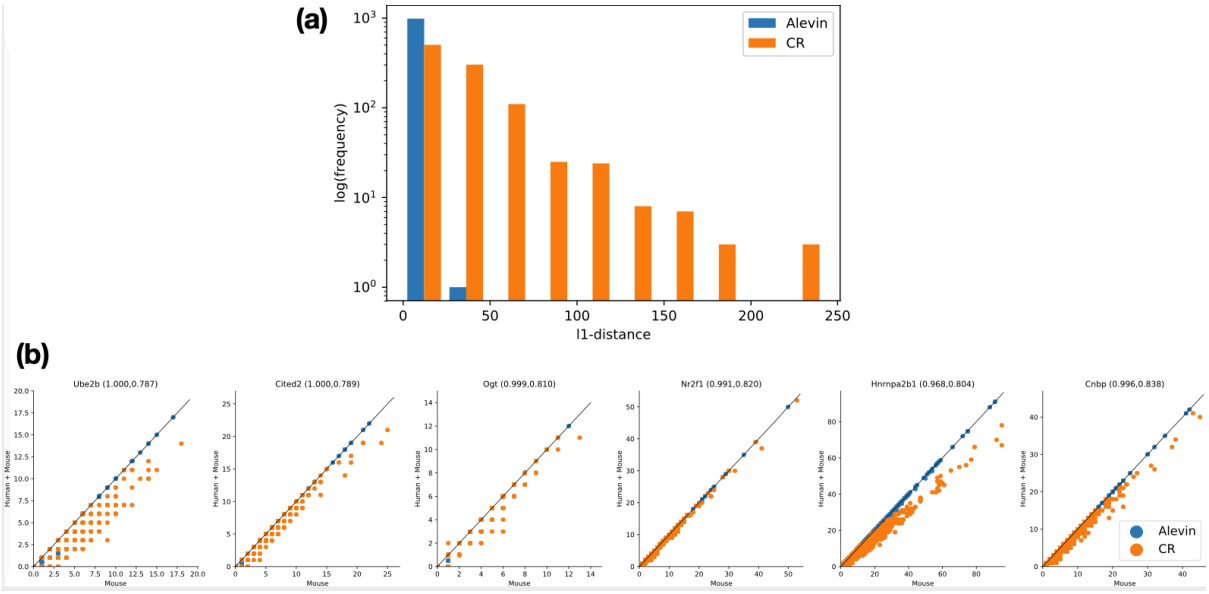


FIGURE 4.7: (a) Histogram of the ℓ_1 distance between the quantification estimates of tools on the mouse neuron 900 data, when run using different references for quantification (just mouse versus mouse and human). Results are presented for both `alevin` and `Cell-Ranger`. Since, in reality, all reads are expected to originate from mouse, deviations from quantifications under the only mouse reference signify misestimation — often due to the introduction of sequence-similar genes in the human genome. `Alevin` is able to resolve this ambiguity well, while `Cell-Ranger` instead discards such reads, leading to different quantification estimates under the two references. (b) Counts for the topmost genes that have high sequence homology between human and mouse but are sequence unique in the mouse reference. The title of each plot is the gene name along with the sequence uniqueness ratio under just the mouse reference and under the joint reference. Hence, the `Cell-Ranger` counts decrease across cells when the gene uniqueness decreases. Note that these genes were filtered such that they have >100 count difference for either `alevin` or `Cell-Ranger` when summed across all cells.

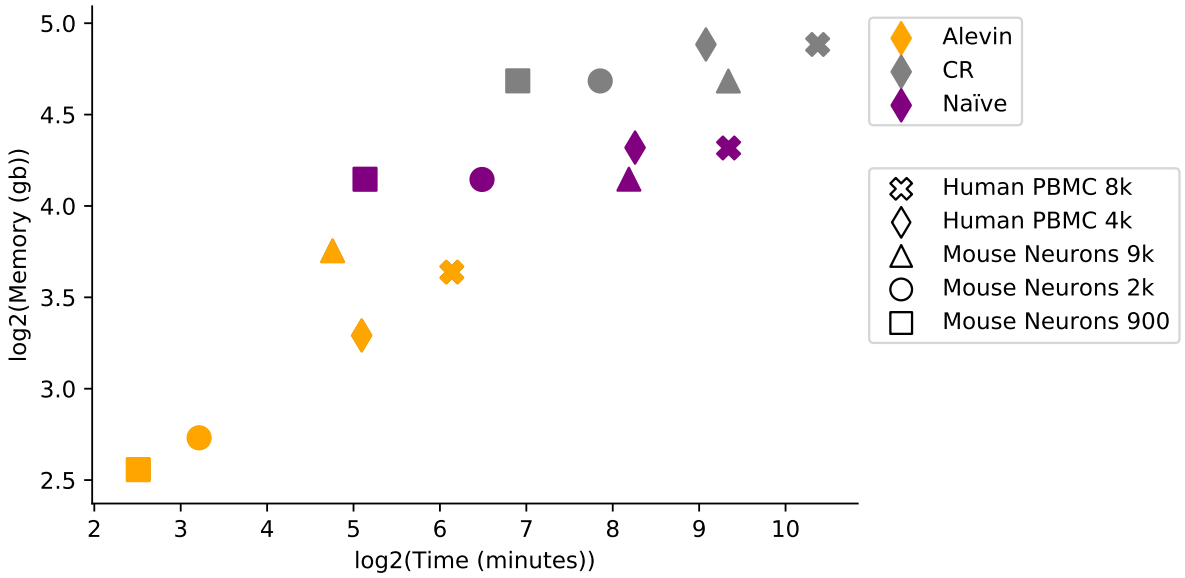


FIGURE 4.8: The time and memory performance of the different pipelines on the five datasets. `Alevin` requires significantly less time and memory than the other pipelines. Note that for `Cell-Ranger`, the memory plotted is the lower bound, which is the size of the index and the actual memory usage can be much higher.

and `dropEst` $\sim$ 52GB. In both cases, `Cell-Ranger` required a minimum of 16GB just for STAR indexing. We note that `Cell-Ranger` allows the user to specify a maximum resident memory limit, and we ran `Cell-Ranger` allowing it to allocate up to 120GB so that the extra runtime was not due to limitations in available memory. We also note that for `dropEst`, we were not able to run the Bayesian collision correction algorithm implemented in dropEstr; however, given the relatively long UMI tags employed in chromium V2 chemistry compared to inDrop, one would expect the effect of this extra phase to be limited anyway.

We observe that the optimal number of threads for running `alevin` is $10 - 12$, where the maximum gain in terms of time and memory is achieved. `Alevin` is designed to make efficient use of multiple threads, though the optimal number of threads can depend on many factors, such as the speed of the underlying disk and the size of the raw input and output matrix to be written. While runtime decreases with the number of threads uses, the memory profile changes very little as threads are added.

4.3.7 Comparison on DropSeq data

In addition to data generated using the 10x chromium protocol(Zheng et al., 2017), we also tested `alevin` on mouse retina data generated using the Drop-seq protocol (Macosko et al., 2015). We compare `alevin` against UMI-tools (the *naïve* pipeline from

the main paper), `dropEst`, and dropseq_utils (Macosko et al., 2015) — the processing pipeline originally used by Macosko et al. (2015). Again, we compared the correlation of gene abundances, summed across all cells and as produced by the different methods with the estimates from bulk data (Grant et al., 2011) in the same tissue (Figure 4.4b). We observe a similar trend across gene-uniqueness bins as was observed for the 10x datasets. `Alevin` demonstrates higher correlation, overall, with the bulk data, and the improvements are particularly substantial for genes that are not sequence-unique. Further, `alevin` is much faster, and takes less memory than the other pipelines. `Alevin` took 17 minutes to process this data, which is much faster than the UMI-tools-based pipeline ($\sim$ 3.2 hours), the dropseq_utils-based pipeline ($\sim$ 15.5 hours), and even `dropEst` (25 minutes). The memory usage of `alevin` was 6.5GB, which is less than half the memory usage of the closest tool (UMI-tools at 17.72GB). The dropseq_utils-based pipeline took 25.07GB and while `dropEst` used 10.8GB, that does not include the memory consumed by `Cell-Ranger` to index the reference and align reads against it to produce the BAM file. While `alevin` has been primarily designed and tested with 10x data in mind, the method is generic for droplet-based tagged-end protocols, and we observe that it also seems to perform well on Drop-seq data.

4.4 Conclusion

We present a new end-to-end pipeline for performing gene-level quantification from `dscRNA-seq` that is accurate, efficient, and easy to use. Our method, `Alevin`, relies on a new formulation of the UMI-resolution problem that both accounts for transcript-level constraints on how UMIs may have been generated and that allows resolving the potential origin of a UMI even when the corresponding reads map between multiple genes.

Our analyses demonstrate that, compared to `Cell-Ranger` (and *naïve*), `alevin` achieves a higher accuracy, in part because of considering a substantially larger number of reads. Further, `alevin` is considerably faster and uses less memory than these other approaches. These speed improvements are due to a combination of the fact that `alevin` uses bespoke algorithms for CB and UMI edit distance computation, read mapping, and other tasks, and is a unified tool for performing all of the initial processing steps, obviating the need to read and write large intermediate files on disk. These optimizations make it possible to efficiently process `dscRNA-seq` datasets on commodity computers reducing computational barriers to processing and re-processing of such data.

Chapter 5

Alevin 2

5.1 Background

RNA-seq quantification has been shown as an important tool to explore the genome-wide quantification of gene expressions for both bulk and single-cell RNA-seq experiments. `dscRNA-seq` experiments have been especially useful, as with the advancement in droplet-based sequencing technologies (Macosko et al., 2015; Klein et al., 2015; Zheng et al., 2017), it can generate data with high quantitative accuracy, sensitivity, and throughput. In Chapter 4, we discussed a principled framework for generating cell-wise gene-expression estimates given the read sequences from a `dscRNA-seq` experiment. We showed how discarding gene-multi mapping reads, typically done by all the existing `dscRNA-seq` quantification pipelines, can lead to biased expression estimates. Specifically, after the UMI resolution phase, we assign the ambiguous reads with more than 1 gene label to genes based on an expectation-maximization (EM) algorithm. Although the framework works well in most of the cases, in situations where there is no unique read evidence to confidently disambiguate the read assignment among genes, which we refer to as tier-3 estimates, the likelihood estimator uniformly divides the reads across the genes in the equivalence class label. In this study, we propose the idea of information sharing across closely related cells using Bayesian priors for specifically improving these tier-3 estimates. Since we expect cells in an experiment to fall into categories of specific cell-types, and cells within a cell-type to have similar expression patterns, we propose that sharing information across neighboring cells in the expression space will improve quantification estimates of the individual cells. We show that our idea of information sharing using Bayesian priors to improve quantification can be successfully applied to a `dscRNA-seq` experiment and results in higher accuracy estimates.

5.2 Material and Methods

5.2.1 Variational Bayesian `dscRNA-seq` quantification

Similar to `Salmon`'s (Patro et al., 2017) collapsed Variational Bayesian (VB) optimization objective, we aim to quantify the expression, given a set of known genes $\mathcal{G}$ and a set of gene-level equivalence classes with their UMI counts, generated post UMI deduplication using `alevin`'s framework, as $\mathcal{E}$. Originally, `alevin` takes a maximum-likelihood based approach to optimize the gene-level ambiguous read assignment objective. However, it cannot utilize the confidence from neighboring cells. Since a high level of sparsity is an inherent property of a `dscRNA-seq` experiment and due to the random process of capturing RNA molecules, in expectation, molecule sampling noise wouldn't be uniform across all cells. We expect that sharing confidence in the expression estimates across cells can be particularly effective in improving cell-level expression.

`Salmon` is a bulk RNA-seq quantification tool that uses Bayesian priors to improve the quantification estimates by its online learning phase. We use the relaxed version of the algorithm to improve the estimates of `alevin`. Specifically, if we define gene-UMI count assignment matrix as $\mathcal{Z}$, where based on $\mathcal{E}$, $z_{ij} = 1$ if UMI j is derived from gene i. We also define the probability of generating a molecule from a particular gene as ρ (analogous to nucleotide fraction in `Salmon` model); we can write the probability of observing a set of deduplicated UMIs $\mathcal{U}$ as follows:

$$\Pr\{\mathcal{U}|\mathcal{Z},\mathcal{G}\} = \prod_{j=1}^{N}\sum_{i=1}^{M}\Pr\{g_i|\rho\}\cdot\Pr\{u_j|g_j, z_{i,j}=1\} \tag{5.1}$$

where $|\mathcal{U}| = N$ is the number of total molecules in the experiment (i.e. the number of deduplicated UMIs) and $|\mathcal{G}| = M$ is the number of total genes.

In this study we take the Bayesian approach for gene-expression estimation i.e. instead of seeking the maximum-likelihood estimates we infer the posterior distribution of ρ. This posterior distribution can be defined as:

$$\Pr\{\rho|\mathcal{U},\mathcal{G}\} \propto \sum_{\mathcal{Z}}\Pr\{\mathcal{U}|\mathcal{G},\mathcal{Z}\}\cdot\Pr\{\mathcal{Z}|\rho\}\cdot\Pr\{\rho\} \tag{5.2}$$

where both $\Pr\{\mathcal{U}|\mathcal{G},\mathcal{Z}\}$ and $\Pr\{\mathcal{Z}|\rho\}$ are data-dependent observables and can be estimated as defined in Patro et al. (2017). The novelty of our method comes with setting the prior term $\Pr\{\rho\}$ specifically for `dscRNA-seq` data. We perform intra-sample anchoring using Seurat3 (details in Section 5.2.2) and use the gene expression estimates from the nearest cells (with anchoring score >0.5) to generate cell-specific prior.

5.2.2 Cellular Barcode Anchoring

In Chapter 4 we defined tier 3 estimates as specifically the estimates which have reads assigned to gene ambiguous labels and that don't have any uniquely assignable read evidence in their equivalence class network. To disambiguate the gene assignment, we use information from cells with similar global expression profile within the sample. To find similar cells for sharing the confidence in their gene expression estimates we use Seurat's (Stuart et al., 2019) cellular barcode anchoring scheme. In Seurat, Stuart et al. (2019) proposed an anchoring scheme in which they defined a framework to connect two experiments based on the similarity in their cell's gene expression pattern. They first find the nearest neighbor using l2 distance of both inter and intra datasets to generate four distance matrices. Later, they look for cells which are neighbors to each other in both directions to define anchors connecting two cells with similar looking expression patterns across different single-cell datasets. We use the same anchoring algorithm to connect two similar cells and define the priors in the Bayesian estimation of gene expression.

The anchoring process can sometimes generate multi-mapping in both directions. To compensate for this we take average of all the neighboring cells which have been chosen as the anchor to define the prior. In a typical 4000 single-cell experiment, we run `alevin` and repeat the Seurat3 anchoring approach 30 times, randomly dividing the cells into two equal sets and mapping one set onto the other. We use anchors with score > 0.5 as a potential list of neighboring cells to define the priors. This prior is then used to optimize the quantification estimates of genes with multi-mapping reads in the `alevin` pipeline. As a proof-of-concept, we use the prior with only tier 3 estimates, while keeping uniform prior for both tier 1 and tier 2 estimates.

5.3 Results

One fundamental property of `dscRNA-seq` experiments is that molecules are randomly captured across cells and, although some cells might randomly sample the read sequence from the ambiguous region (or not sample at all), in expectation, similar cells in a group can potentially have different confidence in their gene estimates.

In Chapter 4, we categorize the confidence in the generated gene counts estimates using tiers. Even though we show that the `alevin` generated quantification estimates are accurate, the confidence in tier 3 assigned estimates is low and often a source of mis-estimation. In Figure 5.1 we show the distribution of genes with the fraction of cells that have tier 3 estimates. This motivates our idea that to improve the expression estimates of tier 3 genes, we can utilize the independent sampling of neighboring cells and share information across them to improve gene-level estimates (details in Section 5.2).

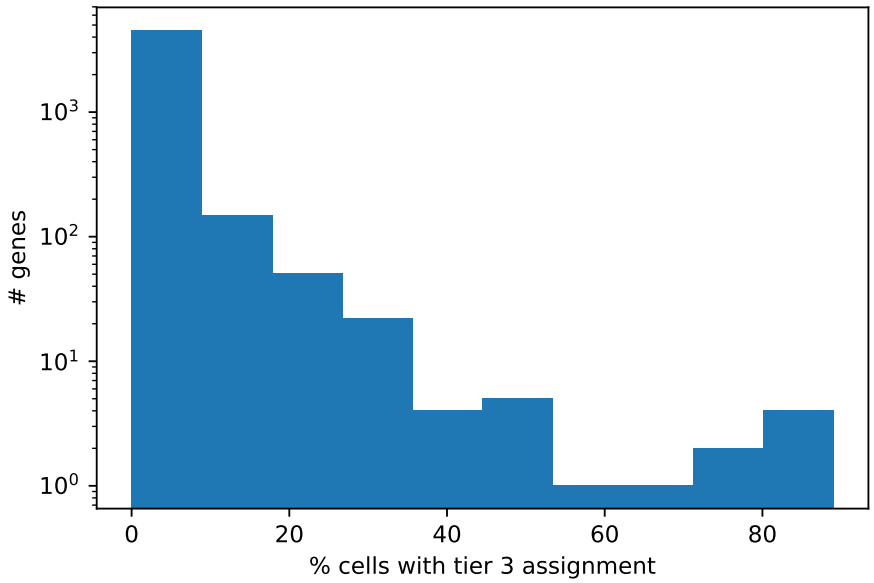


FIGURE 5.1: The distribution of the number of genes with the fraction of cells that have tier 3 assignment.

5.3.1 Variational Bayesian Estimation with Informative Priors Improves `dscRNA-seq` quantification

Over the years, with the advancement in single-cell technologies, various single-cell quantification pipelines have been proposed. However, non-availability of validation datasets and/or criteria for comparing gene-expression estimates generated by the various pipelines has been lacking.

5.3.2 Comparing quantification methods on simulated data

To compare gene estimates, we had previously proposed an empirical `dscRNA-seq` data simulation tool, Minnow (Sarkar, Srivastava, and Patro, 2019). Minnow models various features involved in the generation of the `dscRNA-seq` data, like PCR amplification and sequencing errors to generate fastq file with the reads and the expected true cell by gene counts. We used minnow to simulate a `dscRNA-seq` experiment with 4340 human PBMC cells with $\sim$ 20 Million UMIs and compared the quantification estimates generated by various methods. We found that, in the simulated data, more than 66% of the false discoveries made by `alevin` are in the tier 3 estimates and due to the inherent problem of sparsity in `dscRNA-seq` data, we expect a similar pattern of tier 3 genes in the real data as well.

We validated our prior enhanced VBEM approach on simulated data using the minnow generated data (details in Section 5.3.2). We first used EM based `alevin` on the simulated data to generate quantification estimates on 4340 cells. We used Seurat's anchoring scheme to self anchor the estimates and generate the prior. Using the learned

prior we re-quantified the simulated dataset with our VBEM based approach to generate the new estimates. In Figure 5.2, we compare the cell-wise correlation of both EM and VBEM based approaches to the minnow generated ground truth and observe a higher correlation with VBEM generated gene expression estimates.

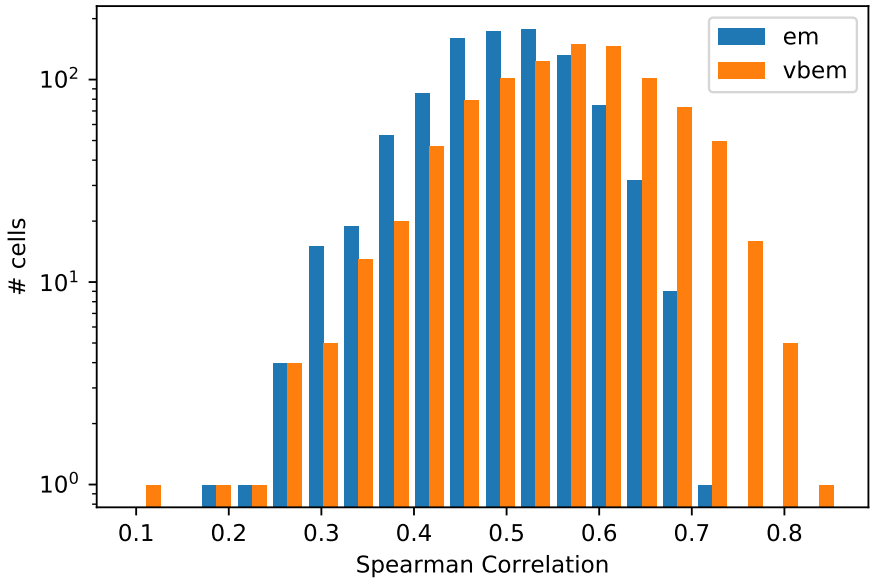


FIGURE 5.2: Comparison of the cell-wise spearman correlation of EM based `alevin` with VBEM based prior enhanced `alevin` on simulated experiment

5.3.3 Knock Out Experiment on Real Data

We validated that the Variational Bayesian estimation of `alevin` is more accurate on real data using the following knock out (KO) experiment. `Alevin`'s pipeline for `dscRNA-seq` quantification has multiple phases. After the initial phases of whitelisting and read-mapping, `alevin` generates an interim file format, called bfh, which contains the transcript equivalence classes along with their Cellular Barcode and UMI count. A major fraction of tier 1 estimates come from unique transcript equivalence classes. In our KO experiment, we knocked out all unique transcript equivalence classes, with the expectation that the KO bfh will migrate some of the tier 1 and tier 2 estimates to tier 3, as they have may lose the unique information which made them high confidence tiers (tier 1 and tier 2) to start with.

We took the quantification estimates from the pbmc_4k experiment with 4340 cells and randomly divided it into two parts of 2170 cells each, we call 2ka and 2kb experiment. We performed the KO experiment on 2ka cells and generated `alevin` EM estimates using KO bfh. Later, we anchored KO 2ka estimates with full 2kb estimates and learned informative priors for KO 2ka estimates to re-quantify using `alevin` VBEM approach. In Figure 5.3, we show the cell-wise correlation of KO 2ka experiment (as

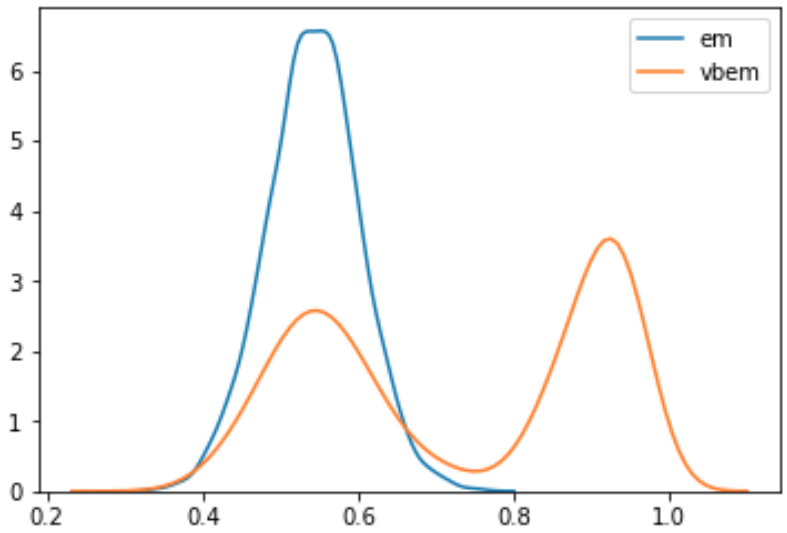


FIGURE 5.3: Comparison of the cell-wise spearman correlation of EM based `alevin` with VBEM based prior enhanced `alevin` on real data KO experiment

em) and prior enhanced version of `alevin` (as VBEM). Since the anchoring scheme can't always find high confidence anchors for all cells, we see a bimodal distribution with VBEM approach which signifies the first mode (overlapping with em) are cells with no informative prior while the second mode (with improved correlation) signifies cells where information can be discovered.

5.4 Conclusion

We proposed a Bayesian framework that extends `alevin`'s maximum likelihood-based quantification procedure. We explore different techniques of generating priors and show that our information-sharing framework consistently improves the tier 3 `dscRNA-seq` quantification estimates and is especially useful for highly ambiguous estimates where there is no intracellular unique information to efficiently quantify. We plan to extend the framework to incorporate priors from new technologies, For example, spatial data can be useful for setting the prior in the proposed `alevin` framework. We think this framework has the potential to open a new direction of enabling multi-modal information sharing to improve the quantification of the data.

Chapter 6

Discussion and Conclusion

In the following sections, we give a brief overview of the efficient computational methods this study presents which improve the quantification of both bulk and dscRNA-seq data along with some of the supplementary work and the future directions.

6.1 Bulk RNA-seq

In Chapter 2 we discuss the usefulness of our novel approach, quasi-mapping, for mapping RNA-seq reads. We show our efficient implementation of quasi-mapping, in the form of `RapMap` is fast, accurate and addresses an important problem of read-mapping. In Chapter 3 we discuss the extension of our quasi-mapping approach into SA, we discuss the pitfalls of using only the simulated dataset for validating computational methods and how including genomic alignment as decoys for the transcriptomic alignment improves the performance of transcript abundance estimation. In addition to that, we are reaching out to find other biological applications where SA can be useful. In the following section, we discuss some of the other applications, which we have worked on.

6.1.1 Supplementary Study

RapClust (Srivastava et al., 2016a)

De novo transcriptome assembly of non-model organisms is the first major step for many RNA-seq analysis tasks. Current methods for de novo assembly often report a large number of contiguous sequences (contigs), which may be fractured and incomplete sequences instead of full-length transcripts. Dealing with a large number of such contigs can slow and complicate downstream analysis. In this extension study, we present a method for clustering contigs from de novo transcriptome assemblies based upon the relationships exposed by multi-mapping sequencing fragments. Specifically, we cast the problem of clustering contigs as one of clustering a sparse graph that is induced by equivalence classes of fragments that map to subsets of the transcriptome. Leveraging recent developments in efficient read mapping and transcript quantification,

we have developed RapClust, a tool implementing this approach that is capable of accurately clustering most large de novo transcriptomes in a matter of minutes, while simultaneously providing accurate estimates of expression for the resulting clusters. We compare RapClust against several tools commonly used for de novo transcriptome clustering. Using de novo assemblies of organisms for which reference genomes are available, we assess the accuracy of these different methods in terms of the quality of the resulting clusterings, and the concordance of differential expression tests with those based on ground truth clusters. We find that RapClust produces clusters of comparable or better quality than existing state-of-the-art approaches, and does so substantially faster. RapClust also confers a large benefit in terms of space usage, as it produces only succinct intermediate files - usually on the order of a few megabytes - even when processing hundreds of millions of reads.

6.2 Single cell RNA-seq

In chapter 4, we present a new end-to-end pipeline for performing gene-level quantification from `dscRNA-seq`. We show that, compared to other methods, `alevin` achieves higher accuracy, in part because of considering a substantially larger number of reads. Further, `alevin` is considerably faster and uses less memory than these other approaches. The optimizations used in `alevin` makes it possible to efficiently process `dscRNA-seq` datasets in a reduced computational burden for processing and re-processing of `dscRNA-seq` data. In chapter 5, we present the extension of `alevin` into using a bayesian approach for solving the expectation-maximization objective. We show the new approach is specifically useful for tier3 (high ambiguous) estimates and significantly improves the cell-wise gene expression estimates by sharing the information across cells. In addition to that, we are reaching out to find other single-cell platforms where `alevin` can be useful. In the following section, we discuss some of the applications, which we have worked on, or are currently working.

6.2.1 Supplementary Study

Swish (Zhu et al., 2019)

A primary challenge in the analysis of RNA-seq data is to identify differentially expressed genes or transcripts while controlling for technical biases present in the observations. Ideally, a statistical testing procedure should incorporate information about the inherent uncertainty of the abundance estimates, whether at the gene or transcript level, that arise from quantification of abundance. Most popular methods for RNA-seq differential expression analysis fit a parametric model to the counts or scaled counts for each gene or transcript, and a subset of methods can incorporate information about the uncertainty of the counts. Previous work has shown that nonparametric models for RNA-seq differential expression may in some cases have better control of the false discovery rate, and adapt well to new data types without requiring reformulation

of a parametric model. Existing nonparametric models do not take into account the inferential uncertainty of the observations, leading to an inflated false discovery rate, in particular at the transcript level. Here we propose a nonparametric model for differential expression analysis using inferential replicate counts, extending the existing SAMseq method to account for inferential uncertainty, batch effects, and sample pairing. We compare our method, "SAMseq With Inferential Samples Helps", or Swish, with popular differential expression analysis methods. Swish has improved control of the false discovery rate, in particular for transcripts with high inferential uncertainty. We apply Swish to a single-cell RNA-seq dataset, assessing sensitivity to recover DE genes between sub-populations of cells, and compare its performance to the Wilcoxon rank sum test.

Minnow (Sarkar, Srivastava, and Patro, 2019)

With the advancements of high-throughput single-cell RNA-sequencing protocols, there has been a rapid increase in the tools available to perform an array of analyses on the gene expression data that results from such studies. For example, 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. Here, we present minnow, a comprehensive 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, we 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.

Appendix A

Supplementary Material for chapter 2

A.1 Parameters for mapping and alignment tools

When `Bowtie 2` was run to produce alignment results, it was run with default parameters with the exception of `-k 200` and `-no-discordant`. When timing `Bowtie 2` the the number of threads (`-p`) was set in accordance with what is mentioned in the relevant text, and the output was piped to `/dev/null`. When `Bowtie 2` was used to produce alignment results for quantification with `RSEM`, `RSEM`'s `Bowtie 2` wrapper (with its default parameters) was used to generate alignments.

When producing alignment results, `STAR` was run with the following parameters: `-outFilterMultimapNmax 200 -outFilterMismatchNmax 99999 -outFilterMismatchNoverLmax 0.2 -alignIntronMin 1000 -alignIntronMax 0 -limitOutSAMoneReadBytes 1000000 -outSAMmode SAMUnosrted`. Additionally, when timing `STAR`, it was run with the number of threads (`-runThreadN`) specified in the relevant text and with the `-outSAMMode None` flag.

To obtain the "pseudo-alignments" produces by `Kallisto`, it was run with the `-pseudobam` flag.

When producing mapping results, `RapMap` was run with the option `-m 200` to limit multi-mapping reads to 200 locations. Additionally, when timing `RapMap`, it was run with the number of threads (`-t`) specified in the relevant text and with the `-n` flag to suppress output.

A.2 Flux Simulator parameters

The Flux simulator dataset was generated using the following parameters:

```
REF_FILE_NAME    Human_Genome
GEN_DIR     protein_coding.gtf

NB_MOLECULES     5000000
```

```
TSS_MEAN    50
POLYA_SCALE NaN
POLYA_SHAPE NaN

FRAG_SUBSTRATE  RNA
FRAG_METHOD UR
FRAG_UR_ETA     350

RTRANSCRIPTION  YES
RT_MOTIF default

GC_MEAN NaN
PCR_PROBABILITY 0.05
PCR_DISTRIBUTION default

FILTERING YES

READ_NUMBER 150000000
READ_LENGTH 76
PAIRED_END  YES
ERR_FILE    76
FASTA       YES
```

The following parameters were used to produce noise reads:

```
PAIRED_END YES
REF_FILE_NAME noisy.gtf

READ_LENGTH 76
PRO_FILE_NAME flux_simulator_noise_expression.pro
ERR_FILE 76
GEN_DIR Human_Genome/
SEQ_FILE_NAME noise_reads.bed
PCR_DISTRIBUTION none
POLYA_SCALE NaN
FASTA YES

NB_MOLECULES 2000000
READ_NUMBER 34382441
UNIQUE_IDS YES
POLYA_SHAPE NaN
```

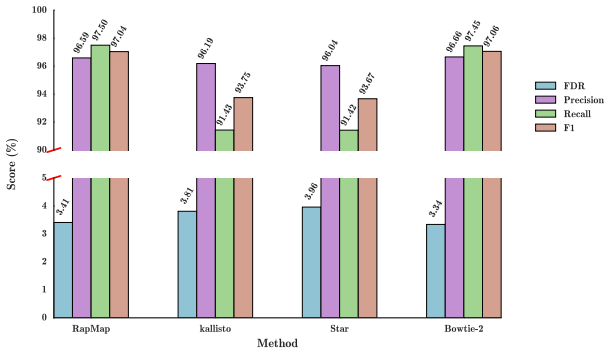


FIGURE A.1: Precision, recall and F1-score (top) and FDR (bottom) on the simulated dataset with noise, for the 4 different tools we consider.

A.3 Mapping accuracy in the presence of noisy reads

We tested the effect of including background (i.e. noise) reads on the accuracy of the different mapping and alignment tools. In this experiment, we sampled 9 million reads from the 48 million read simulated data set used in Section 2.3.1. We then incorporated an additional 1 million "noise" reads from a simulated dataset generated with the Flux Simulator using a custom annotation. This noise annotation was created by constructing a single interval for each transcript, which contained the entire genomic range from the initial until the terminal exons (i.e. it contained all intervening intronic regions). Thus, for each annotated transcript, the noise annotation contains a nascent, un-spliced version of this transcript. This model of noise was motivated from the observation of (Gilbert et al., 2004), that some RNA-seq data (e.g. human brain tissue) contains reads potentially derived from nascent, un-spliced variants of expressed transcripts.

As shown in Supplementary Figure 1 we observe that, in the presence of noise, the precision for all the tools decreases slightly compared to the "clean", 48 million read dataset described in Section 2.3.1. This is because some small fraction of noisy reads are assigned as false positives, as they map to the mature version of their corresponding transcript of origin that appears in the reference. Overall, however, the results follow a very similar trend both with and without noisy reads. Specifically, `RapMap` (quasi-mapping) performs almost identically to `Bowtie 2`, while `Kallisto` and `STAR` yield very similar results — somewhat under-performing `RapMap` and `Bowtie 2`. This clearly demonstrates that, in the presence of noisy reads, all of the tools degrade gracefully and still perform reasonably well, with no discernible difference between mapping and alignment-based tools.

A.4 Quantification results using TPM

In addition to computing the error metrics based on the estimated versus true number of reads originating from each transcript (as provided in Table 2.2), we also evaluated

the same metrics based instead on the TPM of each transcript. That is, all of the metrics defined in Section 2.4.3 and appendix A.5 remain the same, except that x_i now denotes the true TPM value for transcript i and y_i denotes the estimated TPM of transcript i. We note that the Flux Simulator provides neither effective lengths nor TPM estimates directly. To obtain the ground truth TPM values for the Flux Simulator dataset, we first computed the effective length of each transcript (by convolving the characteristic function over the transcript with the *true* fragment length distribution), and then computed the TPM value for each transcript using Equation (2.3). The results are generally similar to what was observed at the read level, except that TIGAR 2 seems to perform considerably worse under a number of metrics on the RSEM-sim dataset when considering the TPM measure of abundance.

TABLE A.1: Performance evaluation of different tools along with quasi enabled sailfish (q-Sailfish) with other tools on synthetic data generated by Flux simulator.

	`Kallisto`	RSEM	q-Sailfish	Tigar 2
Proportionality corr.	0.79	0.80	0.80	0.80
Spearman corr.	0.69	0.73	0.71	0.60
TPEF	0.87	0.88	0.84	0.94
TPME	0.07	0.13	0.12	-0.40
MARD	0.35	0.27	0.31	0.35
wMARD	0.67	1.22	0.69	1.76

TABLE A.2: Performance evaluation of different tools along with quasi enabled sailfish (q-Sailfish) with other tools on synthetic data generated by RSEM simulator.

	`Kallisto`	RSEM	q-Sailfish	Tigar 2
Proportionality corr.	0.94	0.96	0.94	0.93
Spearman corr.	0.91	0.93	0.91	0.89
TPEF	0.51	0.47	0.50	0.95
TPME	0.00	0.00	0.00	0.21
MARD	0.28	0.25	0.28	0.48
wMARD	-0.74	-0.73	-0.74	0.12

A.5 Error Metrics

We define the error metrics reported in Section 2.4.3 below, letting x_i denote the true number of reads originating from transcript i and y_i denote the estimated number of reads.

The relative error for transcript i (RE$_i$) is given by $\mathrm{RE_i} = \frac{x_i - y_i}{x_i}$ and the error indicator for transcript i (EI$_i$) is given by

$$\mathrm{EI_i} = \begin{cases} 1 & \text{if } |RE_i| > 0.1 \\ 0 & \text{otherwise} \end{cases}, \tag{A.1}$$

and it is equal to 1 if the estimated count for this truly expressed transcript (it is undefined, as is RE$_i$, when $x_i = 0$) differs from the true count by more than 10%. Given RE$_i$ and EI$_i$, the aggregate true positive error fraction (TPEF) is defined as TPEF $= \frac{1}{|X^+|}\sum_{i \in X^+} EI_i$. Here, X^+ is the set of "truly expressed" transcripts (those having at least 1 read originating from them in the ground truth). Similarly, the true positive median error is define as TPME $=$ median $(\{\mathrm{RE}_i\}_{i \in X^+})$. Finally, the absolute relative difference for transcript i (ARD$_i$) is defined as

$$\mathrm{ARD}_i = \begin{cases} 0 & \text{if } x_i + y_i = 0 \\ \frac{|x_i - y_i|}{0.5(x_i + y_i)} & \text{otherwise} \end{cases}. \tag{A.2}$$

Consequently, the mean absolute relative difference (MARD) is defined as MARD $= \frac{1}{M}\sum_i \mathrm{ARD}_i$, and the weighted mean absolute relative difference (wMARD) is defined as

$$\mathrm{wMARD} = \sum_{i \in \mathrm{ARD}^+} \frac{\log\left(\max\left(x_i, y_i\right)\right) \mathrm{ARD}_i}{M}, \tag{A.3}$$

where, ARD$^+ = \{i | ARD_i > 0\}$, and M is the total number of transcripts.

Appendix B

Supplementary Material for chapter 3

B.1 Data and reference

The GENCODE v29 Human reference from https://www.gencodegenes.org/human/release_29.html was used for all experiments involving (simulated or experimental) human reads. The mouse reference genome was obtained from ensembl and the GTF was also obtained from ensembl. The VCF files for the SNPs and indels were obtained from sanger and sanger respectively. The list of 109 SRR, scripts to simulate synthetic reads, and the fasta and true abundance files for 10 replicates of simulated data (gencode for human and PWK for mouse) can be found at https://doi.org/10.5281/zenodo.3523437.

B.2 Decoy sequences

Alignment against the genome and transcriptome both have their advantages and disadvantages, as discussed earlier. To avoid aligning genomic reads against the transcriptome, without the need to index the complete genome, requires finding regions with high sequence similarity between them. To obtain similar sequences within a reference, we mapped the spliced transcript sequences against a version of the genome where all exon segments were hard-masked (i.e. replaced with `N`). We performed this mapping using MashMap (Jain et al., 2018), with a segment size 500 and minimum percent identity of 80%. The sequence similar regions were merged (per-chromosome) using BedTools (Quinlan and Hall, 2010) and concatenated, giving a decoy sequence for each chromosome. These decoys were then included during the `Salmon` indexing phase, as described below. A script to obtain these decoy sequences for any reference, given the genome, transcriptome, and annotation is available at: https://github.com.

B.3 Tools

We used `Salmon` v0.15.0 for quasi-mapping and `Salmon` v1.0 for SA and SAF, Bowtie2 version 2.3.4.3, STAR version 2.6.1b, tximport version 1.12.3, DESeq2 version 1.24.0,

`Kallisto` version 0.45.1, edgeR version 3.24.3, limma version 3.38.3, RSEM version 1.2.28, Trim Galore version 0.5.0, bedtools v2.28.0, sleuth version 0.30.0 and MashMap v2.0. All simulated datasets were generated using Polyester version 1.18.0.

For quality trimming the reads we used the following command:

1. **trim_galore** `-q 20 -phred33 -length 20 -paired <fastq file>`

For indexing, we use the following extra command line arguments, along with the regular indexing and threads parameters:

1. **STAR** `-genomeFastaFiles <fasta file> -sjdbGTFfile <gtf file> -sjdbOverhang 100`
2. **Bowtie2** `default`
3. **salmon** `-k 23 -keepDuplicates`
4. **kallisto** `-k 23`

For quantification, we use the following extra command line, along with regular index and threads, with each tools we compare against:

1. **SA and SAF** `-mimicBT2 -useEM`
2. **quasi** `-rangeFactorizationBins 4 -discardOrphansQuasi -useEM -noSA`
3. **Bowtie2** `-sensitive -k 200 -X 1000 -no-discordant -no-mixed -gbar 1`
4. **Bowtie2_strict** `-sensitive -dpad 0 -gbar 99999999 -mp 1,1 -np 1 -score-min L,0,-0.1 -no-mixed -no-discordant -k 200 -I 1 -X 1000`
5. **Bowtie2_RSEM** `-sensitive -dpad 0 -gbar 99999999 -mp 1,1 -np 1 -score-min L,0,-0.1 -no-mixed -no-discordant -k 200 -I 1 -X 1000`
6. **STAR** `-outFilterType BySJout -alignSJoverhangMin 8 -outFilterMultimapNmax 20 -alignSJDBoverhangMin 1 -outFilterMismatchNmax 999 -outFilterMismatchNoverReadLmax 0.04 -alignIntronMin 20 -alignIntronMax 1000000 -alignMatesGapMax 1000000 -readFilesCommand zcat -outSAMtype BAM Unsorted -quantMode TranscriptomeSAM -outSAMattributes NH HI AS NM MD -quantTranscriptomeBan Singleend`
7. **STAR_strict** `-outFilterType BySJout -alignSJoverhangMin 8 -outFilterMultimapNmax 20 -alignSJDBoverhangMin 1 -outFilterMismatchNmax 999 -outFilterMismatchNoverReadLmax`

`0.04 –alignIntronMin 20 –alignIntronMax 1000000 –alignMatesGapMax 1000000 –readFilesCommand zcat –outSAMtype BAM Unsorted –quantMode TranscriptomeSAM –outSAMattributes NH HI AS NM MD –quantTranscriptomeBan IndelSoftclipSingleend`

8. **STAR_RSEM** `–outFilterType BySJout –alignSJoverhangMin 8 –outFilterMultimapNmax 20 –alignSJDBoverhangMin 1 –outFilterMismatchNmax 999 –outFilterMismatchNoverReadLmax 0.04 –alignIntronMin 20 –alignIntronMax 1000000 –alignMatesGapMax 1000000 –readFilesCommand zcat –outSAMtype BAM Unsorted –quantMode TranscriptomeSAM –outSAMattributes NH HI AS NM MD –quantTranscriptomeBan IndelSoftclipSingleend`
9. **RSEM** `default`
10. **`Kallisto`** `default` or `–rf–stranded` as appropriate

	Genome indexed	Alignment scoring	Indels allowed	Quantification method
Bowtie2	✗	✓	✓	`Salmon`
Bowtie2_strict	✗	✓	✗	`Salmon`
Bowtie2_RSEM	✗	✓	✗	RSEM
STAR	✓	✓	✓	`Salmon`
STAR_strict	✓	✓	✗	`Salmon`
STAR_RSEM	✓	✓	✗	RSEM
quasi	✗	✗	✓	`Salmon`
SA	✗*	✓**	✓	`Salmon`
SAF	✓	✓**	✓	`Salmon`

TABLE B.1: Various factors altered under each pipeline. *Here, under SA, only regions of the genome that are sequence similar to the transcriptome are indexed, but not the whole genome. Refer to Appendix B.2 for further details on how the sequences are obtained. **While SA and SAF produce alignment scores, they do not perform backtracing or reconstruct the edit operations that were used to obtain the optimal alignment score.

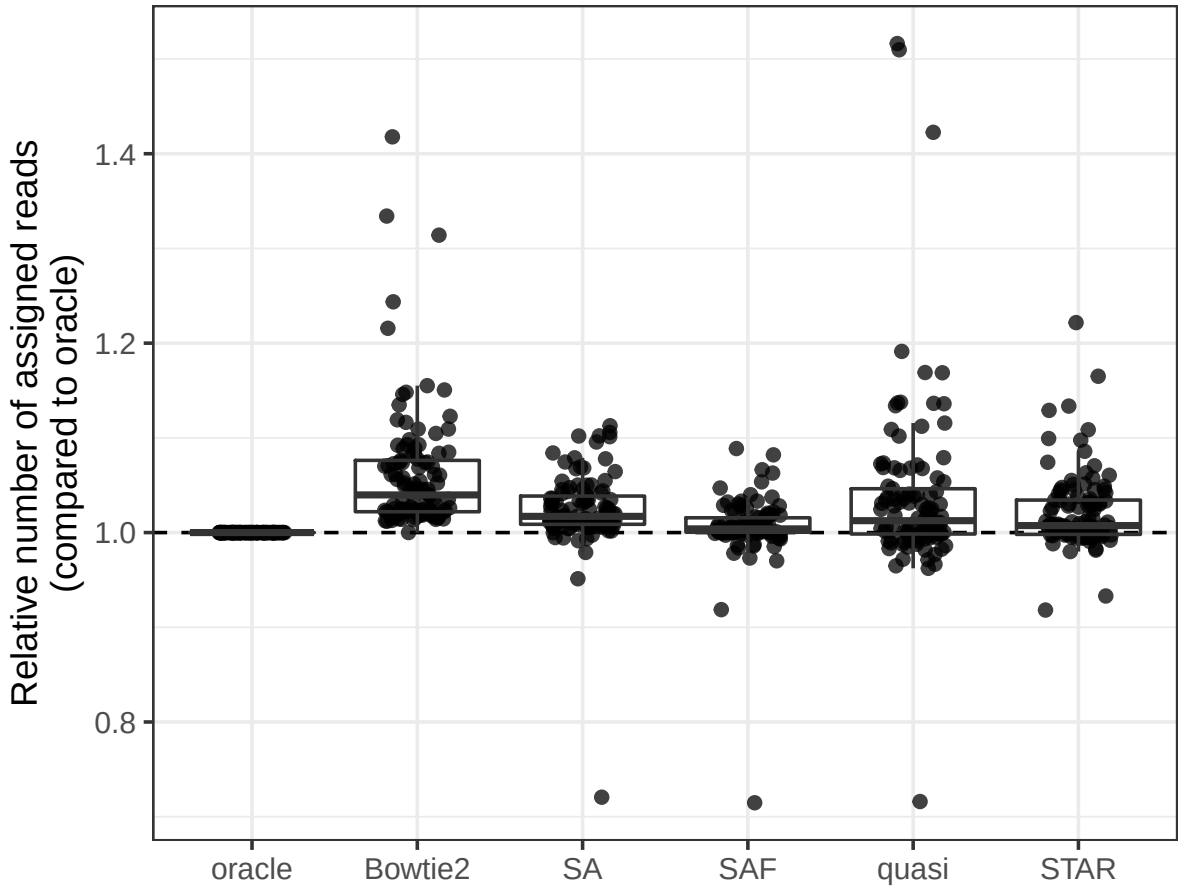


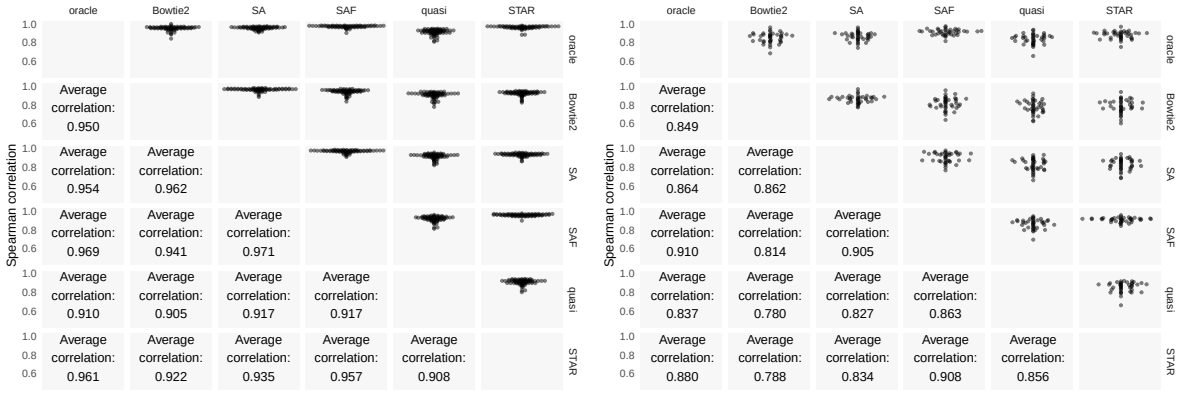
FIGURE B.1: Mapping rates of different methods, relative to oracle, for 109 experimental samples.

	Bowtie2	SAF	SA	quasi	STAR
Oracle	0.858/0.054	0.920/0.029	0.873/0.046	0.843/0.053	0.889/0.039
Bowtie2	1.000/0.000	0.817/0.063	0.863/0.039	0.783/0.068	0.791/0.069
SAF	–	1.000/0.000	0.907/0.052	0.865/0.050	0.909/0.023
SA	–	–	1.000/0.000	0.829/0.063	0.837/0.059
quasi	–	–	–	1.000/0.000	0.858/0.052
STAR	–	–	–	–	1.000/0.000

TABLE B.2: Mean/standard deviation of Spearman correlation between all methods on 40 single-cell experimental datasets after removing short transcripts with length < 300.

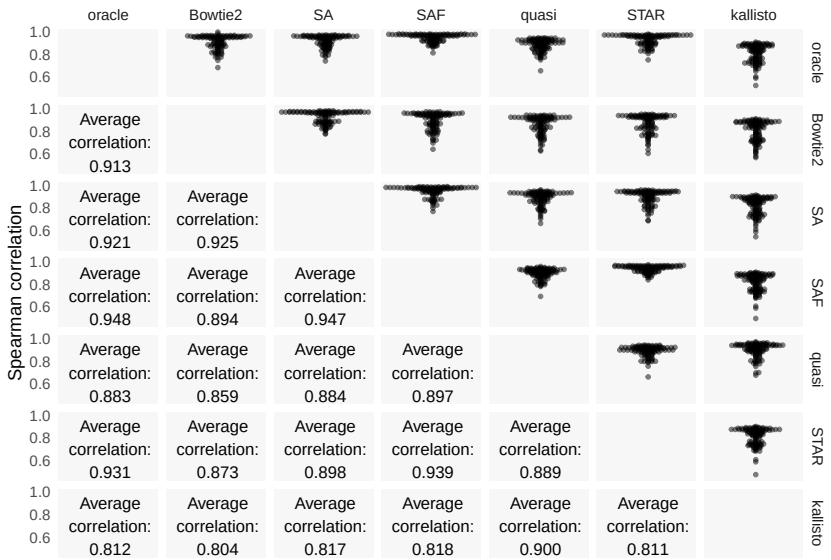
	Bowtie2	SAF	SA	quasi	STAR
Oracle	0.949/0.023	0.971/0.008	0.954/0.014	0.907/0.031	0.962/0.012
Bowtie2	1.000/0.000	0.940/0.026	0.961/0.017	0.901/0.034	0.920/0.025
SAF	–	1.000/0.000	0.972/0.015	0.913/0.031	0.955/0.011
SA	–	–	1.000/0.000	0.914/0.029	0.932/0.017
quasi	–	–	–	1.000/0.000	0.903/0.029
STAR	–	–	–	–	1.000/0.000

TABLE B.3: Mean/standard deviation of Spearman correlation between all methods on 69 bulk experimental datasets after removing short transcripts with length < 300.

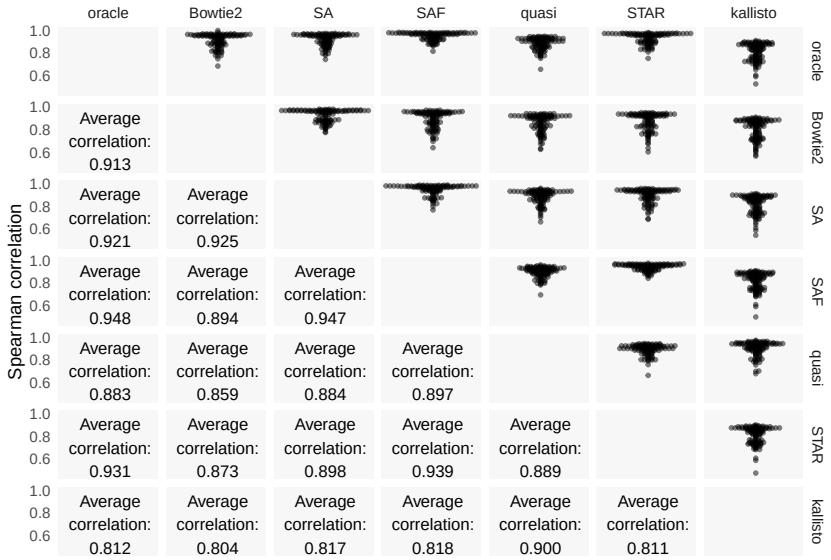


(A) Bulk (B) Single-cell

FIGURE B.2: The upper triangle of the matrix shows swarm plots of pairwise correlations of read counts predicted by the different approaches on the experimental samples. The bottom half shows the average Spearman correlations between methods across the 109 bulk and single-cell samples.

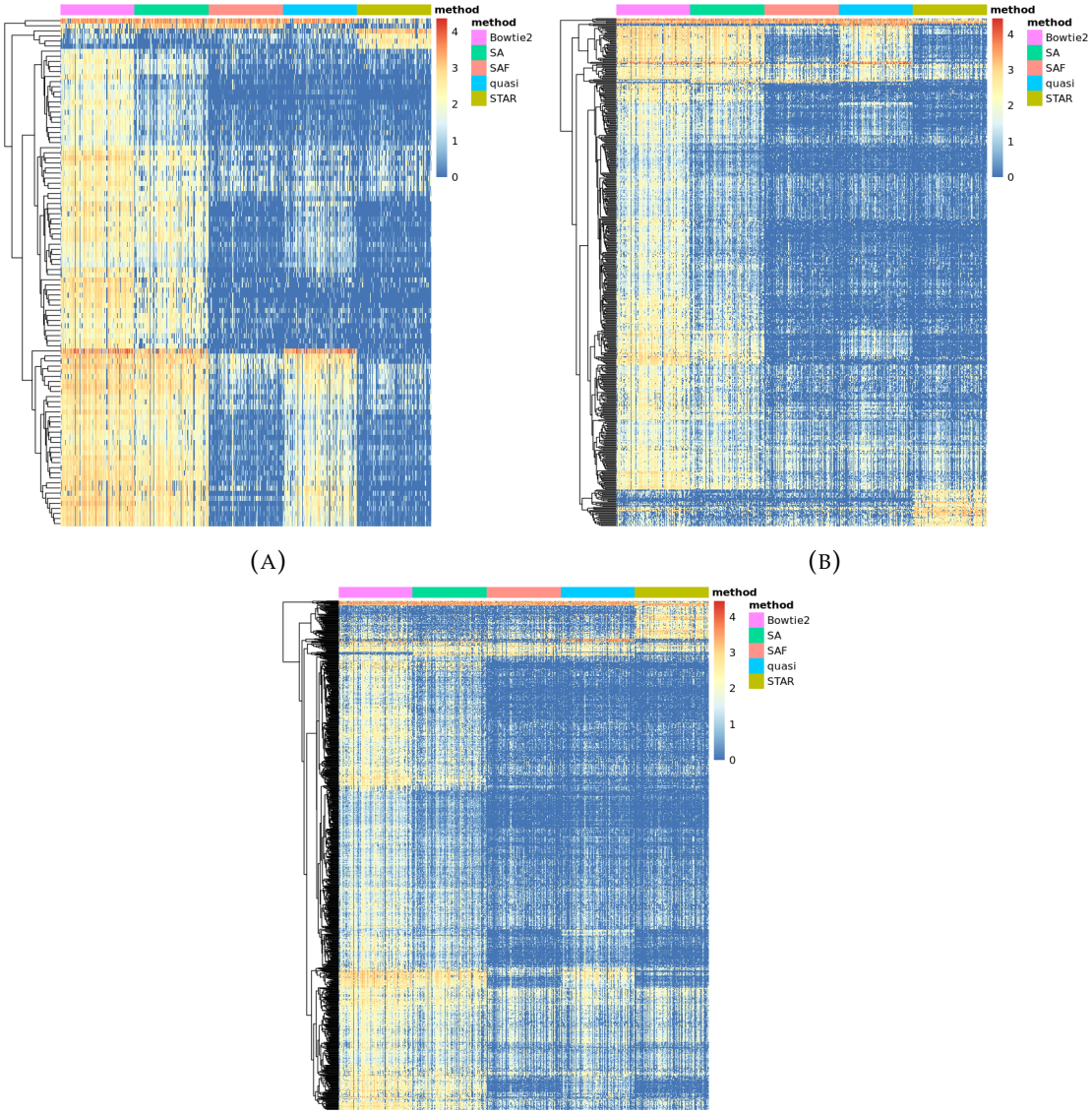


(A)



(B)

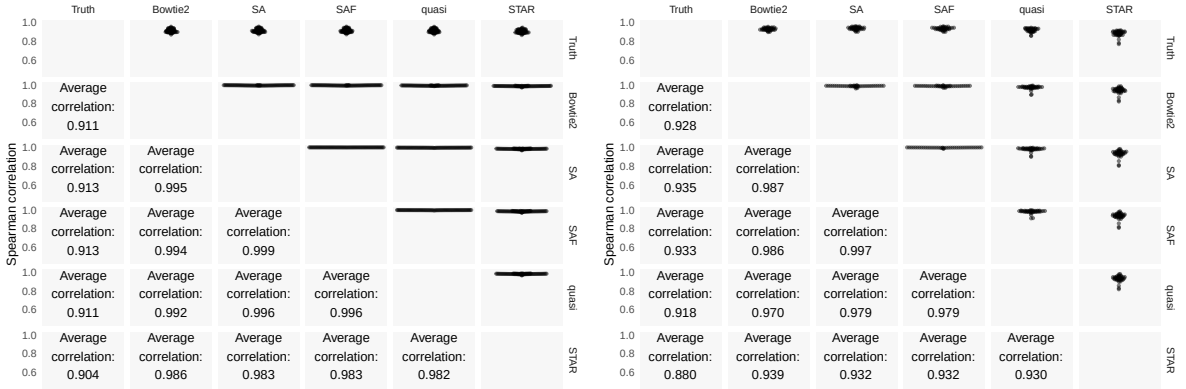
FIGURE B.3: The top half of each matrix shows swarm plots of the pairwise correlations using count (a) and TPM (b) values predicted by the different approaches on the experimental samples. The bottom half shows the average Spearman correlations between methods across the 109 samples. Note that the quantification method for each pipeline is the same, except `Kallisto`, where both the mapping and quantification algorithms are different. Hence, while other methods disallow orphaned reads and dovetailed mappings, the `Kallisto` output will include them, which may explain, in part, the increased divergence from the alignment-based methods.



(A) (B)

(C)

FIGURE B.4: The $\log_2$(CPM) for 109 samples grouped by method for the top 100 (a), 500 (b), and 1000 (c) differential transcripts. Limma-trend was used with `scaledTPM` counts (generating counts from per-sample TPMs by scaling to the library size) via tximport (Soneson, Love, and Robinson, 2016), with a `prior.count` of 3, and using a design of `~sample + method`. An F-statistic was generated by specifying coefficients representing differences among the methods and the top transcripts chosen using the F-test p-value.



(A) Bulk (B) Single-cell

FIGURE B.5: The top half of the matrix shows swarm plots of the pairwise correlations of TPM values predicted by the different approaches with each other and with the ground truth abundances on the simulated samples. The bottom half shows the average Spearman correlations between the different approaches across the 109 samples. The expected effective length of each transcript was computed according to the true fragment length distribution. Given the true fragment counts and expected effective lengths, the TPM is computed as in **rsem**.

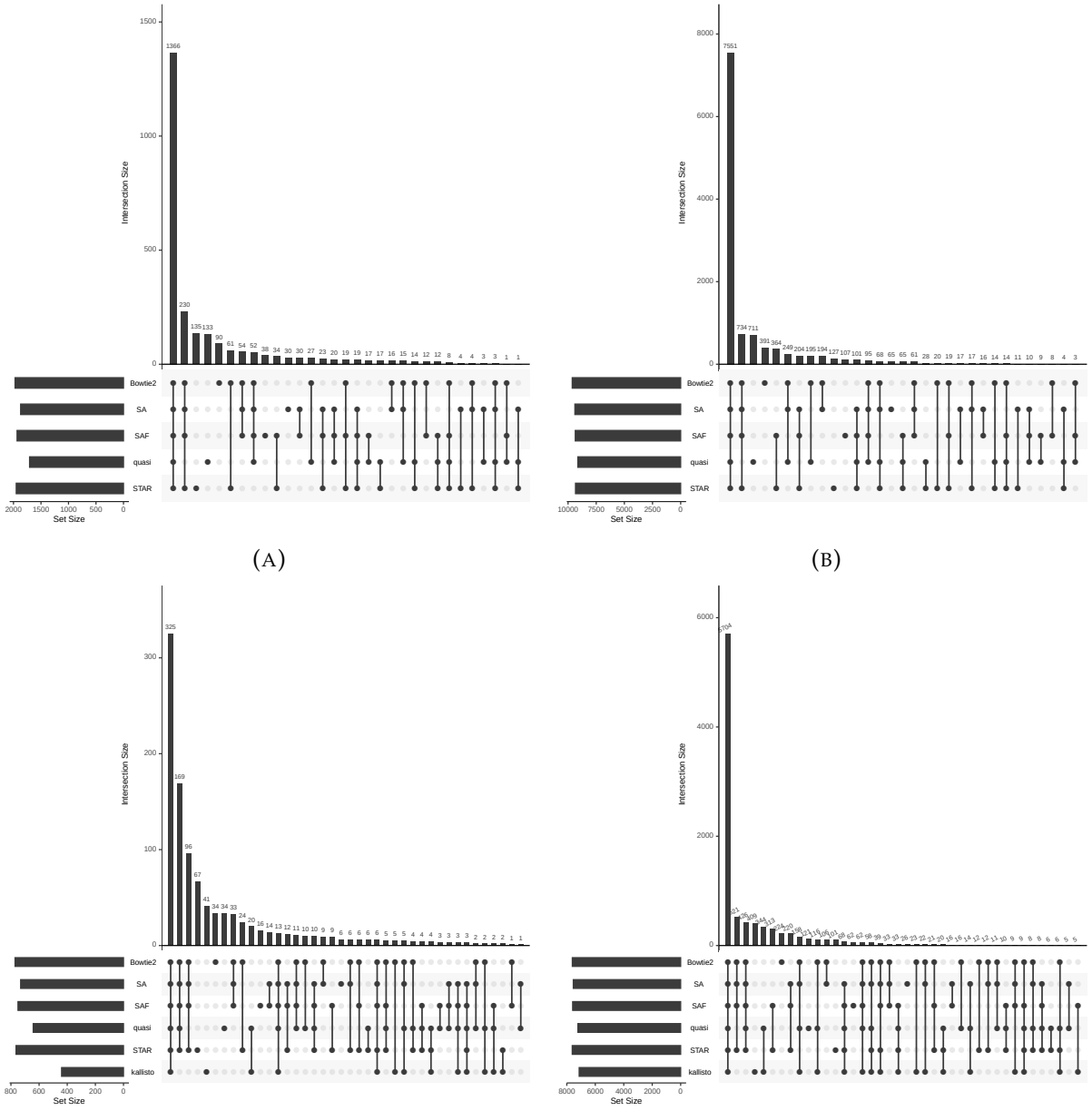
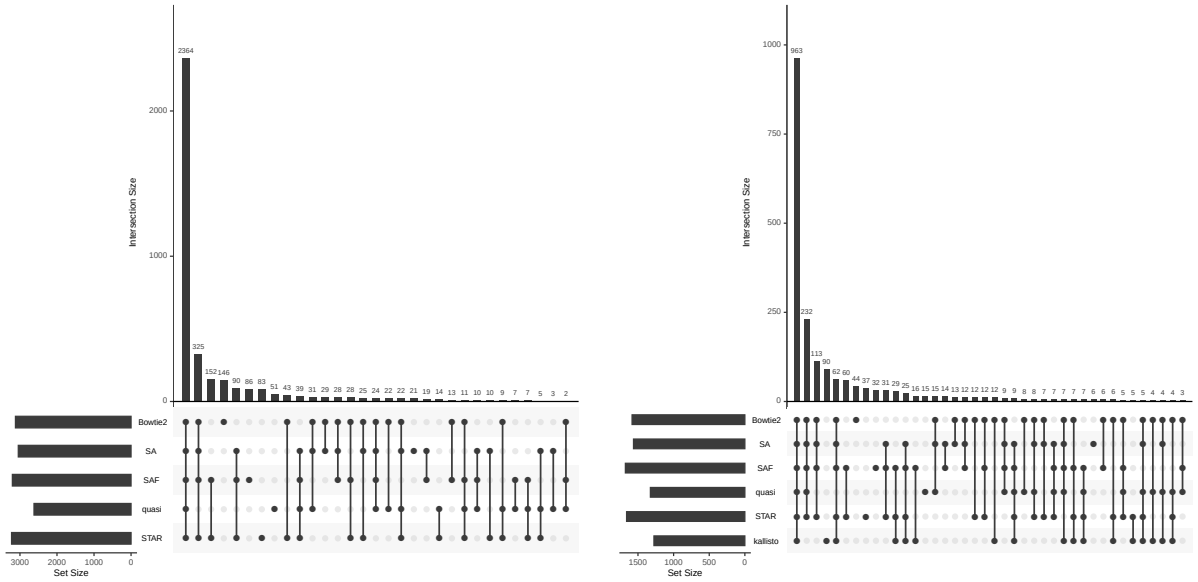


FIGURE B.6: Comparison of sets of differentially expressed genes, and their overlaps, computed using each method. Figures (a) and (b) shows the results for the two datasets when filtered at an FDR of 0.05 and (c) and (d) shows the results at FDR 0.01 after including `Kallisto` as an additional lightweight mapping approach.



(A) (B)

FIGURE B.7: Comparison of sets of differentially expressed transcripts, and their overlaps, computed using each method. Figure (a) shows the results when filtered at an FDR of 0.05 and (b) shows the results at FDR 0.01 after including `Kallisto` as an additional lightweight mapping approach.

Appendix C

Supplementary Material for chapter 4

C.1 Machine Configuration and Pipeline Replicability

10x v2 chemistry benchmarking has been scripted using CGATCore (https://github.com/cgat-developers/cgat-core). The full pipeline and analysis are performed using Stony Brook's seawulf cluster with 164 Intel Xeon E5-2683v3 CPUs.

For all analyses, the genome and gtf versions used for human datasets was GENCODE release 27, GRCh38.p10 and for mouse datasets was GENCODE release M16, GRCm38.p5. All transcriptome files were generated using these with "rsem-prepare-reference".

`Cell-Ranger` (v2.2.0): The following additional flags were used, as recommended by the `Cell-Ranger` guidelines: `-nosecondary -expect-cells NumCells`, where NumCells is 10,000 for PBMC 8k and Neurons 9k, 5,000 for PBMC 4k, 2,000 for Neurons 2k and Neurons 900.

`Alevin` (v0.13.0): Run with default parameters with the chromium protocol and `-keepDuplicates` flags and the `-lISR` to specify strandedness. The mRNA and rRNA lists were obtained from the relevant annotation files and passed as input. Experiments on v1 chemistry can be run using the same flags but with the `-gemcode` protocol flag. `Alevin` also supports 10x v3 chemistry via the command-line flag `-chromiumV3`.

STAR (v2.6.0a): The following flag was used, as recommended by the guidelines of UMI-tools:
`-outFilterMultimapNmax 1`

featureCounts (v1.6.3): This was run to obtain an output BAM file and with stranded input (`-s 1`).

UMI-tools (v0.5.4): The extract command was used to get the CBs/UMIs, when provided with an external CB whitelist, and attach it to the corresponding reads. The following flags were used in the count command to obtain the per cell gene count matrix: `-gene-tag=XT -wide-format-cell-counts`

`dropEst` (v0.8.5): This was run with the default parameters on the 10x BAM files and the predicted cell counts from `Cell-Ranger` were used as input.

Dropseq utils (v2.0.0): All the commands were run as recommended by the authors in the tool's manual.

The bulk datasets were quantified using Bowtie2 and RSEM, run as follows:

Bowtie2 (v2.3.4.3): The following flags were used, as recommended in the guidelines of RSEM: `--sensitive --dpad 0 --gbar 99999999 --mp 1,1 --np 1 --score-min L,0,-0.1 --no-mixed --no-discordant`

RSEM (v1.3.1): Run with default parameters.

C.2 Availability of data and materials

`Alevin` is implemented in `C++14` and is released under the GNU General Public License v3.0. The source code as used in the chapter has been deposited in archived format at `https://doi.org/10.5281/zenodo.2583275` (Srivastava et al., 2019d) and the latest code is available at `https://github.com/COMBINE-lab/salmon` (Srivastava et al., 2019c). The output quantification results of all the tools used in the validation of `alevin`-pipeline have been deposited in archived format at `https://doi.org/10.5281/zenodo.2583228` (Srivastava et al., 2019b).

All the single cell 10x datasets used in the chapter are taken from `https://support.10xgenomics.com/single-cell-gene-expression/datasets` (*10x-Genomics v2 Chemistry Data* 2018) and the DropSeq data is from SRR1853180. The relevant accessions for the bulk RNA-seq datasets used for the validation are listed in Table 4.1.

C.3 Additional Figures

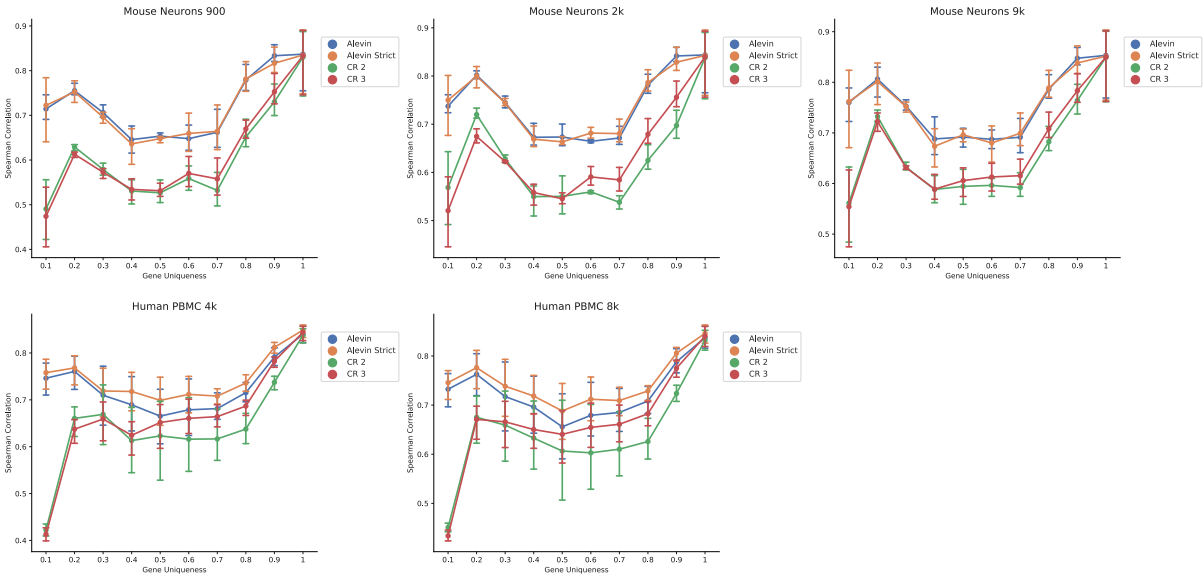


FIGURE C.1: The Spearman correlation between quantification estimates from different runs of `alevin` and `Cell-Ranger`. Note that two different versions `Cell-Ranger` were run with the default parameters and `alevin` strict refers to the same version of `alevin` run with `--minScoreFraction` set to 0.95 and `--consensusSlack` set to 0.99. These parameters in `alevin` make the mapping filter strict and allows fewer spurious mappings.

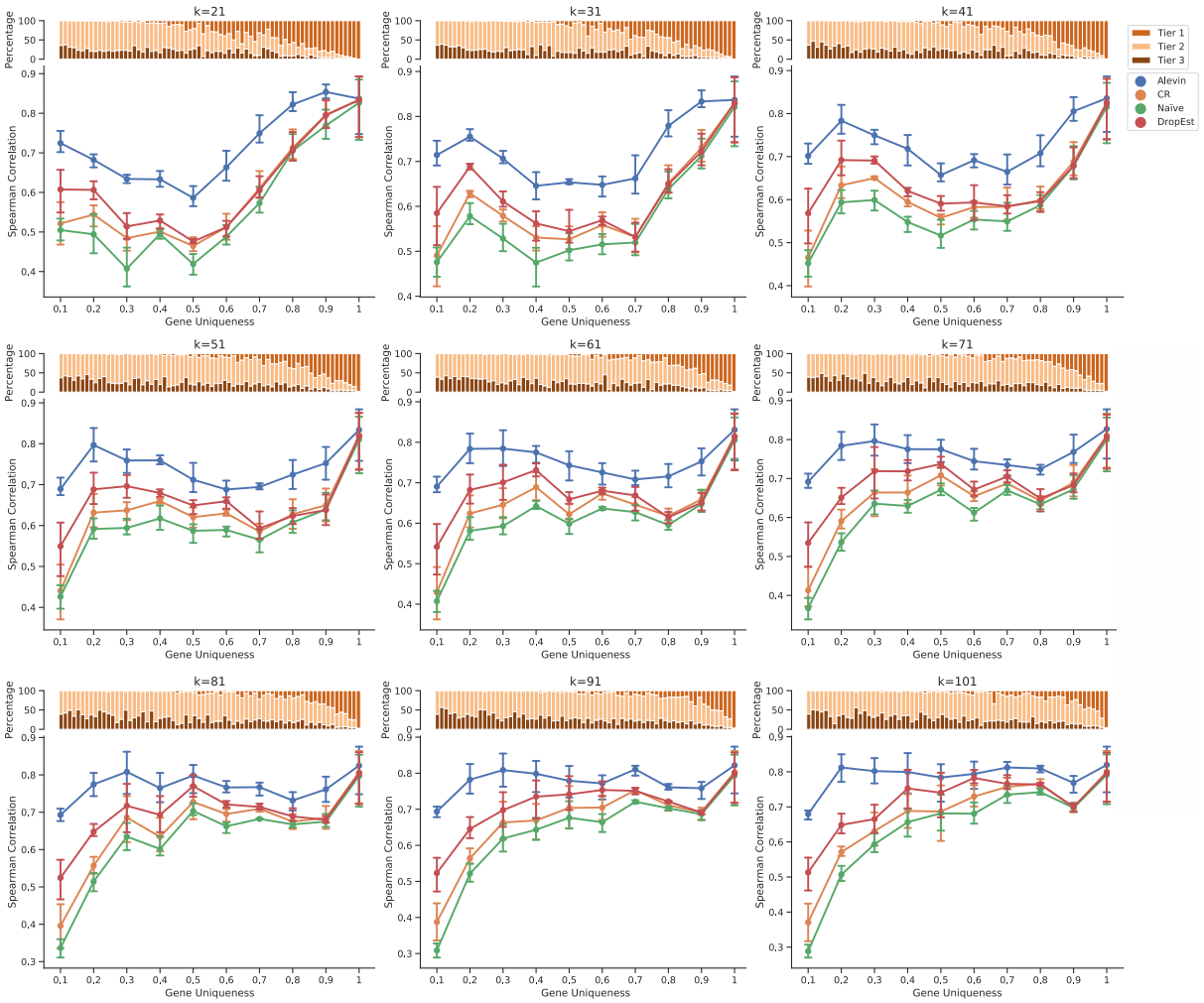


FIGURE C.2: Correlation plots for the mouse neuronal 900 dataset using different values of the k-mer size (k) to calculate gene uniqueness.

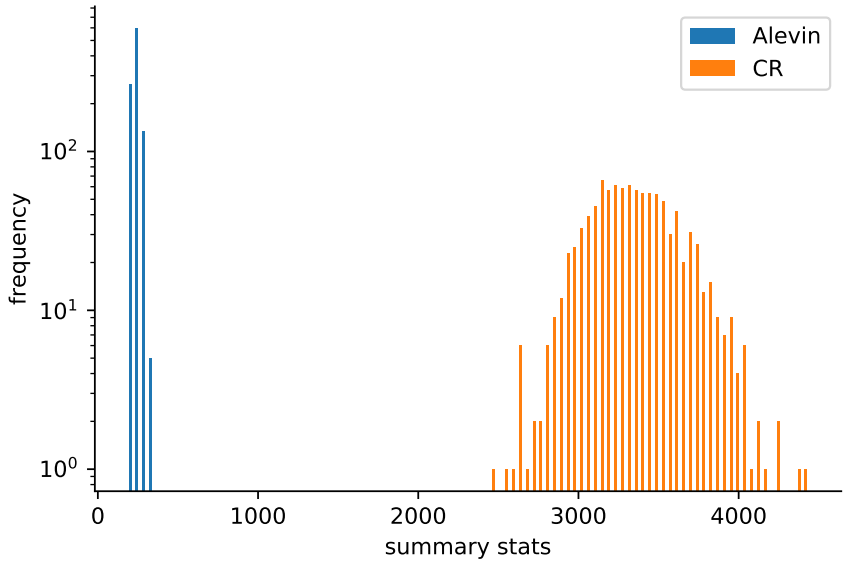


FIGURE C.3: The histogram is the result of taking 1000 samples of 100 cells each from the mouse neuronal 900 dataset, and looking at the sum of absolute differences when quantifying the data under the varying reference genomes (mouse vs. mouse and human combined).

Bibliography

10x-Genomics Single-Cell 3'-V2 Kit (2018). https://teichlab.github.io/scg_lib_structs/data/CG000108_AssayConfiguration_SC3v2.pdf.

10x-Genomics v2 Chemistry Data (2018). https://support.10xgenomics.com/single-cell-gene-expression/datasets.

Alignment vs. Mapping. http://robpatro.com/blog/?p=260. Accessed: 2015-08-15.

Almodaresi, Fatemeh et al. (2018). "A space and time-efficient index for the compacted colored de Bruijn graph". In: *Bioinformatics* 34.13, pp. i169–i177.

Alternative Splicing. https://en.wikipedia.org/wiki/Alternative_splicing.

Altschul, Stephen F et al. (1990). "Basic local alignment search tool". In: *Journal of molecular biology* 215.3, pp. 403–410.

Altschul, Stephen F et al. (1997). "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs". In: *Nucleic acids research* 25.17, pp. 3389–3402.

Bernáth, Attila and Gyula Pap (2011). *Covering minimum cost arborescences*. Tech. rep. TR-2011-13. www.cs.elte.hu/egres. Egerváry Research Group, Budapest.

Bouquet, Jerome et al. (2016). "Longitudinal transcriptome analysis reveals a sustained differential gene expression signature in patients treated for acute Lyme disease". In: *MBio* 7.1, e00100–16.

Bowtie2 user manual. http://bowtie-bio.sourceforge.net/bowtie2/manual.shtml. Accessed: 2019-10-04.

Bray, Nicolas L et al. (2016). "Near-optimal probabilistic RNA-seq quantification". In: *Nature Biotechnology* 34.5, p. 525.

Burrows, Michael and David Wheeler (1994). "A block-sorting lossless data compression algorithm". In: *DIGITAL SRC RESEARCH REPORT*. Citeseer.

Cho, H. et al. (2014). "High-resolution transcriptome analysis with long-read RNA sequencing". In: *PLoS ONE* 9.9, e108095.

Consortium, ENCODE Project et al. (2012). "An integrated encyclopedia of DNA elements in the human genome". In: *Nature* 489.7414, p. 57.

Conway, Jake R, Alexander Lex, and Nils Gehlenborg (2017). "UpSetR: an R package for the visualization of intersecting sets and their properties". In: *Bioinformatics* 33.18, pp. 2938–2940.

Cunningham, Fiona et al. (2015). "Ensembl 2015". In: *Nucleic acids research* 43.D1, pp. D662–D669.

Davidson, Nadia M and Alicia Oshlack (2014). "Corset: enabling differential gene expression analysis for de novo assembled transcriptomes". In: *Genome biology* 15.7, p. 410.

Dobin, Alexander et al. (2013). "STAR: ultrafast universal RNA-seq aligner". In: *Bioinformatics* 29.1, pp. 15–21.

Dvinge, Heidi et al. (2014). "Sample processing obscures cancer-specific alterations in leukemic transcriptomes". In: *Proceedings of the National Academy of Sciences* 111.47, pp. 16802–16807.

Ewing, Brent and Phil Green (1998). "Base-calling of automated sequencer traces using phred. II. Error probabilities". In: *Genome research* 8.3, pp. 186–194.

Ferragina, Paolo and Giovanni Manzini (2000). "Opportunistic data structures with applications". In: *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*. IEEE, pp. 390–398.

– (2001). "An experimental study of an opportunistic index". In: *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, pp. 269–278.

Fratta, Pietro et al. (2018). "Mice with endogenous TDP-43 mutations exhibit gain of splicing function and characteristics of amyotrophic lateral sclerosis". In: *The EMBO Journal* 37.11, e98684.

Frazee, Alyssa C et al. (2015). "Polyester: simulating RNA-seq datasets with differential transcript expression". In: *Bioinformatics* 31.17, pp. 2778–2784.

Fu, Limin et al. (2012). "CD-HIT: accelerated for clustering the next-generation sequencing data". In: *Bioinformatics* 28.23, pp. 3150–3152.

Germain, Pierre-Luc et al. (2016). "RNAontheBENCH: computational and empirical resources for benchmarking RNAseq quantification and differential expression methods". In: *Nucleic Acids Research* 44.11, pp. 5054–5067.

Gilbert, Christopher et al. (2004). "Elongator interactions with nascent mRNA revealed by RNA immunoprecipitation". In: *Molecular cell* 14.4, pp. 457–464.

Glaus, Peter, Antti Honkela, and Magnus Rattray (2012). "Identifying differentially expressed transcripts from RNA-seq data with biological variation". In: *Bioinformatics* 28.13, pp. 1721–1728.

Goldberg, Aaron D, C David Allis, and Emily Bernstein (2007). "Epigenetics: a landscape takes shape". In: *Cell* 128.4, pp. 635–638.

Grabherr, Manfred G et al. (2011). "Full-length transcriptome assembly from RNA-seq data without a reference genome". In: *Nature biotechnology* 29.7, pp. 644–652.

Grant, Gregory R et al. (2011). "Comparative analysis of RNA-Seq alignment algorithms and the RNA-Seq unified mapper (RUM)". In: *Bioinformatics* 27.18, pp. 2518–2528.

Griebel, Thasso et al. (2012). "Modelling and simulating generic RNA-seq experiments with the flux simulator". In: *Nucleic acids research* 40.20, pp. 10073–10083.

Gusfield, Dan (1997). *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. New York, NY, USA: Cambridge University Press. ISBN: 0-521-58519-8.
Hach, Faraz et al. (2010). "mrsFAST: a cache-oblivious algorithm for short-read mapping". In: *Nature methods* 7.8, pp. 576–577.
Han, Xiaoping et al. (2018). "Mapping the mouse cell atlas by Microwell-seq". In: *Cell* 172.5, pp. 1091–1107.
Hashimshony, Tamar et al. (2012). "CEL-Seq: single-cell RNA-Seq by multiplexed linear amplification". In: *Cell reports* 2.3, pp. 666–673.
Hensman, James et al. (2015). "Fast and accurate approximate inference of transcript expression from RNA-seq data". In: *Bioinformatics* 31.24, pp. 3881–3889.
Ilie, Lucian, Gonzalo Navarro, and Liviu Tinta (2010). "The longest common extension problem revisited and applications to approximate string searching". In: *Journal of Discrete Algorithms* 8.4, pp. 418–428.
Islam, Saiful et al. (2012). "Highly multiplexed and strand-specific single-cell RNA 5′ end sequencing". In: *Nature protocols* 7.5, p. 813.
Islam, Saiful et al. (2014). "Quantitative single-cell RNA-seq with unique molecular identifiers". In: *Nature Methods* 11.2, p. 163.
Jain, Chirag et al. (2018). "A fast adaptive algorithm for computing whole-genome homology maps". In: *Bioinformatics* 34.17, pp. i748–i756.
Ju, Chelsea J-T et al. (2017). "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*. ACM, pp. 263–272.
Kanitz, Alexander et al. (2015). "Comparative assessment of methods for the computational inference of transcript isoform abundance from RNA-seq data". In: *Genome biology* 16.1, p. 150.
Kent, W James (2002). "BLAT–the BLAST-like alignment tool". In: *Genome research* 12.4, pp. 656–664.
Kim, Daehwan, Ben Langmead, and Steven L Salzberg (2015). "HISAT: a fast spliced aligner with low memory requirements". In: *Nature methods* 12.4, pp. 357–360.
Kiskinis, Evangelos et al. (2014). "Pathways disrupted in human ALS motor neurons identified through genetic correction of mutant SOD1". In: *Cell stem cell* 14.6, pp. 781–795.
Klein, Allon M et al. (2015). "Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells". In: *Cell* 161.5, pp. 1187–1201.
Köster, Johannes and Sven Rahmann (2012). "Building and Documenting Workflows with Python-Based Snakemake." In: *GCB*, pp. 49–56.
Krueger, Felix (2015). "Trim Galore". In: *A Wrapper Tool Around Cutadapt and FastQC to Consistently Apply Quality and Adapter Trimming to FastQ Files*. URL: http://www.bioinformatics.babraham.ac.uk/projects/trim_galore/.
Langmead, Ben and Steven L Salzberg (2012). "Fast gapped-read alignment with Bowtie 2". In: *Nature Methods* 9.4, p. 357.

Langmead, Ben et al. (2009). "Ultrafast and memory-efficient alignment of short DNA sequences to the human genome". In: *Genome Biology* 10.3, R25.
Lappalainen, Tuuli et al. (2013). "Transcriptome and genome sequencing uncovers functional variation in humans". In: *Nature*.
Law, Charity W et al. (2014). "voom: Precision weights unlock linear model analysis tools for RNA-seq read counts". In: *Genome Biology* 15.2, R29.
Li, Bo and Colin N Dewey (2011). "RSEM: accurate transcript quantification from RNA-seq data with or without a reference genome". In: *BMC bioinformatics* 12.1, p. 323.
Li, Bo et al. (2010). "RNA-seq gene expression estimation with read mapping uncertainty". In: *Bioinformatics* 26.4, pp. 493–500.
Li, Heng (2013). *Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM*.
– (2018). "Minimap2: pairwise alignment for nucleotide sequences". In: *Bioinformatics* 34.18, pp. 3094–3100.
Li, Heng and Richard Durbin (2009). "Fast and accurate short read alignment with Burrows–Wheeler transform". In: *Bioinformatics* 25.14, pp. 1754–1760.
Li, Heng and Nils Homer (2010). "A survey of sequence alignment algorithms for next-generation sequencing". In: *Briefings in bioinformatics* 11.5, pp. 473–483.
Li, Heng, Jue Ruan, and Richard Durbin (2008). "Mapping short DNA sequencing reads and calling variants using mapping quality scores". In: *Genome research* 18.11, pp. 1851–1858.
Li, Ruiqiang et al. (2008). "SOAP: short oligonucleotide alignment program". In: *Bioinformatics* 24.5, pp. 713–714.
Li, Ruiqiang et al. (2009). "SOAP2: an improved ultrafast tool for short read alignment". In: *Bioinformatics* 25.15, pp. 1966–1967.
Liao, Yang, Gordon K Smyth, and Wei Shi (2013a). "featureCounts: an efficient general purpose program for assigning sequence reads to genomic features". In: *Bioinformatics* 30.7, pp. 923–930.
– (2013b). "The Subread aligner: fast, accurate and scalable read mapping by seed-and-vote". In: *Nucleic acids research* 41.10, e108–e108.
Lister, Ryan et al. (2008). "Highly integrated single-base resolution maps of the epigenome in Arabidopsis". In: *Cell* 133.3, pp. 523–536.
Liu, Bo et al. (2016). "deBGA: read alignment with de Bruijn graph-based seed and extension". In: *Bioinformatics* 32.21, pp. 3224–3232.
Love, Michael I, Wolfgang Huber, and Simon Anders (2014). "Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2". In: *Genome Biology* 15.12, p. 550.
Lovell, David et al. (2015). "Proportionality: a valid alternative to correlation for relative data". In: *PLoS computational biology* 11.3, e1004075.
Macosko, Evan Z et al. (2015). "Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets". In: *Cell* 161.5, pp. 1202–1214.

Manber, Udi and Gene Myers (1993). "Suffix arrays: a new method for on-line string searches". In: *siam Journal on Computing* 22.5, pp. 935–948.
Marçais, Guillaume and Carl Kingsford (2011). "A fast, lock-free approach for efficient parallel counting of occurrences of k-mers". In: *Bioinformatics* 27.6, pp. 764–770.
Martin, Marcel (2011). "Cutadapt removes adapter sequences from high-throughput sequencing reads". In: *EMBnet. Journal* 17.1, pp. 10–12.
Mezlini, Aziz M et al. (2013). "iReckon: simultaneous isoform discovery and abundance estimation from RNA-seq data". In: *Genome Research* 23.3, pp. 519–529.
Mortazavi, Ali et al. (2008). "Mapping and quantifying mammalian transcriptomes by RNA-Seq". In: *Nature methods* 5.7, p. 621.
Munger, Steven C et al. (2014). "RNA-Seq alignment to individualized genomes improves transcript abundance estimates in multiparent populations". In: *Genetics* 198.1, pp. 59–73.
Nagalakshmi, Ugrappa et al. (2008). "The transcriptional landscape of the yeast genome defined by RNA sequencing". In: *Science* 320.5881, pp. 1344–1349.
Nakaya, Helder I et al. (2011). "Systems biology of vaccination for seasonal influenza in humans". In: *Nature Immunology* 12.8, p. 786.
Nariai, Naoki et al. (2013). "TIGAR: transcript isoform abundance estimation method with gapped alignment of RNA-seq data by variational Bayesian inference". In: *Bioinformatics* 29, btt381.
Nariai, Naoki et al. (2014). "TIGAR2: sensitive and accurate estimation of transcript isoform expression with longer RNA-seq reads". In: *BMC genomics* 15.Suppl 10, S5.
Nicolae, M. et al. (2011). "Estimation of alternative splicing isoform frequencies from RNA-seq data". In: *Algorithms for Molecular Biology* 6:9.
Nookaew, Intawat et al. (2012). "A comprehensive comparison of RNA-seq-based transcriptome analysis from reads to differential gene expression and cross-comparison with microarrays: a case study in Saccharomyces cerevisiae". In: *Nucleic acids research*, gks804.
Ntranos, Vasilis et al. (2016). "Fast and accurate single-cell RNA-seq analysis by clustering of transcript-compatibility counts". In: *Genome Biology* 17.1, p. 112.
Patro, Rob, Stephen M Mount, and Carl Kingsford (2014). "Sailfish enables alignment-free isoform quantification from RNA-seq reads using lightweight algorithms". In: *Nature Biotechnology* 32.5, p. 462.
Patro, Rob et al. (2017). "Salmon provides fast and bias-aware quantification of transcript expression". In: *Nature Methods* 14.4, p. 417.
Petukhov, V and Guo, J and Baryawno, N and Severe, N and Scadden, DT and Samsonova, MG and Kharchenko, PV (2018). "dropEst: pipeline for accurate estimation of molecular counts in droplet-based single-cell RNA-seq experiments". In: *Genome Biology* 19.1, p. 78.
Pimentel, Harold et al. (2017). "Differential analysis of RNA-seq incorporating quantification uncertainty". In: *Nature Methods* 14.7, p. 687.
Pipeline for initial analysis of droplet-based single-cell RNA-seq data (2018). https://github.com/hms-dbmi/dropEst. Accessed: 2018-10-19.

Poldrack, Russell A et al. (2015). "Long-term neural and physiological phenotyping of a single human". In: *Nature Communications* 6, p. 8885.
Quinlan, Aaron R and Ira M Hall (2010). "BEDTools: a flexible suite of utilities for comparing genomic features". In: *Bioinformatics* 26.6, pp. 841–842.
Reis-Filho, Jorge S et al. (2009). "Next-generation sequencing". In: *Breast Cancer Res* 11.Suppl 3, S12.
Richter, Franziska et al. (2009). "Neurons express hemoglobin α-and β-chains in rat and human brains". In: *Journal of Comparative Neurology* 515.5, pp. 538–547.
Robert, Christelle and Mick Watson (2015). "Errors in RNA-Seq quantification affect genes of relevance to human disease". In: *Genome Biology* 16.1, p. 177.
RSEM manual. https://deweylab.github.io/RSEM/. Accessed: 2019-04-09.
Saha, Ashis and Alexis Battle (2018). "False positives in trans-eQTL and co-expression analyses arising from RNA-sequencing alignment errors [version 1; peer review: 3 approved]". In: *F1000Research* 7:1860.
Saito, Yuhki et al. (2016). "NOVA2-mediated RNA regulation is required for axonal pathfinding during development". In: *Elife* 5, e14371.
Sanger, Fred and Alan R Coulson (1975). "A rapid method for determining sequences in DNA by primed synthesis with DNA polymerase". In: *Journal of molecular biology* 94.3, pp. 441–448.
Sarkar, Hirak, Avi Srivastava, and Rob Patro (2019). "Minnow: a principled framework for rapid simulation of dscRNA-seq data at the read level". In: *Bioinformatics* 35.14, pp. i136–i144.
Sarkar, Hirak et al. (2018). "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. New York, NY, USA: ACM, pp. 27–36.
Schadt, Eric E, Steve Turner, and Andrew Kasarskis (2010). "A window into third-generation sequencing". In: *Human molecular genetics*, ddq416.
Schatz, Michael C, Arthur L Delcher, and Steven L Salzberg (2010). "Assembly of large genomes using second-generation sequencing". In: *Genome research* 20.9, pp. 1165–1173.
Schmitt, Bianca M et al. (2014). "High-resolution mapping of transcriptional dynamics across tissue development reveals a stable mRNA–tRNA interface". In: *Genome Research*, gr–176784.
Shen, Yanting et al. (2017). "Screening effective differential expression genes for hepatic carcinoma with metastasis in the peripheral blood mononuclear cells by RNA-seq". In: *Oncotarget* 8.17, p. 27976.
Shi, Jiandong et al. (2018). "Deep RNA Sequencing reveals a repertoire of human fibroblast circular RNAs associated with cellular responses to herpes simplex virus 1 infection". In: *Cellular Physiology and Biochemistry* 47.5, pp. 2031–2045.
Smith, Temple F and Michael S Waterman (1981). "Identification of common molecular subsequences". In: *Journal of molecular biology* 147.1, pp. 195–197.

Smith, Tom, Andreas Heger, and Ian Sudbery (2017). "UMI-tools: modeling sequencing errors in Unique Molecular Identifiers to improve quantification accuracy". In: *Genome Research* 27.3, pp. 491–499.

Soneson, Charlotte, Michael I Love, and Mark D Robinson (2016). "Differential analyses for RNA-seq: transcript-level estimates improve gene-level inferences [version 2; peer review: 2 approved]". In: *F1000Research* 4:1521.

Šošić, Martin and Mile Šikić (2017). "Edlib: a C/C++ library for fast, exact sequence alignment using edit distance". In: *Bioinformatics* 33.9, pp. 1394–1395.

Srivastava, Avi et al. (2016a). "Accurate, Fast and Lightweight Clustering of de novo Transcriptomes using Fragment Equivalence Classes". In: *arXiv preprint arXiv:1604.03250*.

Srivastava, Avi et al. (2016b). "RapMap: a rapid, sensitive and accurate tool for mapping RNA-seq reads to transcriptomes". In: *Bioinformatics* 32.12, pp. i192–i200.

Srivastava, Avi et al. (2019a). "Alevin efficiently estimates accurate gene abundances from dscRNA-seq data". In: *Genome Biology* 20.1, p. 65.

Srivastava, Avi et al. (2019b). *Alevin efficiently estimates accurate gene abundances from dscRNA-seq data: data*. DOI: 10.5281/zenodo.2583228. URL: https://zenodo.org/record/2583228.

– (2019c). *Alevin efficiently estimates accurate gene abundances from dscRNA-seq data: github*. URL: https://github.com/COMBINE-lab/salmon.

– (2019d). *Alevin efficiently estimates accurate gene abundances from dscRNA-seq data: source Code*. DOI: 10.5281/zenodo.2583275. URL: https://zenodo.org/record/2583275.

Stuart, Tim et al. (2019). "Comprehensive Integration of Single-Cell Data". In: *Cell*.

Suzuki, Hajime and Masahiro Kasahara (2018). "Introducing difference recurrence relations for faster semi-global alignment of long sequences". In: *BMC Bioinformatics* 19.1, p. 45.

Svensson, Valentine, Roser Vento-Tormo, and Sarah A Teichmann (2018). "Exponential scaling of single-cell RNA-seq in the past decade". In: *Nature protocols* 13.4, p. 599.

Svensson, Valentine et al. (2017). "Power analysis of single-cell RNA-sequencing experiments". In: *Nature Methods* 14.4, p. 381.

Tambe, Akshay and Lior Pachter (2019). "Barcode identification for single cell genomics". In: *BMC bioinformatics* 20.1, p. 32.

Tang, Fuchou et al. (2009). "mRNA-Seq whole-transcriptome analysis of a single cell". In: *Nature methods* 6.5, p. 377.

Tang, Hengli et al. (2016). "Zika virus infects human cortical neural progenitors and attenuates their growth". In: *Cell stem cell* 18.5, pp. 587–590.

Tian, Luyi et al. (2018). "scPipe: A flexible R/Bioconductor preprocessing pipeline for single-cell RNA-sequencing data". In: *PLoS Computational Biology* 14.8, e1006361.

Trapnell, Cole, Lior Pachter, and Steven L Salzberg (2009). "TopHat: discovering splice junctions with RNA-seq". In: *Bioinformatics* 25.9, pp. 1105–1111.

Trapnell, Cole et al. (2013). "Differential analysis of gene regulation at transcript resolution with RNA-seq". In: *Nature biotechnology* 31.1, pp. 46–53.

Turro, Ernest et al. (2011). "Haplotype and isoform specific expression estimation using multi-mapping RNA-seq reads". In: *Genome Biology* 12.2, R13.
Van Dongen, Stijn (2000). "A cluster algorithm for graphs". In: *Report-Information systems* 10, pp. 1–40.
Venter, J Craig et al. (2001). "The sequence of the human genome". In: *science* 291.5507, pp. 1304–1351.
Vieth, Beate et al. (2017). "powsimR: power analysis for bulk and single cell RNA-seq experiments". In: *Bioinformatics* 33.21, pp. 3486–3488.
Vigna, Sebastiano (2008). "Broadword implementation of rank/select queries". In: *Experimental Algorithms*. Springer, pp. 154–168.
Vincent, Matt and Kwangbom "KB" Choi (2017). *Churchill-Lab/G2Gtools: V0.1.31*. DOI: 10.5281/zenodo.292952. URL: https://zenodo.org/record/292952.
Vuong, Hy et al. (2018). "A revisit of RSEM generative model and its EM algorithm for quantifying transcript abundances". In: *BioRxiv*. DOI: https://doi.org/10.1101/503672.
Wang, Zhong, Mark Gerstein, and Michael Snyder (2009). "RNA-Seq: a revolutionary tool for transcriptomics". In: *Nature Reviews Genetics* 10.1, pp. 57–63.
Wu, Douglas C et al. (2018). "Limitations of alignment-free tools in total RNA-seq quantification". In: *BMC Genomics* 19.1, p. 510.
Yi, Lynn et al. (2018). "A direct comparison of genome alignment and transcriptome pseudoalignment". In: *BioRxiv*. DOI: https://doi.org/10.1101/444620.
Zakeri, Mohsen et al. (2017). "Improved data-driven likelihood factorizations for transcript abundance estimation". In: *Bioinformatics* 33.14, pp. i142–i151.
Zappia, Luke (2019). "Tools and techniques for single-cell RNA sequencing data". In: DOI: 10.5281/zenodo.2673034. URL: https://zenodo.org/record/2673034.
Zhang, Chi et al. (2017). "Evaluation and comparison of computational tools for RNA-seq isoform quantification". In: *BMC genomics* 18.1, p. 583.
Zhang, Zhaojun and Wei Wang (2014). "RNA-Skim: a rapid method for RNA-Seq quantification at transcript level". In: *Bioinformatics* 30.12, pp. i283–i292.
Zhao, Lu et al. (2017). "Bartender: a fast and accurate clustering algorithm to count barcode reads". In: *Bioinformatics*.
Zheng, Grace XY et al. (2017). "Massively parallel digital transcriptional profiling of single cells". In: *Nature Communications* 8, p. 14049.
Zhu, Anqi et al. (2019). "Nonparametric expression analysis using inferential replicate counts". In: *BioRxiv*, p. 561084.
Ziegenhain, Christoph et al. (2017). "Comparative analysis of single-cell RNA sequencing methods". In: *Molecular cell* 65.4, pp. 631–643.