Computational Problems for RNA-Seq Data Analysis

Shunfu Mao

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2020

Reading Committee:

Sreeram Kannan, Chair

Georg Seelig

Walter Larry Ruzzo

Program Authorized to Offer Degree:
Electrical and Computer Engineering

University of Washington

Abstract

Computational Problems for RNA-Seq Data Analysis

Shunfu Mao

Chair of the Supervisory Committee:
Assistant Professor Sreeram Kannan
Electrical and Computer Engineering

High throughput sequencing of RNA (RNA-seq) has become a staple in modern molecular biology, with a wide range of applications including RNA transcripts assembly, variants detection, and gene expression estimation for downstream cellular analysis. RNA-seq data is therefore able to provide us with unprecedented insights into cellular organisms. However, they have also introduced a new set of computational challenges because of the nature of the sequenced RNA transcripts and an ever increasing number of RNA-seq experiments. For instance, the RNA transcripts have different expression levels, making the sequenced reads potentially unable to fully cover some lowly expressed gene regions. In addition, the RNA transcripts also share many repetitive patterns, making it ambiguous to determine the regions where some RNA-seq reads are actually sampled. Moreover, there are still many laborious procedures in the RNA-seq data analysis, making it difficult to keep pace with the constantly produced large amounts of RNA-seq data. There is an urgent need for better computational methods that are able to analyze the RNA-seq data more accurately and efficiently.

Motivated by this, in the thesis, we have presented novel computational solutions for three computational problems for RNA-seq data analysis:

Firstly, we have developed RefShannon - a new genome-guided RNA transcripts (transcriptome) assembly software. RefShannon reconstructs RNA transcripts, based on the align-

ments of RNA-seq reads onto a reference genome. It exploits the pair-end linking information of RNA-seq reads, and the varying expressions of RNA transcripts, in enabling an accurate reconstruction of the transcripts. Experiments demonstrate RefShannon has superior assembly performance over the state-of-art genome-guided assembly tools.

Next, we have developed abSNP - a new RNA-seq SNP calling software. AbSNP detects SNPs in expressed gene regions, based on the alignments of RNA-seq reads onto a reference transcriptome. It exploits the mapping quality scores of RNA-seq reads, and the varying expressions of different genes. AbSNP is a cost-effective method as it requires no additional DNA-seq. It is also able to call SNPs with significantly improved sensitivity in repetitive gene regions, while other RNA-seq SNP callers are unable to make any calls in such regions.

Finally, we have developed CellMeSH - a new web server and API package for automatic cell-type identification in single-cell RNA-seq (scRNA-seq) analysis. CellMeSH predicts cell types, based on a set of marker genes as query input. CellMeSH builds its database in a scalable and easy-to-update way using prior literature, and adopts a novel probabilistic method to better query the database. Through a variety of experiments on human and mouse scRNA-seq datasets, CellMeSH has demonstrated richer gene and cell-type information in its database, robust query method, and an overall superior annotation performance.

TABLE OF CONTENTS

ACKNOWLEDGMENTS

I feel extremely fortunate to have been working with my advisor Prof. Sreeram Kannan in the last few years. His great enthusiasm towards the research, sharp insights into the problems and many innovative ideas have deeply inspired me. He has provided me with so much freedom, spent so much time discussing with me on various topics to help expand my vision and introduced me to work with so many amazing researchers of different expertise. His great patience, generosity and amazing personality have set role models for me, which will be so important for me to learn for the rest of my career.

I am deeply grateful to Prof. Georg Seelig to provide me with such a great opportunity to work in the interdisciplinary area of single-cell analysis and natural language processing (NLP). He has provided me with so many important directions toward my work. He has also spent tremendous efforts to help me to better present my thoughts. He has been so kind to me and is always available to help me. This thesis would have been impossible without him.

I would also like to thank my other thesis committee members Prof. Walter Larry Ruzzo and Dr Yuliang Wang, for agreeing to serve on my committee and taking time for my defense. I am really appreciated that I can have their suggestions and comments for my work.

I would like to thank my other collaborators: Prof. Soheil Mohajer and Prof. Suhas Diggavi. Thanks for having me work with you. We had so many fruitful discussions from which I have learned a lot. I would also like to thank Prof. Lior Pachter, Prof. David Tse, and Prof. Kannan Ramachandran. Thanks for providing me with the opportunities to work on these problems. Some of the works have served as the main chapters of this thesis.

I would like to thank many professors at University of Washington (UW) who have helped me during my PhD study. I am really grateful to Prof. Matt Reynolds for providing TA

opportunity to me after I initially came to UW. I also feel so fortunate to get much extra advice from Prof. Mari Ostendorf to guide me on the NLP study. I would also like to thank Prof. James A. Ritcey, Prof. Eric Klavins, and Prof. Su-In Lee for their wonderful classes that have built my knowledge across many fields including signal processing and synthetic/computational biology, and their precious time to serve my qualifying exam and offer great advices to me.

I would also like to thank my previous advisors Prof. Jungwoo Lee, from my master study, and Prof. Fang Li, from my undergraduate study. They had spent tremendous efforts to guide me in research and helped me so much for my PhD application. Even after many years, their enthusiasms and advices have always been so precious to me.

I am really grateful to get opportunities to co-work with many brilliant students: Yue Zhang, Dhaivat Joshi, Joshua Fan and Edwin Basil Mathew. Your expertises and persistences in the projects have impressed me so much and helped me learn a lot.

I feel so proud of having my amazing labmates Eugene Lin, Himanshu Asnani, Sudipto Mukherjee, Arman Rahimzamani, Yihan Jiang, Bowen Xue, Viswa Virinchi and Soubhik Deb. I really enjoy our occasionally "unsupervised" weekly seminars, inspiring discussions, and consistently helping and supporting each other like a family.

Finally, I want to thank my family. My parents and parents-in-law, far away as they are, have always been there supporting me. And my dear wife and dear daughter: thanks for being with me, supporting and encouraging me; nothing is more blessing and important than having you in my life, and nothing would have been possible for me without you.

DEDICATION

To my dear wife Xian and my dear daughter Linyue

Chapter 1

INTRODUCTION

1.1 Background

In many higher organisms including mammals, the genomic DNA is comprised of thousands of genes, along with other sequences that regulate when, where and how the genes are produced. These genes have distinct regions called exons and introns [1, 2]. By combining its exons in various ways - a procedure called alternative splicing [3, 4, 5], a gene, especially in eukaryotes, can produce multiple different messenger RNAs (mRNA [6]), typically in the form of single-stranded nucleotide sequences. mRNAs present the genetic information contained in DNA sequences and direct the synthesis of different protein products that perform vast array of biological functions [7, 8].

Ever since mRNAs were initially discovered in 1960 [9], researchers have been working persistently to get an enhanced understanding of mRNAs and their biological impacts. In light of this, there are several important questions listed as follows. Question 1. How to better detect new mRNAs? Newly detected mRNAs could help us find novel proteins and related biological functions never seen before [10]. Question 2. How to better detect the variants in the expressed genes? Some variants in mRNAs may be transcribed from the underlying genes and hence could indicate health issues such as genetic disorders [11], diseases [12], or even cancer [13]. Question 3. How to better analyze the cells based on their mRNA expressions? Different mRNAs are expressed with different levels in the living cells, so the mRNA expressions (or aggregated into gene expressions) could serve as the cell specific signatures, which could be used to distinguish the cell types in a cell population. Understanding the specific cell types in a cell population could aid in disease and development analysis. For instance, a higher portion of immune-related cell types could indicate that the

cells are disease infected [14].

To aid in answering these questions, we first need mechanisms to measure the mRNAs. There are several commonly used methods to measure the mRNAs [15], among which RNA-seq [16] is currently the most popular one. The RNA-seq method first extracts mRNAs from an organism, next fragments and copies them into double-stranded complementary DNAs (ds-cDNAs), and finally sequences the ds-cDNAs by high-throughput short read sequencing techniques. The other mRNA measurement methods include quantitative real-time polymerase chain reaction (qPCR) [17], microarrays [18], and recently long read sequencing such as PacBio [19] and Nanopore [20]. The key issue for qPCR and microarray is that they are only able to measure the expression levels of prior-known mRNAs, whereas RNA-seq can be used to access previously unknown mRNAs. Though PacBio and Nanopore are also able to sequence unknown mRNAs (with even longer reads), they are not mature yet as their read error-rates are still higher (10% to 15%) than RNA-seq (0.1%), and they are also more expensive [21, 22]. In addition, RNA-seq recently can be applied to single-cells [23, 24, 25], which is useful to reveal the heterogeneity of cell population at single-cell resolution.

Given the RNA-seq as the mRNA measurements, we also need computational methods to analyze them, in order to answer the previously imposed questions. Various methods and tools for RNA-seq data analysis have been developed for this purpose. However, there are still many challenging issues.

As for the first question of how to better detect mRNAs using RNA-seq, this is essentially a transcriptome (i.e. set of mRNA transcripts) assembly problem, where the goal is to obtain a complete and accurate recovery of transcriptome based on observed RNA-seq reads. To handle this problem, existing methods follow a graph construction step, and then a graph traversal step. There are several limitations to existing methods. For instance, the linking information among pair-end reads is not properly utilized [26]. It is possible that some mRNA transcripts have low expression levels, which could make the constructed graph more fragmented if the pair end information is not well explored. In addition, the existing graph traversal algorithms are mainly designed by intuitions and thus sub-optimal [27]. Due

to alternative splicing, it is possible that different sets of transcripts can yield the same observation of RNA-seq reads. A sub-optimal traversal algorithm could fail to resolve such ambiguities.

As for the second question of how to detect the variants in the expressed genes using RNA-seq, this is essentially an RNA-seq variant calling problem. Here our goal is to find variants (in particular single nucleotide polymorphisms) in the expressed gene regions. To achieve this goal, we could use many existing methods [28, 29, 30, 31, 32] that are specially designed to call variants in the whole genome or exome regions using DNA-seq, and then use RNA-seq to filter out the variants in non-expressed regions. Such kind of methods are not cost-effective as they require additional DNA sequencing, which samples reads from the genome instead of from the mRNA transcripts. There are only limited number of works to call variants using RNA-seq directly [33, 34, 28, 35], however, there could be many RNA-seq reads that could be multiply mapped to many mRNA transcripts. Existing RNA-seq callers will discard all of them and are not able to make any reliable calls in repetitive regions.

As for the third question of how to better analyze the cells based on the mRNA expressions, this is related to the single-cell RNA-seq analysis, where we typically process the raw RNA-seq reads into a gene-cell matrix, cluster the matrix and annotate the clusters with cell types, based on cluster specific genes. The last step of cell-type annotation remains an open problem. Existing methods usually follow the way of preparing a database first and then querying it based on cluster specific genes. A key issue for these methods is that the underlying databases that connect genes and cell types are usually hand-curated from literature. There could be limited gene-cell coverage, and it is also laborious to update the databases.

In this thesis, we address these three computational problems for RNA-seq data analysis, with the ultimate goal to better understand mRNAs and the related biological systems (e.g. cells). We have designed novel computational methods and developed related open source software to address the above open issues that exist in these problems.

1.2 Contributions

In this section, we highlight our contributions in developing novel computational solutions for the three computational problems for RNA-seq data analysis.

Contributions to RNA-seq transcriptome assembly. We have presented RefShannon - a new genome-guided transcriptome assembler. RefShannon is able to utilize the pair end reads to infer the low-expressed mRNA transcripts that otherwise would have been missed in a fragmented graph. RefShannon also adopts the novel sparse flow decomposition algorithm that was initially proposed in the Shannon assembler [36], which was proved to work toward optimal transcriptome assembly in theory. Our evaluation shows effective performance gain of RefShannon over state-of-art genome-guided assemblers for both simulated and real datasets. RefShannon has been open sourced at Github[1].

Contributions to RNA-seq variant calling. We have presented abSNP [37] - a novel RNA-seq SNP caller that is able to call SNPs even in repetitive regions. AbSNP is able to call variants using RNA-seq directly without requiring to use additional DNA-seq first, so it is cost-effective. In comparison to existing RNA-seq variant callers, abSNP shows comparable precision and recall in non-repetitive regions. In addition, abSNP is able to get significantly improved sensitivity in repetitive regions while existing methods are unable to make any calls. AbSNP has been open sourced at Github[2].

Contributions to single-cell RNA-seq analysis. We present CellMeSH - a web server and API to annotate clustered single-cell data using indexed literature. CellMeSH takes input of marker genes and outputs a ranked list of predicted cell types. Compared to existing approaches, it builds a database in a scalable and easy-to-update way, by automatically linking the gene and cell-type information from millions of publications. It queries the noisy database by a novel probabilistic query method based on maximum likelihood estimation. Through a variety of experiments on human and mouse scRNA-seq datasets, CellMeSH has

[1] `https://github.com/shunfumao/RefShannon`

[2] `https://github.com/shunfumao/abSNP`

demonstrated richer gene and cell-type information in its database, robust query method, and an overall superior annotation performance. CellMeSH has been integrated as the default annotation option into our recently developed web server UNCURL-App [38] for interactive single-cell analysis. CellMeSH has its standalone website[3] and we have also open sourced the API at Github[4].

1.3 Organization

The thesis is organized in the following structure.

- Chapter 2 explains basic concepts helpful to understand the remaining chapters.
- Chapter 3 discusses our contributions to the RNA-seq transcriptome assembly problem.
- Chapter 4 discusses our contributions to the RNA-seq variant calling problem.
- Chapter 5 discusses our contributions to the single-cell RNA-seq annotation problem.
- Chapter 6 discusses future directions.

[3] `https://uncurl.cs.washington.edu/db_query`

[4] `https://github.com/shunfumao/cellmesh`

Chapter 2

BACKGROUND

Though different chapters are mostly self-inclusive, here we explain several basic concepts, which can be helpful to understand the remaining chapters. Specifically, Section 2.1 explains assembly graphs and is related to Chapter 3, Section 2.2 explains probability estimation and is related to both Chapter 4 and Chapter 5, Section 2.3 explains statistical tests and is related to Chapter 5.

2.1 Assembly

2.1.1 Kmer graph for de novo transcriptome assembly

Kmer graphs (or De-Bruijn graph [39]) are commonly used in de novo genome and transcriptome assembly [40, 41, 36]. It is a directed graph where each node represents a sequence of k letters. Each edge (u, v) represents a sequence of $k + 1$ letters and the source node u contains the first k letters of the edge and the destination node v contains the last k letters of the edge.

A simple workflow of de novo assembly using kmer graph is illustrated in Fig 2.1 (a). We have three reads, each of length 4, sampled from target RNA transcript "ABCDEF"[1]. Each read represents two nodes connected by an edge in the graph. For instance, read "ABCD" is split into 3-mer "ABC" and 3-mer "BCD" and they are connected by an edge representing "ABCD". The resulting kmer graph contains four nodes and three edges. In this simple example, the target RNA transcript can be successfully reconstructed by traversing the flow path through the nodes "ABC", "BCD", "CDE" and "DEF".

[1]Here we use English alphabets for better illustration.

However, it is common in de novo assembly that the kmer graph could be fragmented especially when a transcript is lowly expressed, as illustrated in 2.1 (b). Here we only have two sampled reads "ABCD" and "CDEF" and the resulting kmer graph is disconnected. Consequently, we are unable to recover the original "ABCDEF" sequence.

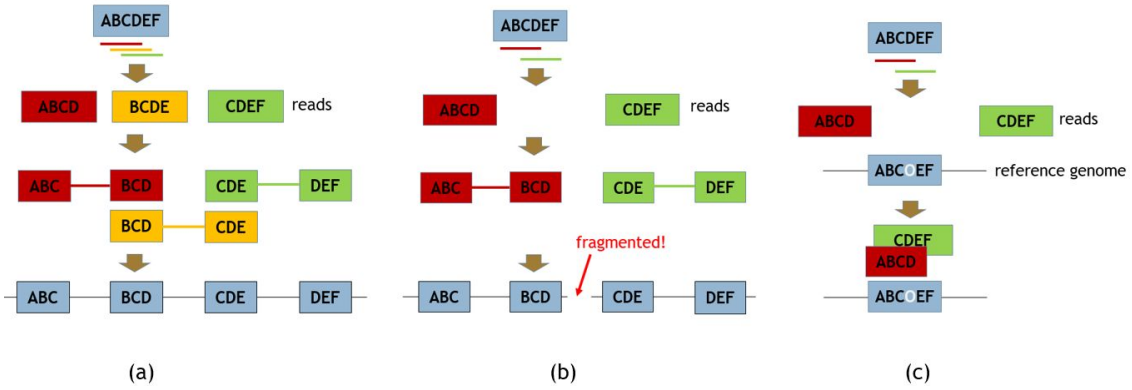


Figure 2.1: **Example of (a) kmer graph (b) fragmented kmer graph (c) splice graph**

2.1.2 Splice graph for genome-guided transcriptome assembly

Unlike de novo transcriptome assembly, genome-guided transcriptome assembly utilizes extra reference genome, and the assembly is commonly conducted in a spliced graph [27, 42]. A splice graph is a directed graph where each node represents an exon region, and each edge (u, v) represents there's a read aligned onto the two exon regions corresponding to u and v.

A simple workflow of genome-guided assembly is illustrated in Fig 2.1 (c). We also sampled two reads "ABCD" and "CDEF" from target RNA transcript "ABCDEF". We first align them onto a reference genome "ABCOEF". Note the reference genome is different from the target RNA transcript. Still, it is possible to align the reads onto the genome. Here the resulting splice graph is a simple one-node graph, and we have recovered "ABCOEF" as the RNA transcript. Whereas de novo assembly is unable to achieve this, the genome-guided assembly is able to recover most of the transcript.

2.2 Probability

2.2.1 Maximum a posteriori (MAP) estimation

The MAP estimation [43] tries to guess what is the best possible hidden X given the observed Y. It has been used in designing the variant calling algorithm in Chapter 4.

Consider two discrete random variables X as unknown and Y as observed. We are interested to estimate X given the observation of Y. One way is to estimate X which maximizes the posterior distribution $P_{X|Y}$. Formally,

$$\hat{x}_{MAP} = max_x P(X|Y) \tag{2.1}$$

From Bayesian principles, we know that:

$$P(X,Y) = P(X|Y)P(Y) = P(Y|X)P(X) \tag{2.2}$$

Apply Equation 2.2 into Equation 2.1, and remove $P(Y)$ (as it does not affect the maximization procedure), we get:

$$\hat{x}_{MAP} = max_x P(Y|X)P(X) \tag{2.3}$$

Essentially $\hat{x}_{map}$ indicates that if we observe a known quantity y, what is the most-likely unknown quantity x. It requires the distribution of X as prior knowledge.

2.2.2 Maximum likelihood (ML) estimation

The ML estimation is a special case of MAP estimation, as it is possible that we have no prior knowledge of X. The ML estimation has been used in designing the database query method in Chapter 5.

Intuitively it tries to estimate X given the observation of Y by estimating X which will maximize the chance of seeing Y (what X best explains Y). Formally,

$$\hat{x}_{ML} = max_x P(Y|X) \tag{2.4}$$

The ML estimation is essentially MAP estimation with uniform prior $P(X)$.

2.3 Gene set enrichment analysis

Gene set enrichment analysis [44] aims at using statistical approaches to find known groups of genes that are significantly enriched for the input genes. The known groups of genes could come from existing knowledge sources such as Gene Ontology (GO) [45] and MSigDB [46]. In this section, we explain several common approaches.

2.3.1 Hypergeometric test

Hypergeometric test (or Fisher's exact test [47], or Over Representation Analysis) is a simple and effective statistical test that has been widely used in various gene set enrichment analysis [48, 49]. Let c be a known group of K genes (c could be a GO term or a cell type). Let N be the total number of genes. Let q be a query input of n genes. Let k be the number of overlapping genes between c and q. Then the probability of k genes existing in both c and q can be written as: $P(k|q,c) = \frac{\binom{K}{k}\binom{N-K}{n-k}}{\binom{N}{n}}$. A smaller probability indicates higher significance for the group c to be enriched for the input genes q.

2.3.2 Gene Set Enrichment Analysis (GSEA)

The GSEA method [46] is based on Kolmogorov-Smirnov test [50]. The method takes input of genome-wide gene expression profiles from samples belonging to two phenotype classes. To calculate an enrichment score between the input genes (q) and a known group of genes (c), the genes in q are first sorted by their weights (correlation with phenotype). A score is calculated by walking down the sorted genes in q, increasing a running-sum statistic by the gene weight if the gene is in c, and decreasing the running-sum statistic if the gene is out

of c. The running-sum's maximum deviation from zero in the random walk serves as the enrichment score of c for q.

2.3.3 Gene Set Variance Analysis (GSVA)

The GSVA method [51] is also based on Kolmogorov-Smirnov test [50]. The method takes input of gene expression matrix from microarray or RNA-seq experiments. The weight of a gene i in sample j is first adjusted in the context of sample population distribution, then replaced by the gene i's rank in sample j. The calculation of an enrichment score between the input genes (q) of a sample and a known group of genes (c) is similar to GSEA method. The key differences of GSVA from GSEA are: GSVA takes input of microarray or RNA-seq expression, and during its enrichment score calculation, the genes' integer ranks are essentially utilized. As benchmarked in [52], GSVA shows a top performance in assigning cell types to clustered single-cells.

Chapter 3

REFSHANNON: GENOME-GUIDED TRANSCRIPTOME ASSEMBLY USING SPARSE FLOW DECOMPOSITION

3.1 Introduction

In many higher organisms including mammals, the genomic DNA is comprised of thousands of genes, along with other sequences that regulate when, where and how the genes are produced. These genes have distinct regions called exons and introns [1, 2]. By combining its exons in various ways - a procedure called alternative splicing [3, 4, 5], a gene, especially in eukaryotes, can produce multiple different messenger RNAs (mRNA [6], or RNA transcript in this chapter) that can be converted into different protein products in cells.

Transcriptome is the set of RNA transcripts. With RNA sequencing technology [53], we can now obtain millions of short RNA fragments (RNA-seq reads) from the transcriptome. The transcriptome assembly problem [54] is to obtain a complete and accurate recovery of transcriptome based on observed RNA-seq reads. It helps us find new RNA transcripts and their expression levels (or abundance) in order to better understand proteins and cells.

Transcriptome assembly is not an easy task due to several factors. Because of alternative splicing, it is possible that different set of transcripts can yield the same observation of RNA-seq reads. Adding to this complexity is the fact that distinct transcripts are expressed at different expression levels [55]. Thus, transcripts which have low expression levels are harder to reconstruct; also, transcripts which are part of complex isoform [4] families are difficult to assemble correctly.

There are two kinds of transcriptome assembly problems [54]: de novo assembly and genome-guided (or reference-based) assembly. They usually first construct an assembly graph from reads, and then traverse the graph in order to recover transcripts. For de novo assembly,

there is no knowledge other than that of the observed reads. This is common in non-model organisms or when an approach unbiased by prior knowledge is needed (e.g. cancer transcriptome). For genome-guided assembly, in addition to the observed reads, there is also knowledge of the genome of the organism. This is common in scenarios where a model organism[56] is sequenced.

De novo assembly is typically more challenging and less accurate than genome-guided assembly, since the latter utilizes additional side information. While algorithms and software packages are available for both the de novo (TransAByss [57], Trinity [58], OASES [59], SOAPdenovo-Trans [60] etc.) and genome-guided assembly problems (Scripture [61], Cufflinks [26], StringTie [27], TransComb [62] and CLASS2 [63],Ryuto[42], Strawberry[64], Trinity (reference-guided mode) [58] etc.), much remains to be done [65, 66]. Recently Kannan et al [36] developed an assembler called Shannon assembler that utilized principles from information theory to solve the de novo transcriptome assembly problem, and demonstrated benefits over state-of-the-art assemblers. Here we develop a genome-guided assembler called RefShannon by exploiting the framework from Shannon.

To begin with we note that, while genome-guided algorithms have an advantage in general, Shannon is able to recover even more transcripts than the leading genome-guided assembler StringTie when the coverage of the transcripts is high. We attribute this to the careful utilization of transcript abundances while performing assembly in Shannon. However, as expected, for transcripts of low abundance, Shannon is inferior because the k-mer graph utilized by Shannon needs a higher coverage in order to stay connected. Therefore, it should be possible to design a genome-guided assembler, that combines the superior reconstruction method of Shannon along with the aid of the genome side-information in order to deliver optimal performance. This is the main motivation for this work - to build a superior genome-guided assembler.

RefShannon takes a graph preparation step and graph traversal step as most existing methods do, but it has adopted several main ideas (as summarized in Fig 3.1) that differentiate it from the existing methods and bring its superior performance.

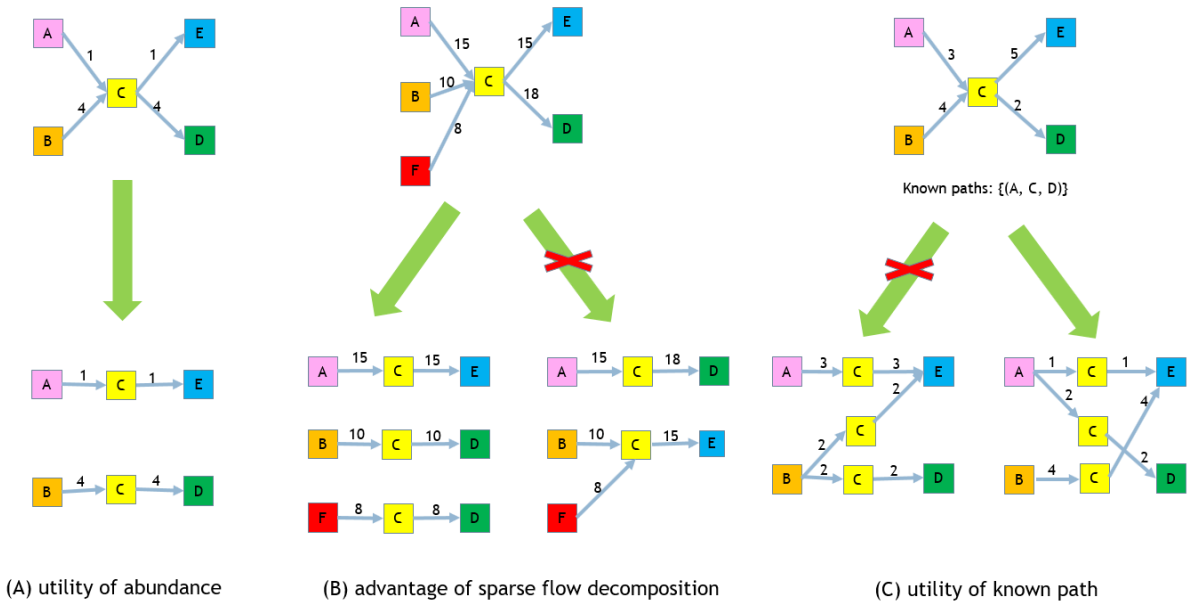


Figure 3.1: **Main ideas of RefShannon**. In these conceptual examples, each graph means a splice graph where nodes represent exonic regions and edges indicate there are reads aligned across the nodes. The edge weights are abundance, calculated by the number of supporting reads. (A) RefShannon explores the utility of varying abundance, which is essential for a correct decomposition of paths. For example, given that AC, CE, BC, CD have weights 1, 1, 4, 4 respectively, we're confident to decompose the graph into two paths as ACE and BCD. (B) RefShannon adopts a sparse flow algorithm that tries to find the minimum number of paths that explains edge weight constraints (e.g. the decomposition of the graph into ACE with weight 15, BCD with weight 10 and FCD with weight 8 explain the splice graph well). If we follow a greedy approach (as used by StringTie) that iteratively finds and removes the heaviest path, we will get an inaccurate transcript ACD first. (C) RefShannon extracts known path information from read alignments to resolve decomposition ambiguity. For example, when the node C is decomposed, a flow ambiguity (i.e. non-unique decomposition) happens: both ACE, BCE, BCD and ACE, ACD, BCE explain node C's edge weight constraints. Suppose ACD is a known path, then the first decomposition of ACE, BCE, BCD can be excluded.

One of the main ideas of RefShannon is to utilize the varying abundance information while traversing the splice graph (Fig 3.1A). This information is essential for a correct decomposition of flow paths. Cufflinks[26] does not exploit the abundance information during assembly; instead, it relies on overlap graphs, which require a strict partial order between

read alignments. Consequently, Cufflinks may throw away reads (especially pair end reads) of uncertain compatibility, while those read alignments can contain useful information to construct a more accurate graph.

The second idea RefShannon has adopted is the sparse flow decomposition algorithm that was initially proposed in the Shannon assembler [36], which applies linear programming to efficiently decompose for the minimum number of paths (flows) at each node restricted by the node's in-edge and out-edge weights. This approach has been proved in [36] to work toward optimal transcriptome assembly. The similar goal of parsimonious assembly is pursued by Cufflinks [26], which is based on overlap graph and does not exploit abundance information. StringTie [27] also explores the abundance information, but it follows a sub-optimal greedy approach by iteratively extracting the heaviest path as transcript. As illustrated in Fig 3.1B, this will lead to incorrect assembly.

The third idea RefShannon has adopted is to utilize pair end reads to supplement additional edge and path information so that an originally disconnected transcript could be found. A similar idea was also used in Scripture [61] and Ryuto [42]. However, Scripture exhaustively enumerates all possible transcript candidates first and filter them later based on certain significance criteria. Ryuto has also adopted such information mainly during the graph construction stage. Differently, RefShannon has also applied these additional edge and path information in the graph decomposition stage so that certain assembly ambiguity can be resolved (Fig 3.1C).

By incorporating ideas, we have developed RefShannon as an open-source Python software[1]. Based on the experiments using both simulated and real datasets, RefShannon has demonstrated superior assembly performance than existing genome-guided assembly tools.

The remained chapter is organized as follows. We first describe how RefShannon works. We next show its assembly performance in terms of ROC curve using simulated datasets, and sensitivity using real datasets. We then explain its computational resources required.

---

[1] `https://github.com/shunfumao/RefShannon`

Lastly, we will conclude our work and discuss future directions.

3.2 Methods

3.2.1 Overall workflow

To do genome-guided transcriptome assembly based on sampled RNA-seq reads, RefShannon first aligns the reads onto the reference genome (Fig 3.2 step 1). The alignment is conducted by using an external tool (STAR [67], Tophat2 [68], Hisat2 [69], GMAP [70], minimap2 [71] etc) that is able to capture splice events, so that a read crossing two distinct exons of an RNA transcript can be split aligned onto the genome.

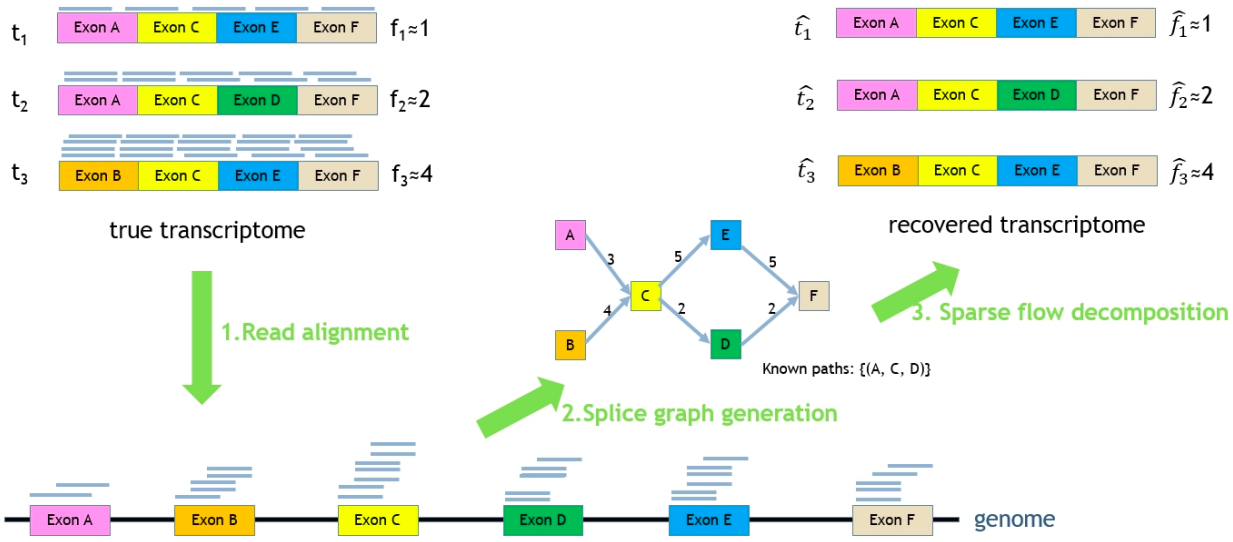


Figure 3.2: **Overall flow of RefShannon**. In this example, the true transcriptome contains 3 RNA transcripts t_1, t_2, t_3 with expression levels $f_1 = 1$, $f_2 = 2$, $f_3 = 4$. RNA reads are sampled from the transcriptome first, and are aligned onto a reference genome using external aligners. RefShannon will take the read alignment as input, generate splice graphs, and apply a novel sparse flow decomposition algorithm to recover the transcriptome as $\hat{t}_i$ $i \in \{1, 2, 3\}$. The corresponding $\hat{f}_i$ estimates related abundance.

Based on read alignments, RefShannon produces splice graphs (Fig 3.2 step 2). The graph has each node to represent a unique exonic region that is supported by aligned reads

and each edge between two nodes to imply there exist reads going through those nodes. Additional known paths are also collected, indicating that the nodes along the path belong to some transcript, in order to resolve the ambiguity that could occur.

Based on splice graphs, RefShannon decomposes the graphs to reconstruct the transcripts (Fig 3.2 step 3). Specifically, it applies a sparse flow decomposition algorithm, originally proposed in [36], to reconstruct the minimum number of flow paths (as assembled transcriptome) that satisfy node and edge constraints.

3.2.2 Splice graph generation

Based on the read alignments (Fig 3.3A), RefShannon conducts the splice graph generation in three steps: split, merge and connect (Fig 3.3B-D).

In the split step (Fig 3.3B), we divide inferred exon regions (Fig 3.3A) into sub exon regions where splice junctions may occur. Splice junctions represent the exon or sub-exon boundaries where alternative splicing could occur. They can be determined according to the read alignments, because a read sampled across two exon parts in a transcript can be split aligned onto disconnected genome regions, with the locus where the read leaves as the splice donor and the locus where the read enters as the splice acceptor. For example, a 100-bp read could be split-aligned onto the genome (chromosome 15) at loci $[78837259, 78837318]$ and loci $[78837519, 78837558]$, then we consider there is a splice junction at locus 78837318 (as splice donor) and at locus 78837519 (as splice acceptor). Once we determine a splice junction (splice donor or acceptor) in the middle of an exon, we need to divide the exon further into sub exons, because the splice junction indicates that a divided sub exon could link to another non-adjacent (sub) exon that may belong to the same RNA transcript.

In the merge step (Fig 3.3C), we empirically merge two exon regions based on their gap and expression levels in order to reduce the chances that the gap is actually part of exon but not covered by reads. For example, we merge two exon regions if they are very close to each other. Statistics have shown that less than 0.01% of introns are smaller than 20 bp in length [72], so a gap within 10 bp is highly unlikely intronic but should be an uncovered

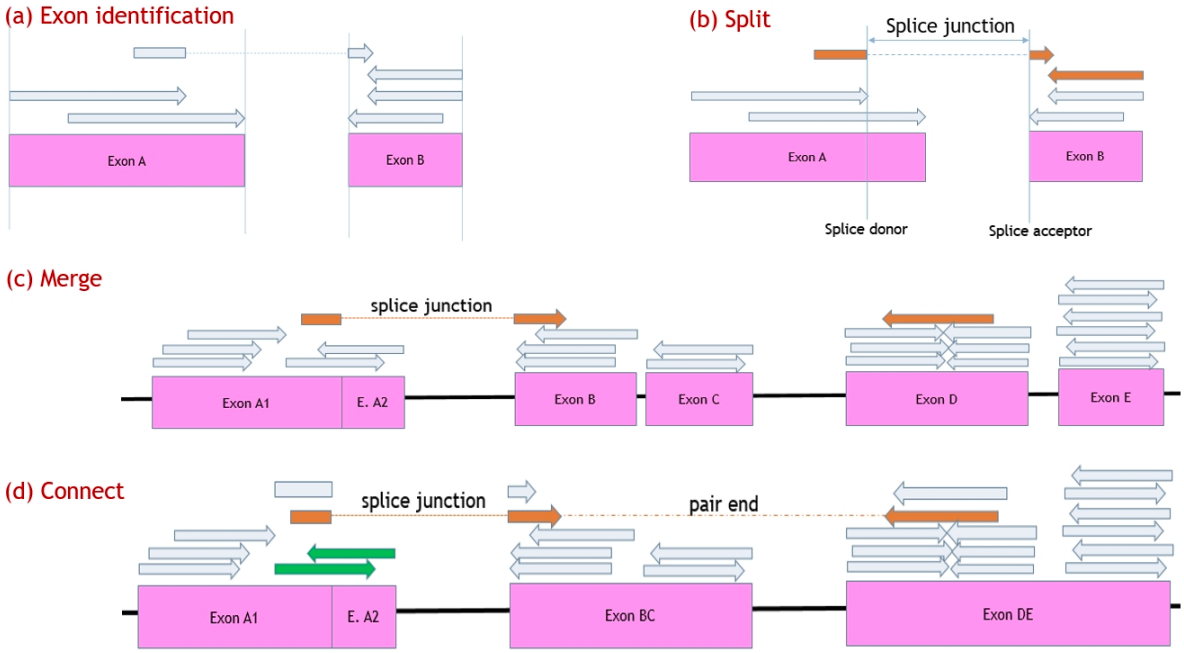


Figure 3.3: **Illustrations of splice graph generation** (A) Exon regions are identified based on read alignments. (B) A splice event happens in the middle of Exon A, which implies Exon A should be further split into two sub exons at splice donor location. (C) Exon A1 and A2 should not be merged due to a splice event after A1. Exon B and C shall be merged if their gap is small. Exon D and E shall be merged if their gap is moderate but their coverage is high. (D) Exon A1 and A2 are connected because there're read alignments (green color) crossing them. Exon A1 and BC are connected because there're splice junctions (brown color) between them. Exon BC and DE are connected because they each contain one side of read pair alignment (brown color) and there are no other exon regions between them. A known path (A1, BC, DE) is collected because the read pair alignment (brown color) involves these three nodes.

exonic region. We also merge two exon regions if they are moderately close to each other and have high read coverages. However, all merge procedures require two regions have no splice donors or acceptors between them, otherwise the genome gap between them should not occur together with the two regions in true transcripts.

In the connect step (Fig 3.3D), we establish weighted edges among nodes and collect known path information. Two nodes are connected by a directed edge if there is a read alignment crossing one node to the other. The two nodes can be continuous or discontinuous

on the genome when there's a splice junction between them. The related edge weight is proportional to the number of crossing read alignments. Edges can be augmented by using pair end reads (Fig 3.3D). If a read pair alignment has its first segment onto one node and the second segment onto another node, and there're no other nodes between them, these two nodes should be adjacent in some true transcript and hereby an edge should be added to connect them, even if there're no read alignments crossing these two nodes. Known paths (sequences of nodes inferred from read alignments) are collected when a read alignment crosses more than two nodes, or a read pair alignment for an augmented edge involves more than two nodes (also Fig 3.3D). We store known paths as a tuple of triple nodes. This not only provides sufficient extra information for flow decomposition later, but also helps reduce memory.

3.2.3 Sparse flow decomposition

Sparse flow decomposition has been proposed in [36] for de novo assembly and here we adapt it into the RefShannon framework for genome-guided assembly. Given a splice graph, it finds the minimum set of flows (e.g. paths with weights) that explains the splice graph.

Mathematically, given graph $G = (V, E)$, we need to find $\text{argmin}_T|T|$ such that $\forall v \in V$, $\forall e \in \text{InEdges}(v) \cup \text{OutEdges}(v)$, $w_e = \sum_{t \in T, e \in t} f(t)$ where $t \in T$ is a path in G corresponds to an RNA transcript and $f(t)$ represents the flow weight of t.

This is a hard task, because there can be $|P| = 2^{|V|}$ paths, and thus $2^{|P|}$ possible sets of flows. Instead, we try to decompose a set of flows out of the splice graph node by node in topological order, as illustrated in Fig 3.4. Specifically, we first add a special source node to connect all nodes without in-edges and a special sink node to connect all nodes without out-edges. We then do local sparse flow decomposition iteratively for each node. Finally, we collect flows that start from source node and stop at sink node as reconstructed RNA transcripts.

The local sparse flow decomposition at node v tries to find the minimum number of flows through v restricted by edge weight constraints. It can be formally described as: find

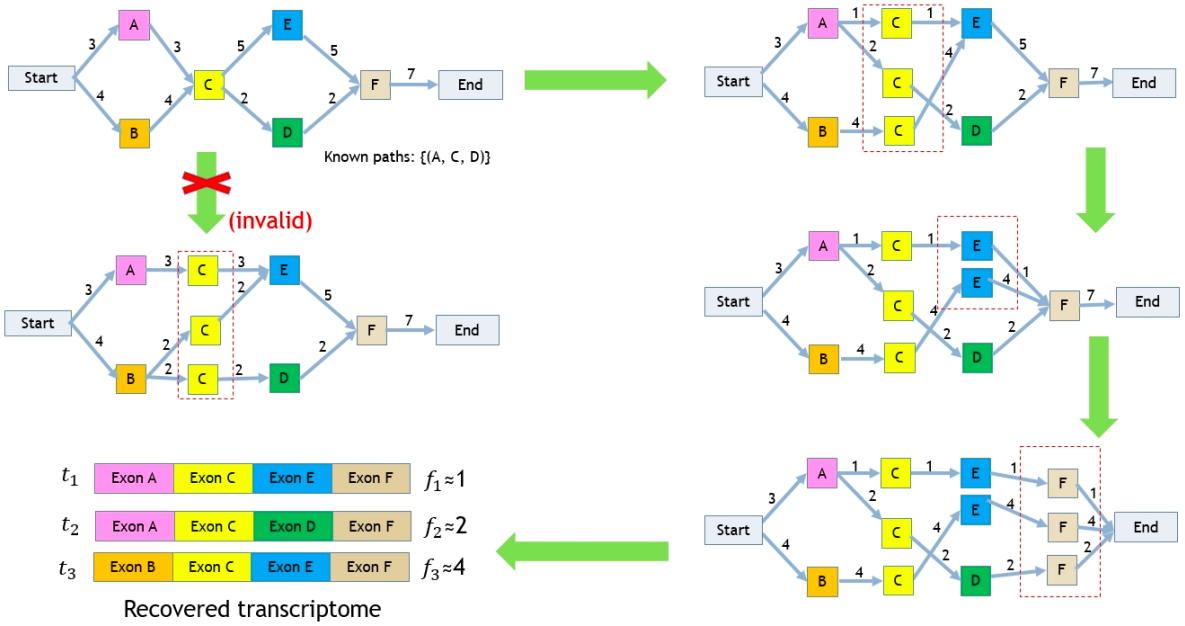


Figure 3.4: **Example of sparse flow decomposition.** In this figure, starting from the initial splice graph on the top left side, we try to recover transcriptome by doing local decomposition for node C,E and F iteratively.

$\text{argmin}_f ||f||_0$ such that $\sum_{i \in \text{InEdges}(v)} f_{i,j} = w_j, \forall j \in \text{OutEdges}(v)$ and $\sum_{j \in \text{OutEdges}(v)} f_{i,j} = w_i, \forall i \in \text{InEdges}(v)$ and $f_{i,j} \geq 0$. Here $||f||_0$ means the support set of $\{f_{i,j}\}$, or the number of positive $f_{i,j}$s.

Solving this problem may not be practically efficient. Consider for each node v, we have $m \times n$ possible flows ($m = |\text{InEdges}(v)|$, $n = |\text{OutEdges}(v)|$). We need to enumerate all 2^{mn} path combinations to figure out which combination offers us min $||f||_0$ and keeps the edge weight constraints.

One possible thought to ease the problem, as commonly used in signal processing, is to approximate $||f||_0$ by $||f||_1$, which unfortunately turns out to be a constant: $||f||_1 = \sum_{i,j} f_{i,j} = \sum_i w_i = \sum_j w_j$. Meanwhile, we have noticed that $||f||_0 = ||f \odot r||_0, r = \{r_{i,j} | r_{i,j} > 0\}$($\odot$ is element-wise product). So instead of approximating $||f||_0$ by $||f||_1$, we try to approxiate $||f||_0$ by $||f \odot r||_1 = \sum_{i,j} f_{i,j} r_{i,j}$.

Therefore, we have relaxed the original problem as: find $\text{argmin}_f \sum_{i,j} f_{i,j} r_{i,j}$ such that

$\sum_{i \in \text{InEdges}(v)} f_{i,j} = w_j, \forall j \in \text{OutEdges}(v)$ and $\sum_{j \in \text{OutEdges}(v)} f_{i,j} = w_i, \forall i \in \text{InEdges}(v)$ and $f_{i,j} \geq 0, r_{i,j} > 0$. We could use linear programming (e.g Python CVXOPT package (`http://cvxopt.org/`)) to solve the above problem.

Since the result may contain noise, we also need to do some thresholding to get a sparse solution.To get the sparsest result, we generate r by a number of times and select the sparsest f as the final local sparse flow decomposition solution.

It is also possible that the local sparse flow decomposition may bring two results that have the same lowest sparsity and satisfy the edge constraints. This can be resolved if one of them includes a known path, as illustrated in Fig 3.4 as well as Fig 3.1C.

3.3 Results

To demonstrate its superior performance, we have compared RefShannon to two widely used assemblers StringTie [27] and Cufflinks [61], which are recommended in previous benchmark work [66]. We have also compared RefShannon to guided Trinity [58] and recently published Ryuto[42], as they show relatively good performance among various assemblers in our initial analysis using smaller datasets. Our performance evaluation metrics include ROC (including sensitivity and false positive) for simulated datasets and sensitivity for real datasets. We will also discuss its computational complexity.

3.3.1 Datasets

Our performance evaluation is based on both simulated datasets and real datasets, as summarized in Table 3.1.

The real datasets include 132.05M Illumina single end reads (50-bp) sampled from human embryonic stem cells (HESC) (GSE51861 [73]) with 13274 reference transcripts, 115.36M Illuminar pair end reads (101-bp) sampled from Lymphoblastoid cells (LC) (SRP036136 [74]) with 207266 reference transcripts. We also use 183.53M Illuminar pair end reads (100-bp) sampled from HEK293T (Kidney) cells (SRX541227) previously produced and studied in StringTie [27]. The HESC reference transcripts are assembled by hybrid assembler IDP

from the 135M short reads together with 7.8M long PacBio reads [73]. The LC reference transcripts are based on GENCODE annotations augmented by utilizing a combination of short reads with long PacBio reads line (700K CCS reads) [74]. We also use LC reference transcripts for Kidney dataset.

	HESC	LC	Kidney	HESC-Sim	LC-Sim	Kidney-Sim
PE or SE	SE	PE	PE	SE	PE	PE
Read Length	50	101	100	50	101	100
Num of Reads (SE) or Read Pairs (PE)	132.05M	115.36M	183.53M	150M	150M	150M
Num of Reference Transcripts	13274	207266	207266	207266	207266	207266
Num of Reference Transcripts (Oracle Set)	2694	32394	15605	-	-	-
Access Number	GSE51861 [73]	SRP036136 [74]	SRX541227 [27]	-	-	-

Table 3.1: **Data Statistics**. There're three real datasets (HESC - human embryonic stem cells, LC - Lymphoblastoid cells, Kidney) and the corresponding three simulated datasets (HESC-Sim, LC-Sim, Kidney-Sim) used for evaluation. PE stands for pair-end reads and SE for single-end reads. Oracle Set refers to reference transcripts fully covered by reads.

The simulated datasets are generated based on the real ones. Specifically, we choose LC reference transcripts as the ground truth transcriptome for all simulated datasets. We then use RSEM [30], which is a popular transcriptome quantification tool, to learn parameters from each real dataset based on the alignments of real reads onto the reference transcripts. Based on the ground truth and learned parameters, we generate the relevant simulated reads.

3.3.2 Experiment procedure

The experiment procedure is as follows. For each dataset, we first use STAR aligner [67] to align reads onto reference genome (human genome hg19, downloaded from `http://hgdownload.cse.ucsc.edu/goldenpath/hg19/bigZips/`) that contains multiple chromosomes. We choose STAR aligner since it had better performance for most of the assemblers than using alternative aligners such as Tophat2 [68] and Hisat2 [69], in our initial evaluation using smaller datasets; in addition different aligners did not affect the comparison conclusions.

We then apply assemblers onto read alignments to reconstruct transcripts for all chromosomes. The assemblers which we have selected to compare RefShannon to include Cufflinks (v2.2.1), StringTie (v1.3.4d), Ryuto (v1.3m) and Trinity (v2.9.1) as they show relatively good performance in our initial analysis of various assemblers using smaller datasets.

Finally, we evaluate the reconstructed transcripts and compare them with relevant reference transcripts to see the performance of ROC (including sensitivity and false positive) for simulated datasets and sensitivity for real datasets.

3.3.3 ROC of simulated datasets

For simulated datasets (HESC-Sim, LC-Sim, Kidney-Sim in Table 3.1), since we know the ground truth reference transcripts, we check the performance of receiver operating characteristic curves (ROC), which includes sensitivity as well as false positive.

The metric of sensitivity describes how many reference transcripts have been correctly reconstructed. To evaluate sensitivity, we first use blat (`https://genome.ucsc.edu/goldenpath/help/blatSpec.html`) to create a mapping between reconstructed transcripts T_{rec} and reference transcripts T_{ref}. We associate each reference transcript $t \in T_{ref}$ with a reconstructed transcript $r \in T_{rec}$ that mostly matches t. We then consider each $t \in T_{ref}$ is correctly recovered if it is over 90% matched with its associated $r \in T_{rec}$.

The metric of false positive describes how many transcripts are falsely reconstructed and do not belong to the true reference transcripts. Based on the blat result, we consider a reconstructed transcript $r \in T_{rec}$ to be a false positive if it is below 90% matched with any reference transcript $t \in T_{ref}$. Note if $r \in T_{rec}$ is contained in a reference transcript $t \in T_{ref}$, it is not considered to be a false positive.

In addition to the threshold of 90%, we have also tried other thresholds, and it does not affect our comparison conclusions.

To have a comprehensive understanding of the ROC performance, we run StringTie and Cufflinks in both default mode and max sensitivity mode; we run Ryuto and guided Trinity in their default modes after tuning several parameters which don't affect sensitivity much.

We also tune RefShannon so that it is able to adaptively adjust its splice graph generation as well as sparse flow decomposition output in order to trade off its sensitivity and false positive performance.

Fig 3.5 illustrates the ROC performance. Overall RefShannon shows higher sensitivity at a given false positive ratio than other assemblers. For example in simulated LC dataset, the max sensitivity point of RefShannon (the top right blue point) has higher sensitivity and lower false positive than StringTie in max sensitivity setting and the min false positive point of RefShannon (the lower left blue point) has higher sensitivity and lower false positive than Cufflinks in default setting. Quantitatively, if we fix false positive ratio at 15.5%, we obtain a sensitivity gain of 13.5% over guided Trinity which has the second best ROC point. The conclusion is similar in the Kidney dataset, where RefShannon obtains a sensitivity gain of 22% over guided Trinity at 19% false positive ratio. In the HESC dataset, RefShannon shows an ROC trend similar to guided Trinity and has a sensitivity gain of 10.6% over Cufflinks which shows the next best ROC point at 12% false positive ratio. There is a larger gain in LC and Kidney datasets of pair-end reads than in HESC dataset of single-end reads probably because RefShannon is able to better utilize the pair-end information not only in the splice graph construction but also in the flow decomposition stage.

3.3.4 Sensitivity of real datasets

For real datasets, since it is hard to judge if a reconstructed transcript is a false positive or an unknown transcript yet to be discovered, we thus focus only on sensitivity performance which means among the known reference transcripts, how many of them are correctly recovered.

The sensitivity calculation is similar to previous ROC approach. The difference is (1) For fair evaluation of sensitivity, we run assemblers all in their max sensitivity settings. (2) We only use a subset of reference transcripts as an "oracle set" (statistics in Table 3.1) for each real dataset. The oracle set contains reference transcripts that are fully covered by reads (tolerated by 25 bp from both ends). We consider these as expressed reference transcripts and expect them to be reconstructed by assemblers.

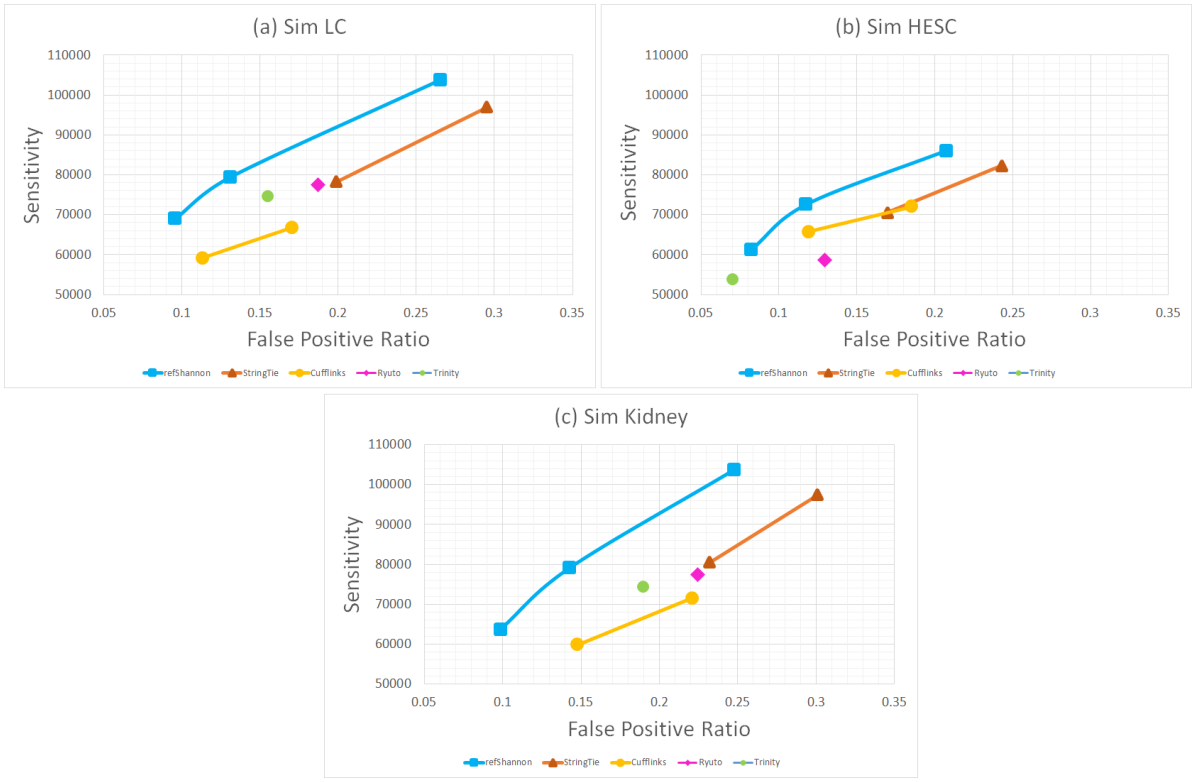


Figure 3.5: **ROC analysis of simulated datasets.**

Fig 3.6 shows our sensitivity evaluation for the three real datasets (LC, HESC, Kidney). In the three subplots along the first column, reference transcripts in the oracle set are grouped according to their read coverage. If a reference transcript has a low (or high) read coverage, it implies its expression level in cells is low (or high). Note the oracle sets of reference transcripts are different among three datasets, so the group values per dataset are different. For LC and Kidney datasets, most of the reference transcripts are within lower coverage. Therefore, we group the reference transcripts into read coverage of 60, 10, 10, 10 and 10 percentile. For HESC, the reference transcripts are grouped into 20 percentiles each. For each dataset under various read coverage conditions, RefShannon has recovered reference transcripts better than all other assemblers. In the three subplots of the second column, reference transcripts are grouped according to their isoform multiplicity. Recall that a gene

Figure 3.6: **Sensitivity analysis of real datasets.** The y-axis of each subplot is recovery ratio, the ratio of the number of correctly reconstructed reference transcripts over the total number of (oracle) reference transcripts. The x-axis of each subplot in left column is read coverage, and the x-axis of each subplot in right column is isoform multiplicity. A lower read coverage implies less expression level of reference transcript in cells, and a higher isoform multiplicity implies more complex splicing patterns. Isoform multiplicity in HESC is lower than that of LC data, implying a simpler splicing structure of reference transcripts in HESC data.

may contain multiple transcripts (i.e. isoforms) due to alternative splicing, and the isoform multiplicity of a transcript refers to the number of isoforms of that transcript's gene. A higher isoform multiplicity value implies a more complex splice graph for recovery and also implies longer transcript length. Here the reference transcripts for HESC have relatively simple isoform multiplicity (most equals to 1), so we group reference transcripts into about 70, 15 and 15 percentile, while the reference transcripts for LC and Kidney datasets are grouped into 20 percentiles each. For each dataset under various isoform multiplicity regions, RefShannon

has also recovered more reference transcripts than all other assemblers. Note that Cufflinks's performance drops in LC and Kidney compared to HESC, this could be because reads from LC and Kidney datasets are pair end and there are more read alignments Cufflinks may consider to be compatibility uncertain and thus throw away. Note also that guided Trinity does a good job to recover the transcripts of high expression levels in HESC dataset, this could be because the relevant underlying assembly graphs are less fragmented.

3.3.5 Computation resources

We have run RefShannon, StringTie, Cufflinks, Ryuto and Trinity by their default mode on the three real datasets to measure their computational resources.

The experiments were conducted using 20 CPUs (AMD Opteron 6380, 1400MHz) of a lab server with 515G total memory. The memory consumption (the high-water RSS plus CACHE memory usage of involved threads[2]) and time consumption (the total wall clock time) are captured using cgmemtime (`https://github.com/gsauthof/cgmemtime`), which was previously adopted to compare the computational complexities for read aligners [75].

Datasets	RefShannon	StringTie	Cufflinks	Ryuto	Trinity
HESC (132M SE reads)	34.5 min	**4.3 min**	58.6 min	10.32 min	12.1 hrs
LC (115M PE reads)	5.19 hrs	**21.7 min**	8.27 hrs	5.26 hrs	30.8 hrs
Kidney (183M PE reads)	9.23 hrs	**38.37 min**	43.77 hrs	16.69 hrs	41.7 hrs

Table 3.2: **Time Consumption.** Best performance is in bold-font.

Table 3.2 and 3.3 show the time and memory results respectively. In terms of time, StringTie is the fastest whereas RefShannon is overall faster than Cufflinks, guided Trinity and Ryuto. In terms of memory consumption, Stringtie and Cufflinks work better especially in the largest Kidney dataset. RefShannon's memory consumption is high, this could be

[2]For RefShannon, we apply multiprocess instead of multithreads.

Datasets	RefShannon	StringTie	Cufflinks	Ryuto	Trinity
HESC (132M SE reads)	59.69	0.98	7	**0.65**	100.26
LC (115M PE reads)	106.2	15	13.18	**3.56**	187.43
Kidney (183M PE reads)	166	**27.36**	30.99	70.50	281.48

Table 3.3: **Memory Consumption (GB).** Best performance is in bold-font.

because RefShannon is written in Python and memory sharing is less efficient especially for multiprocessing. Guided Trinity takes more time and memory, which is reasonable as it is essentially doing de novo assembly but in smaller regions.

Currently, a typical lab server with at least 20 CPU cores and over 200GB memory would be sufficient to run RefShannon on large real datasets. One of our future direction is to further improve its computational efficiency.

3.4 Conclusion

We have developed RefShannon - a new genome-guided transcriptome assembler as an extension of the original de novo Shannon project[36]. It utilizes a careful splice graph generation procedure aimed at capturing as much information as possible from read alignments, and utilizes a sparse flow decomposition algorithm aims at reconstructing as small number of transcripts as possible under splice graph constraints. Our evaluation shows performance gain of RefShannon over state-of-art genome guided assemblers for both simulated and real datasets. We expect RefShannon will help discover novel genes and isoforms that may be missed by existing transcriptome assemblers. We also expect its intermediately generated splice graphs (with nodes, edges and paths) will provide helpful interface to be used by other relevant research.

There are several future directions. One of them is to further improve the computational efficiency as described previously.

In addition, it shall be useful to incorporate the third generation long reads (e.g. PacBio

[19] and Nanopore reads [20]) to assist the short read assembly process. Compared to the short RNA reads (e.g. < 300 bp), long reads (e.g. > 10000 bp) will be able to better bridge multi-exons and resolve repetitive regions. However, they are more expensive (e.g. 100 to 280 Euro per isolate) and have higher errors (typically 10% to 15%), whereas short reads are more cost-effective (e.g. 40 Euro per isolate) and also much more accurate (typically below 0.1% error rate) [21, 22]. Ideally, applying long contigs (or reads) into short read assemblers could be an effective feature [27, 42].

Moreover, it could be interesting to apply RefShannon onto the downstream analysis such as variant calling (e.g. SNP or small indels). RefShannon shows a better sensitivity (under a similar false positive rate) than existing assemblers, which implies additional new RNA transcripts could be discovered. Using variant callers (e.g. GATK [76] or our recently developed abSNP [37]) to look for variants from newly discovered transcripts may help scientists gain new medical insights.

Chapter 4

ABSNP: RNA-SEQ SNP CALLING USING ABUNDANCE ESTIMATION

4.1 Introduction

Decoding individual-specific (or even tissue or cell-specific) variations with respect to a reference genome is an important task, downstream of DNA sequencing. Among the variations that can be detected with high throughput sequencing data, the most frequently called variants are single nucleotide polymorphisms (SNPs), which specify single base loci at which the target sequence differs from the reference allele. A reliable detection of them is an important task because they are relevant in predicting organismal traits as well as implicated in several diseases.

Existing SNP calling software (i.e. DNA-seq SNP callers), such as GATK [28], GlfMultiples [29], SAMtools mpileup [30], FreeBayes [31] and VarScan [32], mainly rely on whole genome sequencing (WGS) or whole exome sequencing (WES). While WGS can call SNPs throughout the entire genome, many downstream pipelines only consider SNPs in gene exon regions, as their impact is easier to quantify. Therefore, WES is a widely used cheaper alternative, which focuses on DNA reads from the exonic regions. A third strategy is to utilize RNA-seq reads from a tissue of interest and use those reads both for (a) expression estimation as well as (b) variant calling. Beside being fast and inexpensive [77], this third strategy is advantageous when the genes harboring SNPs of interest are likely to have non-negligible expression, such as in cancer tissue analysis. However existing DNA-seq SNP callers are not

This is joint work with Professor Soheil Mohajer from Electrical and Computer Engineering Department, University of Minnesota.

suitable to properly handle RNA-seq data directly. One reason is that RNA-seq reads may be sampled from two different exons, and these splice junctions are typically not captured by these callers. In addition, RNA-seq data also has a specific feature, namely the varying expression of different genes, with expression levels varying over several orders of magnitude.

To address the above-mentioned challenges, designing SNP callers tailored for RNA-seq data (i.e. RNA-seq SNP callers) is necessary. There are only a limited number of works on RNA-seq SNP calling, such as eSNV-detect (2014) [33], SNPiR (2013) [34], GATK (2011) [28] and SNVMix (2010) [35]. eSNV-detect and SNPiR essentially rely on SAMtools mpileup and GATK to call SNPs respectively, and SNVMix depends on SAMtools mpileup to prepare necessary statistics for SNP calling. Among these, only GATK is still under constant maintenance and development.

Even though most existing SNP callers (especially GATK) offer excellent performance on benchmark sets, these sets are usually only representative of regions in the genome without repeats. On the repetitive regions, where reads are not uniquely mapped, most callers are unable to make any reliable calls, since they simply discard all of the multiply mapped reads and consequently the corresponding SNP information will be missed. We note that even projects that are designed to catalog the performance of SNP callers, such as Genome in a bottle project [78], consider high-quality calls only in non-repetitive regions. This limitation fundamentally comes from the fact that existing read aligners are unable to differentiate between multiply mapped reads, and therefore cannot make any predictions on the origin of the SNP with confidence.

There are some studies regarding DNA-seq SNP calling that consider multiply mapped reads, such as Sniper [79], SiRen [80] and GW-CALL [81], but to the best of our knowledge there is no work yet on RNA-seq SNP Calling that addresses the problem of multiply mapped reads on repetitive genomic regions.

To fill this gap, we've designed abSNP [1], which is a novel RNA-seq SNP caller that is able

[1] "a" stands for abundance and "b" stands for Bayesian principles. abSNP is written in Python and is freely available at `https://github.com/shunfumao/abSNP`.

to call SNPs even in repetitive regions. The key idea, is to use the products of abundance estimation (also known as quantification), which include estimated gene expression levels as well as more accurate read mapping quality scores. As far as we know, such kind of information has not been exploited yet for RNA-seq SNP calling.

We demonstrate that utilizing such information leads to significant gains in SNP calling performance. In comparison to existing callers that are unable to make any calls in multiply mapped regions, abSNP is able to get significantly improved sensitivity. In particular, in SNPs that have high coverage, abSNP demonstrates high accuracy, making it a viable alternative to existing SNP callers.

The remained chapter is organized as follows. We first describe the abSNP methods. We then show its superior SNP detection performance using simulated datasets. Lastly, we will conclude our work and discuss future directions.

4.2 Methods

4.2.1 Problem Statement

AbSNP calls SNPs in *diploid* genome based on RNA-seq data. The input is a set of RNA-seq reads sampled from the transcriptome.The goal is to identify SNPs located within the gene regions of the target individual, i.e., the loci at which the target genome is different from a known reference genome. To this end, we use the standard technique of read alignment of the sampled reads onto the reference sequence and compare the nucleotides on the mapped reads to those of the reference genome to call SNPs.

There are several challenges that need to be addressed: (i) The most important factor is due to existence of repetitive regions in the target/reference; reads sampled from repetitive regions get mapped to multiple loci, and the algorithm has to figure out where they are sampled from. (ii) Not all the reads sampled from a locus carry SNP information. This is due to the heterozygous SNPs, in which one of the alleles contain a SNP, and the other one matches with the reference genome. (iii) A unique feature of RNA-seq data is that there is

potentially a wide gap between the number of reads sampled from the paternal and maternal alleles, due to the varying expression levels of the corresponding genes.

4.2.2 Assumptions

To handle the above-mentioned challenges, we develop abSNP based on the following three key assumptions in order to simplify our modeling of the problem: (i) *Single SNP across repetitive regions*: When there are repetitive regions, we assume that at most one copy has a SNP at a given locus. This assumption is valid since the probability of SNP is small ($p \approx 0.001$) and the probability of two SNPs p^2 is negligible. We note that each copy of a repetitive region can have many SNPs; just that they do not occur at the same base locus. (ii) *Heterozygous SNPs*: We assume that SNP only appears in one of the paternal or the maternal allele, while the other allele is consistent with the reference genome. Our methods can also detect SNP occurring in both alleles (i.e. homozygous SNP), but further method is needed to distinguish whether a SNP occurs in one or both of the alleles. (iii) *Equal allele contribution*: This means each paternal and maternal allele contributes equally to the abundance at each genomic locus[2].

4.2.3 Definitions

We briefly review a set of terms and notations (as in Figure 4.1) that are useful for presentation of the algorithm and the following discussions.

We denote the base of the reference genome (*reference base*) at the current locus i (*target locus*) by $r \in \{A, C, G, T\}$, and the estimated expression level as λ. The locus i is called a SNP if among the two alleles of the individual sample, one allele (*non-SNP allele*) has base r, and the other allele (*target allele*) has $x \neq r$ (recall the heterozygous SNP assumption).

Let $\mathcal{R} = \{R_1, R_2, \ldots, R_J\}$ represent the J sampled reads mapped onto the reference genome so that they cover locus i. For each R_j, we denote its base covering the locus i as

[2]This assumption is used to develop our algorithm, however, this assumption is not critical. In our actual evaluation each allele has a randomly assigned (thus different) expression levels.

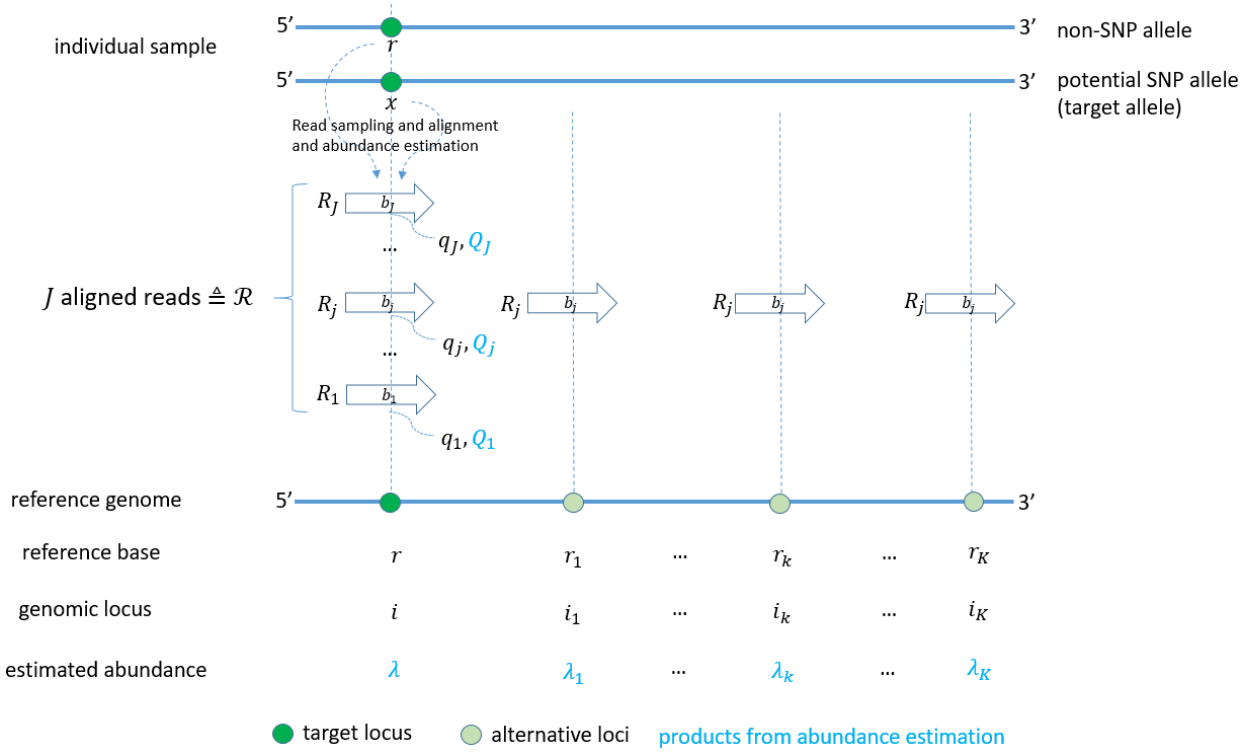


Figure 4.1: A typical scenario of SNP calling at locus i

b_j, and its base quality[3] at locus i by q_j. The *read mapping quality* ($MAPQ$) of R_j at i is denoted as Q_j. The read R_j is called a *SNP read* at locus i if $b_j \neq r$. Depending on whether a read R_j is mapped to several loci or not, it is also called a *multiply mapped read* or a *uniquely mapped read* (*unique read* for short) respectively. If a read is multiply mapped to K loci, then it has *multimapping degree* K.

In addition, we define *repetitive regions* as the loci connected by multiply mapped reads. For example, in Figure 4.1 R_j is mapped to loci $\{i, i_1, ..., i_K\}$, so they form repetitive regions, and we call $\{i_1, ..., i_K\}$ the *alternative loci* of locus i. In contrast, *unique region* refers to the loci having no multiply mapped reads.

[3]Note the base quality is encoded in read file, and is different from mapping quality of a read, which is encoded in the read alignment file.

4.2.4 Overall Workflow

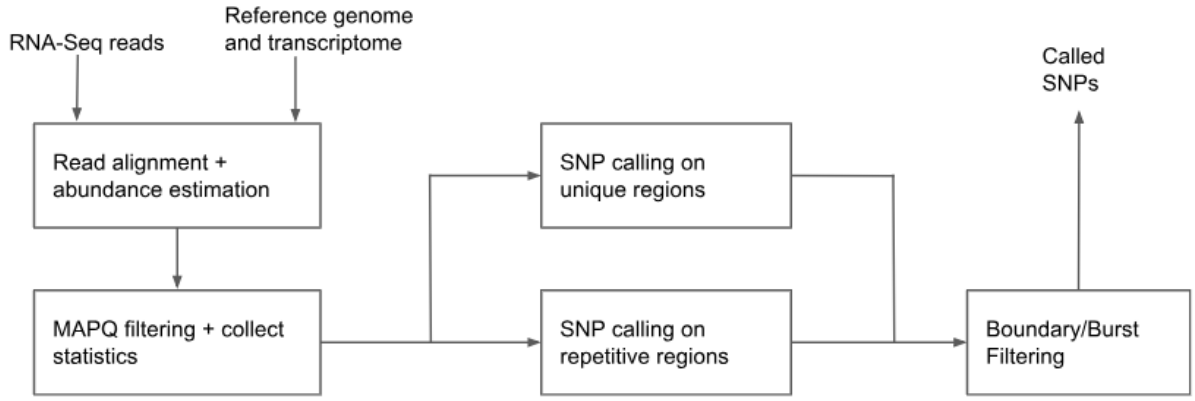


Figure 4.2: Overall Flow of abSNP

The overall workflow is illutrated in Figure 4.2. AbSNP takes input of raw reads, reference genome and transcriptome annotations. It utilizes STAR [67] to align reads onto the transcriptome and utilizes RSEM [82] to do abundance estimation. Based on RSEM outputs, we obtain expression values per genome locus as well as the genome-based read alignments which contain rich MAPQ information. Although it is also possible to use pure alignment software (e.g. STAR [67] or TopHat2 [68]) to obtain genome-based read alignments directly, there will be no expression value and rich MAPQ information (e.g. λ and Q highlighted in Figure 4.1) which are useful signals needed by abSNP.

The alignments of low MAPQ are filtered (Section 4.2.5) and necessary statistics are collected. AbSNP then calls SNPs in both unique and repetitive regions (Section 4.2.6). Lastly, abSNP filters both the SNPs slightly outside the splice boundaries and the SNP bursts[4] in order to reduce false positives (Section 4.2.7).

4.2.5 MAPQ Filtering

MAPQ (MAPping Quality of read alignments) is a metric used to capture the confidence about mapping of a read to a reference region. As described in [83] as well as in official read

[4]SNP burst refers to a group of called SNPs that are very close to each other.

alignment format [84], it is defined as: $-10\log_{10}(1 - P(\text{correct mapping}))$. Since a read can be multiply mapped onto different loci (i.e. repetitive regions), a better knowledge of MAPQ for each alignment can potentially help us remove false alignments and consequently achieve a better SNP calling performance.

Although well defined, the MAPQ scores reported by existing RNA-seq read aligners (such as STAR and TopHat2) are usually uninformative and have the same value for all of the multiply mapped reads. For example, in STAR (also similar in TopHat2), a uniquely mapped read will have MAPQ = 255 and a read multiply mapped onto N_{map} loci will have MAPQ $= -10\log_{10}(1 - \frac{1}{N_{map}})$ corresponding to $P(\text{correct mapping}) = \frac{1}{N_{map}}$. If the read maps equally well to all possible loci, there is not enough information to determine its true locus.

However, if we take another view from abundance estimation, it is possible to get additional information. To estimate transcripts' abundances, an Expectation-Maximization (EM) algorithm is typically involved, which alternates between the two steps: (1) given the read alignments onto RNA transcripts, the abundance of transcripts is estimated; (2) given the abundance of transcripts, the read alignment probabilities are refined. This iterative procedure calculates the probability that a given read is assigned to a particular genomic locus and therefore can be used as a sharper estimate of MAPQ. Here we use RSEM [82], a software extensively used for abundance estimation, to provide us with refined MAPQ scores[5].

We then filter out some of the low quality read alignments with MAPQ scores lower than certain threshold (e.g. 0.1) via the MAPQ filtering process (Figure 4.2). We empirically choose this threshold, since we find this helps effectively removing false alignments of multiply mapped reads that may cause false positives.

[5]It is also possible to replace RSEM with other abundance estimators such as eXpress [19].

4.2.6 SNP Calling Algorithm

AbSNP separates the strategies to call SNPs for unique and repetitive regions. For a unique locus i, abSNP calculates the probability scores of the target allele base being $\{A, C, G, T\}$. These scores are used directly to identify if a SNP exists at locus i. For repetitive regions, there could be a multiply mapped read propagates SNP signal to different loci, introducing false positives if these loci are called as SNPs because of the SNP signal. To avoid false positives, these loci will be called jointly according to their probability scores and other statistics.

Probability Score

We apply maximum a-posterior probability (MAP) to estimate $S_i(x)$, the probability score of the target allele base at locus i being $x \in \{A, C, G, T\}$, as[6]:

$$S_i(X = x|\mathcal{R}) = \frac{P(\mathcal{R}|X = x)P(X = x)}{P(\mathcal{R})} \tag{4.1}$$

where $\mathcal{R} = \{R_1, \ldots, R_J\}$ is the set of reads mapped over locus i of the reference genome, and X is a random variable, which represents the possible target allele base $x \in \{A, C, G, T\}$ at locus i.

Here $P(X = x)$ can be further expressed as:

$$P(X = x) = \begin{cases} \frac{P_{\mathsf{SNP}}}{3} & \text{if } x \in \{A, C, G, T\} \setminus \{r\} \\ 1 - P_{\mathsf{SNP}} & \text{if } x = r \end{cases} \tag{4.2}$$

where P_{SNP} indicates the prior probability (i.e. general knowledge) for a SNP to occur per genomic locus[7].

---

[6]The notations are illustrated in Figure 4.1 and the dependency of variables on i is eliminated, whenever it is clear from the context.

[7]P_{SNP} can be set based on the knowledge of SNP rate of the genome of interest. Suppose there are around 10 million SNPs across human genome of 3 billion bases, then we could set P_{SNP} as $\frac{10^7}{3\times10^9} \approx 3 \times 10^{-3}$.

To decompose $P(\mathcal{R}|X = x)$. A common approach is to assume reads are independent from each other (as used in [83]), so we have:

$$P(\mathcal{R}|X = x) = \Pi_{j=1}^{J} P_j = \Pi_{j=1}^{J} P(R_j = b_j|X = x, r, q_j, Q_j) \tag{4.3}$$

Here P_j indicates the probability of the j-th read (i.e. R_j) having base b_j at locus i, given all the other assumptions, including base x at the target allele, the base r in the reference, the read base quality score q_j, and the mapping quality score Q_j of read R_j. We can further expand P_j as:

$$P(R_j = b_j|X = x, r, q_j, Q_j) = Q_j \times \begin{cases} q_j & \text{if } b_j = x = r \\ \frac{q_j}{3} + \frac{1}{6} & \text{if } b_j = x \neq r \\ \frac{q_j}{3} + \frac{1}{6} & \text{if } b_j = r \neq x \\ \frac{1-q_j}{3} & \text{if } b_j \notin \{x, r\} \end{cases} \tag{4.4}$$

To understand Equation (4.4), let's first consider a uniquely mapped read R_j (i.e. $Q_j = 1$)[8]. For $b_j = x = r$, P_j is the probability that read R_j is sampled from target individual's paternal or maternal allele at locus i (which is 1) and no error has occurred (which happens with probability q_j). Therefore, we have $P_j = 1 \times q_j = q_j$. If $b_j = x \neq r$, there are two possibilities for observing R_j: either P_j is the probability of sampling R_j from the SNP allele at locus i (which is $\frac{1}{2}$, due to the assumption of equal allele contribution) without error (which is q_j), or P_j is the probability of sampling R_j from the non-SNP allele (which is $\frac{1}{2}$) with error (which is $\frac{1-q_j}{3}$). Therefore, $P_j = \frac{1}{2}q_j + \frac{1}{2}\frac{1-q_j}{3} = \frac{q_j}{3} + \frac{1}{6}$. Similar reasoning applies to the remaining cases. For a multiply mapped read R_j, the MAPQ factor Q_j at locus i (representing the confidence of R_j belonging to locus i) will be small if R_j is more likely to align to an alternative locus and consequently penalize P_j at locus i .

[8]It's possible that $Q_j \in (0.9, 1)$ if a read is multiply mapped but becomes uniquely mapped after MAPQ filtering. In this case, we still use $Q_j = 1$.

SNP Calling on Unique Regions

At *unique* regions (assume it is locus i in Figure 4.1), we need to estimate the target allele base as: $\hat{x} = \arg\max_{x \in \{A,C,G,T\}} S_i(x) = \arg\max_{x \in \{A,C,G,T\}} P(\mathcal{R}|X = x)P(X = x)$. The second equation holds since $P(\mathcal{R})$ is a constant. Therefore, $\hat{x}$ can be estimated by using Equation (4.2) to (4.4), and we will call a SNP at locus i if $\hat{x} \neq r$.

SNP Calling on Repetitive Regions

On repetitive regions (assume they are $I = \{i_1, ..., i_K\}$ in Figure 4.1), multiple SNPs could be called. Based on the assumption of single SNP across repetitive regions (Section 4.2.2), there should be one true SNP among them whereas the others are false positives[9]. Indeed, if a read sampled from a true SNP locus is multiply mapped to its alternative loci, it will falsely propagate its SNP information to these loci, causing false positives. Therefore, we need to call SNP jointly on the repetitive regions, and keep only the most likely SNP across the repetitive regions if it exists.

To formulate, let us consider the repetitive regions (assume they are $I = \{i_1, ...i_k, ..., i_K\}$ in Figure 4.1) where SNPs are called (the estimated target allele base is different from the reference base, i.e. $x_j \neq r_j$ for $j \in I$). Our goal is to choose one (or none) of these SNPs. We apply several heuristic rules in this selection procedure. For example, if the expression levels across repetitive regions are very similar ($\frac{|\lambda_{j_1} - \lambda_{j_2}|}{\lambda_{j_2}} < 0.5\%, \forall j_1, j_2 \in I$), we will pick none of these SNPs. We will also skip the SNPs where the expression levels are low ($\lambda_j < 5, j \in I$) or the average read multimapping degree (defined in Section 4.2.3) of the repetitive regions is big ($>= 3$). We prefer to select the locus supported by more unique SNP reads or larger probability score(e.g. $j^* = \arg\max_{j \in I} S_j(x_j)$).

[9]In our simulation, there are only 1 to 2 cases where more than 1 true SNPs occur in repetitive regions. All other cases follow the assumption of single SNP across repetitive regions.

4.2.7 Boundary and Burst Filtering

In addition to MAPQ filtering (Section 4.2.5) and several filtering heuristics used on repetitive regions (Section 4.2.6), we design further filtering rules, such as *boundary and burst filtering*, to reduce false positives caused by read alignment errors.

Specifically, we find many false positives occur near splice boundaries[10]. We filter the SNPs called within certain window size (e.g. 10-bp) either after a splice donor or before a splice acceptor. These are mainly caused by read alignment errors. We also find most SNP bursts (SNPs called in continuous loci) tend to be false positives. We filter them as well.

4.3 Results

In this section, we perform simulation studies to compare the RNA-seq SNP calling performance of abSNP with that of GATK, which is widely used and has a best-practice guideline for RNA-seq SNP calling. We resort to simulation studies because it is difficult to obtain real data with ground truth for SNPs that have multiply mapped reads, as existing methods are unable to call these loci reliably. We demonstrate that while GATK is unable to make any calls on the loci having only multiply mapped reads, abSNP can call SNPs with significant accuracy.

4.3.1 Simulation Setup

To evaluate performance, we have developed an RNA-seq SNP simulator. The simulator takes as input a reference genome, a transcriptome annotation, the requested number of SNPs, and the read requirements (e.g. number, length, error rate, single-end or pair-end). It assigns each transcript a random expression level according to a log-normal distribution. We explicitly account for the effect of allele-specific expression with maternal and paternal transcripts having different expression levels (in our case, we simulate these expression levels

[10]If a read is split aligned onto the genome region [a,b] and [c,d], we call b and c are splice boundaries, and b is the splice donor and c is the splice acceptor.

to be independent of each other). The requested number of SNPs are generated randomly in the gene regions.

We then generate reads independently from the paternal and maternal transcriptomes that contain SNPs using the UC Riverside RNA-seq simulator [85], with error rate set at 1% (to mimic Illumina error rates). We then pool the reads from the two alleles in order to generate the dataset of RNA-seq reads.

We prepared 2 datasets each with $100M$ 100-bp single-end or pair-end reads. We choose GRCh37 [86] as the reference genome and the relevant UCSC gene annotations [87] as the transcriptome, and generate 0.1% SNPs for each dataset. To compare performance, we run both abSNP and GATK by taking simulated reads as input and obtaining SNP candidates as output. For abSNP, the process is described in Section 4.2.4. For GATK (version 3.4-46), we apply its best practice [88] and incorporate the annotated transcriptome to improve its read alignment.

4.3.2 Performance

We first utilize *Venn Diagram Analysis* to get a performance overview of abSNP and GATK on the pair-end simulated datasets in Figure 4.3. The single-end datasets show similar results.

AbSNP shows an overall better performance than GATK (Figure 4.3a). Specifically, among the loci where both abSNP and GATK have SNP reads[11], abSNP has correctly detected $40,217$ out of $60,917$ true SNPs, with 152 false positives while GATK has correctly detected $39,265$ out of $60,917$ true SNPs, with 16 false positives. We notice that although the loci all have SNP signals for both abSNP and GATK, there are $19,631$ true SNPs still undetected, this could be because most of these loci have only weak SNP signals (e.g. 1 or 2 SNP reads). Compared to GATK, abSNP has detected 952 more true SNPs at a relatively

[11]Note abSNP and GATK utilize different read alignments. AbSNP aligns reads onto transcriptome, filter the alignments of low MAPQ, and then convert the transcriptome-based alignments to the genome-based alignments. GATK utilizes genome-based alignments directly.

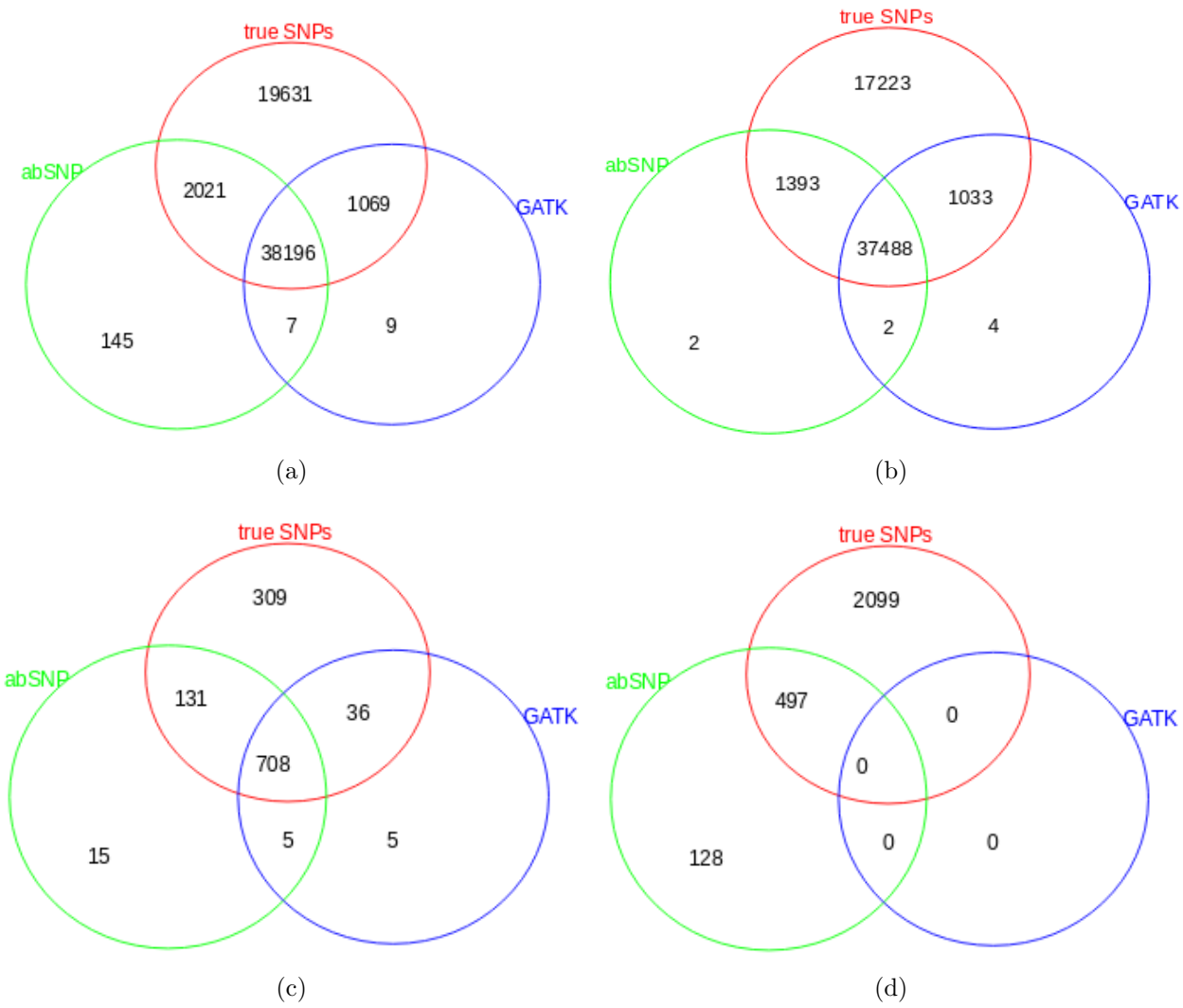


Figure 4.3: **Venn Diagram Analysis.** (a) The loci where both abSNP and GATK have SNP reads. (a) is further decomposed into (b-d). (b) The loci where abSNP has SNP reads while GATK has only unique reads. (c) The loci where abSNP has SNP reads while GATK has both unique and multiply mapped reads. (d) The loci where abSNP has SNP reads while GATK has only multiply mapped reads.

minor cost of 136 more false positives.

The sensitivity gain of abSNP over GATK comes from both the regions where GATK is able to capture SNP signals to make calls and the regions where GATK is unable to make

any calls. To understand this, the loci in Figure 4.3a are further categorized into Figure 4.3b where GATK has only unique reads, Figure 4.3c where GATK has both unique and multiply mapped reads, and Figure 4.3d where GATK has only multiply mapped reads.

For the loci where GATK has only unique reads, the SNP signals all come from unique reads and GATK is able to capture them and make SNP calls. In these loci where GATK is traditionally good at, abSNP has detected 360 (37.8% of abSNP's overall sensitivity gain) more true SNPs with comparable (or even less) false positives (Figure 4.3b).

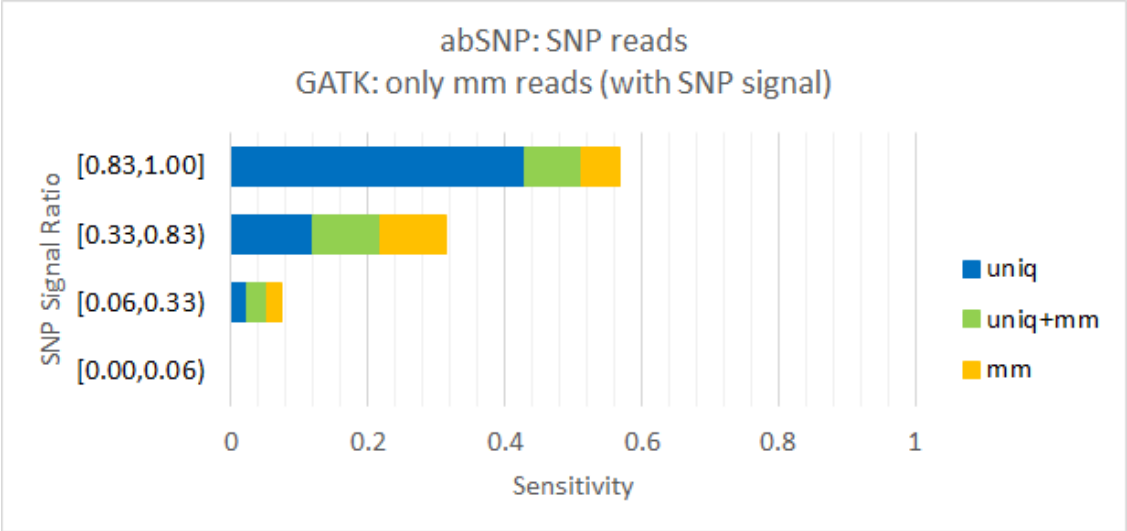
For the loci where GATK has both unique and multiply mapped reads, the SNP signals may come from unique reads and therefore GATK is still able to capture them and make some calls. In these loci, abSNP has detected 95 (10% of abSNP's overall sensitivity gain) more true SNPs with slightly more false positives (Figure 4.3c).

Together for these loci (Figure 4.3b and 4.3c), even if GATK is able to capture SNP signals and call SNPs, abSNP has achieved better sensitivity (47.8% of its overall sensitivity gain) with comparable false positives.

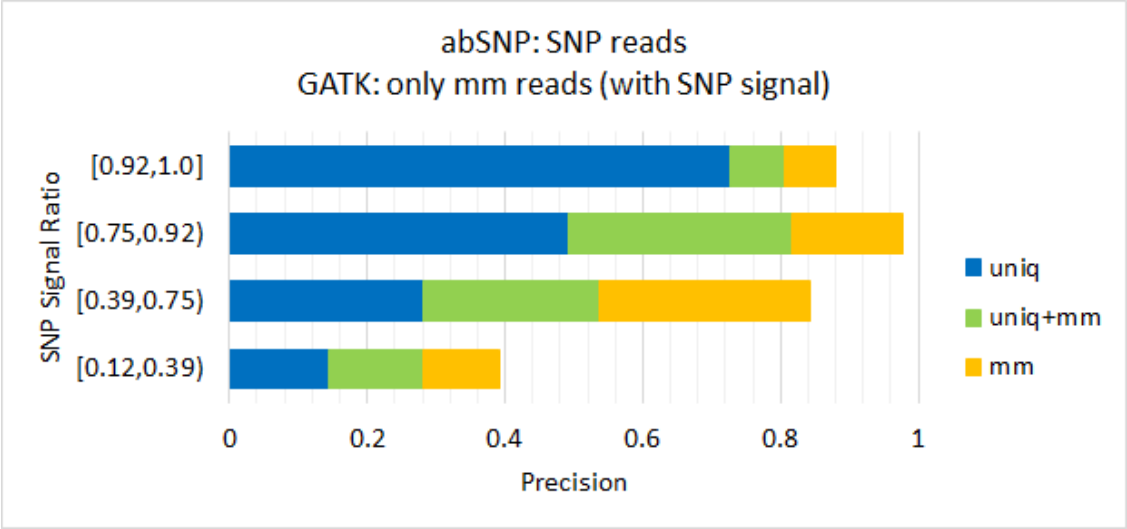
For the loci where GATK has only multiply mapped reads (Figure 4.3d), GATK will fail to make any calls since it will throw all multiply mapped reads away. In contrast, abSNP is still able to detect 497 true SNPs (52.2% of its overall sensitivity gain) at a comparatively small cost of 128 false positives (84.2% of its overall false positive loss).

While on average abSNP achieves 19.1% sensitivity and 79.5% precision on loci where GATK fails to make calls, we show that both sensitivity and precision improve as the SNP signal ratio (the number of SNP reads over the total number of reads at a locus) increases, as illustrated in Figure 4.4.

Specifically, in Figure 4.4a, true SNP loci are grouped by the SNP signal ratio, and the sensitivity of abSNP reaches up to 56%. In Figure 4.4b, the SNPs called by abSNP are grouped by the SNP signal ratio, and the precision of abSNP reaches up to 97% at SNP signal ratio of $[0.75, 0.92)$ range. The $[0.92, 1.0)$ range has slightly decreased precision, because many called SNPs within the range of this SNP signal ratio have low abundances and are filtered away.



(a)



(b)

Figure 4.4: **AbSNP's Sensitivity and Precision Analysis** at the loci where GATK has only multiply mapped reads and fail to make any calls. The loci can be further categorized into the loci where abSNP has only unique reads (blue), the loci where abSNP has unique and multiply mapped reads (green), and the loci where abSNP has only multiply mapped reads (orange).

To see what factors contribute to the sensitivity and precision gains, the bars in Figure 4.4 are further categorized into blue, green and orange colors. The blue color bars refer to the loci where abSNP has only unique reads whereas GATK has only multiply mapped reads. These could be contributed by MAPQ filtering, which turns an original multiply mapped read into a unique read for abSNP. The orange color bars refer to the loci where abSNP also have only multiply mapped reads. These could be contributed by our SNP calling algorithm on repetitive regions. The green color bars refer to the loci where abSNP has both unique and multiply mapped reads. These could be contributed by both MAPQ filtering and SNP calling on repetitive regions.

4.4 Conclusion

While many algorithms have been developed in order to reveal SNPs in the human genome (both coding and non-coding regions) based on different sequencing technologies, SNPs at repetitive genomic regions remain mostly unexplored, because the current SNP discovery mainly relies on methods that ignores all multiply mapped reads due to repetitive genomic regions. We have developed abSNP that is especially designed in order to fill this gap (in particular with regard to the usage of RNA-seq), through Bayesian principles and filtering methods that utilize the unique products of RNA-seq abundance estimation that contain rich mapping quality information and estimated abundance. We believe this is the first work to explore this kind of information through an abundance estimation procedure. Our simulated results have shown abSNP's promising performance gain over the widely used GATK best practice. The main gain over GATK is in multiply mapped reads, where GATK does not make any SNP calls, whereas abSNP can get most SNP calls right on the high SNP signal ratio regions.

There are several directions we leave for future work: (1) Testing abSNP on real datasets is an important direction for future work. This is complicated by the lack of ground-truth SNP calls in multiply mapped regions. The present gold-standard datasets focus on SNPs in non-repetitive regions, which is the reason for the excellent performance on these

datasets. (2) abSNP does not factor RNA-editing into account, therefore the SNPs called are post-transcriptional. Thus abSNP in combination with DNA SNP calling can be used to quantify the impact of RNA-editing; although this requires strong statistical controls to reduce the impact of false-positives. (3) Currently abSNP assumes that both alleles have equal expression levels. While we have tested this in the simulation by having differing allele specific expressions, the algorithm can be potentially improved if the effect of allele-specific expression is accounted for. This is a chicken-and-egg problem since SNP calls are needed in order to quantify allele-specific expression, whereas, knowledge of allele-specific expression can improve SNP calls. Thus a joint SNP-calling and allele-specific expression detection can be useful. (4) An interesting application of abSNP is to utilize cancer datasets (e.g. DNA reads from normal and tumor samples, and RNA reads from tumor samples) to detect somatic mutations, different from the common approach to utilize only the DNA reads sequenced from normal samples and tumor samples. This is important because the common approach may not contain enough signal due to low variant allele frequency of somatic variants. RNA-seq could be highly expressed at somatic loci and therefore offer signals for better detection.There are several works exploring this direction [89][90]. One limitation for such kind of exploration is it requires access to authorized datasets of DNA (normal), DNA (tumor), and RNA (tumor) from TCGA [91].

Chapter 5

CELLMESH: PROBABILISTIC CELL-TYPE IDENTIFICATION USING INDEXED LITERATURE

5.1 Introduction

Single-cell RNA sequencing (scRNA-seq) is becoming an essential strategy for identifying cell types and understanding biological principles in complex tissues [23, 24, 25]. A typical scRNA-seq analysis workflow begins with the preparation of a gene-cell expression matrix (e.g. read quality control, read-to-transcriptome alignment and quantification), to which dimensionality reduction and clustering are then applied (Fig. 5.1). Finally, the resulting clusters are mapped to cell types based on their differentially expressed genes. While the initial steps in the analysis workflow are increasingly becoming more mature [92, 93, 94], the last step of assigning cell types to clustered cells remains an open problem [95].

There are several methods able to aid in assigning cell types based on cluster differentially expressed genes [96, 97, 98, 47, 51, 99].The methods require databases that connect genes to cell types and that can be queried. The databases are mainly collected either from a few specific studies [96, 97], from manual literature surveys [98, 47, 51], or from scRNA-seq experiments which have their clustered cells pre-annotated according to the cell-type markers manually compiled from literature [99]. The database query mechanisms can be summarized into two ways. The first way is to enumerate from database the cell types where the query genes are expressed [97, 98, 99]. The second way is to return a list of cell types sorted by their statistical significance with the query genes, based on Fisher's exact test [47, 96] or

---

This is joint work with Yue Zhang from Computer Science and Engineering Department, University of Washington.

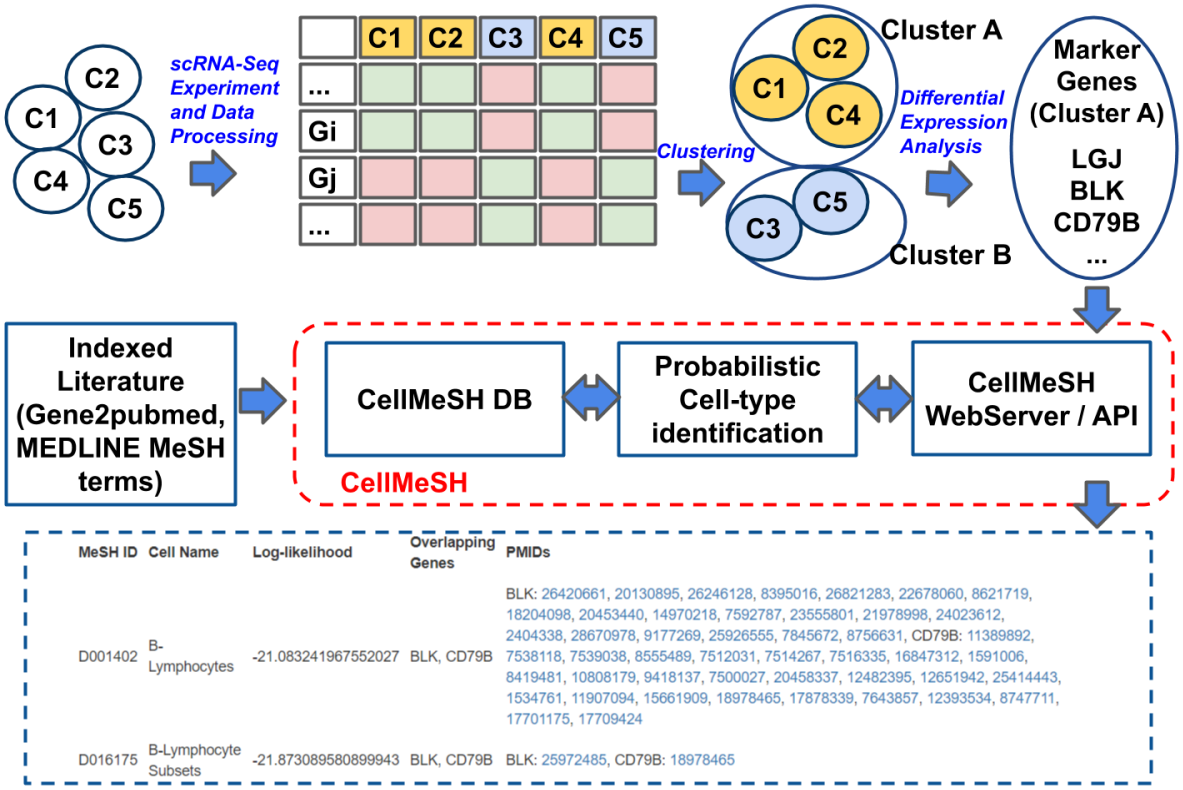


Figure 5.1: **CellMeSH overview.** CellMeSH mainly addresses the annotation of cell types, which is usually the last step of scRNA-seq analysis. It provides web service to take input of the differentially expressed genes of clustered cells, and produce output including ranked candidate cell types, overlapping genes and related literature resources. It relies on a database by linking the Gene2pubmed genes and MEDLINE MeSH cells. The database is queried by a probabilistic method based on maximum likelihood estimation.

Kolmogorov-Smirnov test [51]. The common issue for these cell-type identification methods is that their databases are not comprehensive; more critically it is also laborious to update and expand them.

Instead of identifying cell types of clusters based on cluster expressed genes, another line of recent work predict cell types of cells based on gene-cell expressions directly, by using machine learning [14, 100].For example [14] first utilizes prior experiments to learn a neural network model which is able to project the input gene expression into an embedding vector. It then aims to find the cell type from prior experiments so that its embedding vector is the

most similar to the one of the query gene expression. In [100], a classifier is first trained using inputs of gene expressions and a cell-type marker-gene file, and then it classifies the gene expression into the cell type from the marker file. The marker file is typically hand-curated, and as [101] shows, will strongly affect the classifier performance. These methods are unable to annotate the cell types that are either unseen in prior experiments (as used in [14]) or absent from the marker file (as used in [100]). Given these limitations, it is still useful to inspect the cluster expressed genes, as in a typical scRNA-seq analysis (Fig. 5.1).

Here, we propose CellMeSH, a new approach to annotate clustered single-cell data using indexed literature. Similar to the approaches that predict cluster cell types, CellMeSH is comprised of a database of gene cell-type mappings and a query method. Unlike the machine-learning based approaches that directly use gene expression data, CellMeSH does not need a separate training dataset, and could potentially handle cell types unseen in prior experiments.

CellMeSH builds its database in a scalable way, by automatically linking the gene and cell-type information from millions of publications, based on the indexed literature resources of Gene2pubmed [102] genes and MEDLINE [103] Medical Subject Headings (MeSH) [104] cell terms. Such large-scale gene-cell type linking makes the database comprehensive and easy to expand when new literature comes online. On the other hand, it could also make any particular gene-cell association error prone; but the true associations can be potentially unearthed by aggregating information across millions of papers.

To query the noisy CellMeSH database, we develop a novel probabilistic query method using maximum likelihood estimation. It is different from the existing statistical query methods [47, 96, 51], as they implicitly assume that the databases have only true gene cell associations and are therefore noiseless. These methods do not work best in our setting.

In addition to the database and the query method, CellMeSH also provides web server[1] and API[2]. The web server takes an input of the differentially expressed genes (or marker genes) of a cluster of cells, and outputs a list of candidate cell types sorted by their relevance

[1] `https://uncurl.cs.washington.edu/db_query`

[2] `https://github.com/shunfumao/cellmesh`

to the genes (Fig 5.1). For better interpretation, the overlapping genes and the corresponding publications for each candidate cell type are also listed. The open-sourced API enables CellMeSH to be integrated into other scRNA-seq analysis workflows, such as the UNCURL-App [38], our recently developed interactive scRNA-seq analysis web server and tool.

Through a variety of experiments on human and mouse scRNA-seq datasets, CellMeSH has demonstrated richer gene and cell-type information in its database, robust query method, and an overall superior annotation performance.

The remained chapter is organized as follows. We first describe the CellMeSH methods including the database and query method. We then show its annotation performance in terms of the annotation heatmap, the top-k accuracy, and the UNCURL-App workflow results. Lastly, we will conclude our work and discuss future directions.

5.2 Methods

5.2.1 CellMeSH Database

The CellMeSH database associates MeSH cell types indexed in the MEDLINE references and genes indexed in Gene2pubmed.

To construct the CellMeSH database (Fig. 5.2), we first filter MEDLINE for references containing MeSH cell types. MEDLINE [103] is a bibliographic database containing around 30 million references to articles in the biomedical field, including to all articles in PubMed. Each MEDLINE reference associates a subset of terms from Medical Subject Headings (MeSH) [104] with each article. MeSH is a hierarchically organized terminology for indexing and cataloging of biomedical information. Importantly, MeSH includes 570 terms related to cell types (nested under the MeSH category "Cell" with tree number A11). Filtering MEDLINE for MeSH cell types results in a reduced dataset of 3.8M articles.

Next, we integrate gene information from Gene2pubmed [102], a database that links standardized NCBI genes [102] with PubMed articles. Gene2pubmed currently references $20,164$ human and $27,322$ mouse gene names. A gene and a cell type are considered to

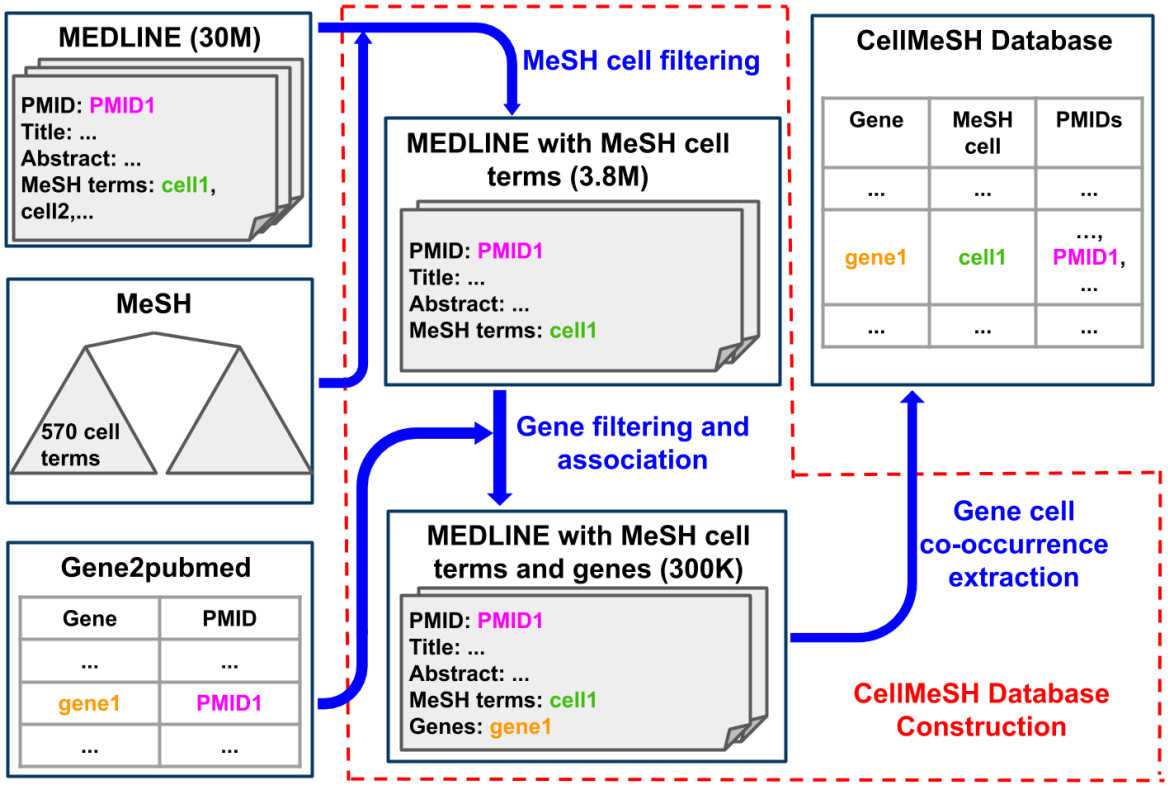


Figure 5.2: **CellMeSH database construction.** The construction is species specific. For instance, to construct the CellMeSH database for human, we start from all 30 million MEDLINE references, and filter to keep 3.8 million ones containing MeSH cell terms. We further filter to keep 300 thousand MEDLINE references co-occurring with human genes in Gene2pubmed. Each remained MEDLINE reference p contains several MeSH cell terms $\{c\}$ and several genes $\{g\}$, and we append p to each (g, c) pair in the final CellMeSH database.

co-occur if there is at least one article that is associated with the cell type in MEDLINE and with the gene in Gene2pubmed. We construct a matrix where each gene is a row and each cell type is a column and the entry denotes the number of articles in which the gene co-occurs with the cell type. For example, in the article with PubMed ID p=1591006, we have indexed gene g = "CD79B" (according to Gene2pubmed) and indexed MeSH cell types $c1$ = "B-Lymphocytes" and $c2$ = "Hematopoietic Stem Cells" (according to MEDLINE), meaning that g co-occurs with both $c1$ and $c2$ in p.

The CellMeSH database has its statistics as follows. For human, 3.8% of all possible

(20,164×570) gene-cell pairs have non-zero counts, and around 300,000 PubMed articles each contain at least one pair. For mouse, the numbers are 2.4% (27,322×570) gene-cell pairs with non-zero counts, and around 209,000 PubMed articles containing at least one pair.

5.2.2 Probabilistic Cell-Type Identification

There are two major issues with using a literature-derived database. The first issue is the publication biases. For example, certain genes or cell types are studied much more, so there will be more publications and thus more associations containing those genes or cell types. The second issue is noise in the gene-cell type mapping. The CellMeSH database is inherently noisy, as it links genes and cell types at an article level, and the simple fact of an article mentioning a cell type and a gene together does not imply that the gene serves as a marker for the cell type. This leads to potentially spurious associations between genes and cell types.

To better utilize the literature-derived database, we first apply TF-IDF [105] to transform the CellMeSH database, which addresses the issue of the publication biases of genes since the transformation could penalize the weights of common genes. Specifically each weight value $w_C(g)$, which is the number of co-occurrences of gene g in the cell type C, has been adjusted using $w_C(g) \leftarrow w_C(g) \times \log \frac{N_C}{K_g}$ where N_C is the total number of available cell types in the database, and K_g is the total number of cell types with non-zero weight in gene g, i.e. $K_g = \sum_C 1_{C:w_C(g)>0}$.

We then query the weight-adjusted database using a probabilistic method, which inherently addresses the issue of publication biases of cell types (as reflected in the normalization of $w_C(g)$ in Equation 5.1) and the issue of noises from spurious gene cell associations (as reflected in the noise factor α in Equation 5.1). Our query method takes input of $w_C(g)$ which is the adjusted weight of gene g in cell type C. The method also takes input of a query Q which is a subset of genes. The method outputs the database cell types sorted by their significance to the query.

Our probabilistic query method assumes the following generative model for the observed

query data (based on which the inference is performed): (1) a cell type is first chosen (with a uniform prior probability) (2) associated with the cell type is a probability distribution on the genes given by $p(g|C)$. A natural model for $p(g|C)$ is to take it to be proportional to the weight $w_C(g)$. (3) However, the previous model ensures that only genes with non-zero weight for the cell type will be present in the query - this need not be the case in our noisy dataset. To model this noise, we assume that with probability $1-\alpha$ the gene is sampled randomly from the list of all genes, and with probability α it is sampled from the cell-type specific gene distribution.

We also denote the total number of genes as N_g and the total number of genes with non-zero weight in cell type C as K_C, i.e., $K_C = \sum_g 1_{g:w_C(g)>0}$. Thus, the probability of picking a gene from a cell type can be written as follows:

$$P(g|C) = \begin{cases} \alpha \frac{w_C(g)}{\sum_{g'} w_C(g')} & \text{if } g \in Q \cap C \\ (1-\alpha)\frac{1}{N_g - K_C} & \text{if } g \in Q \cap \bar{C} \end{cases} \tag{5.1}$$

We denote by $P(Q|C)$, the probability that the list of query genes is obtained from a particular cell type C. We utilize $P(g|C)$ as the probability that we see gene g in the query given the cell type is C. Assuming that each gene is sampled independently, we have $P(Q|C) = \prod_{g \in C} P(g|C)$. We can then utilize maximum likelihood estimation to predict the cell type $\hat{C^*}$ that maximizes our chance of seeing the query:

$$\begin{aligned} \hat{C^*} &= \text{argmax}_C \log P(Q|C) \\ &= \text{argmax}_C \sum_g \log P(g|C) \end{aligned} \tag{5.2}$$

In practice, we fix α as 0.5. Note that α is a data dependent tuning parameter, which controls our belief about how noisy the dataset is. Small α implies more noise ($\alpha = 0$ implies genes are sampled randomly independent of cell type) and larger values of α are useful when the noise is less (for example, $\alpha = 1$ implies that the sampled distribution is identical to the cell-type distribution). In our experiments, $\alpha = 0.5$ is applied to all queries.

5.3 Results

5.3.1 Cell Type Annotation Performance

To quantify the overall cell-type identification performance of CellMeSH, we used three scRNA-seq datasets with known cell types: two Tabula Muris (TM) datasets [106], and the Mouse Cell Atlas (MCA) dataset [107].

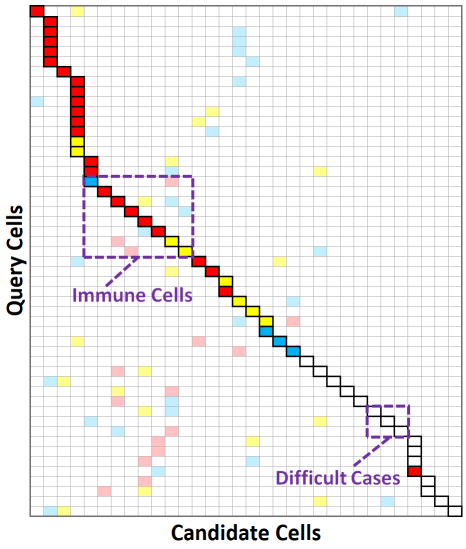
In the evaluation, we used clusters and reference annotations obtained in the original papers. For each cluster, we extracted the top $n = 50$ differentially expressed genes by one-versus-rest gene expression ratio. These genes are assumed to be the marker genes of the reference cell type and are used as a query input for CellMeSH. We then queried CellMeSH with marker genes for each cluster and visualized results using heatmaps that show how well the top-three retrieved candidate MeSH cell types agree with the reference cell type, according to the hand-made mappings between the reference cell types and their correct MeSH cell types.

Tabula Muris Datasets

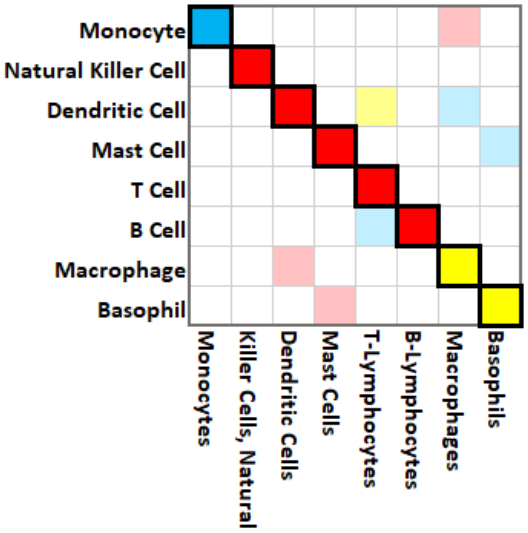
The Tabula Muris (TM) datasets [106] consist of scRNA-seq data from cells captured from 20 different tissues in 3-month-old mice. TM contains two sub-datasets, with cells captured using a microfluidic-droplet method (denoted as the TM-Droplet dataset) or by cell sorting (denoted as the TM-FACS dataset), containing 55656 and 44949 cells, respectively, with a total of 99 annotated cell types. Here we present the TM-Droplet dataset (Fig 5.3) but results for TM-FACS are similar (Fig 5.4 (a)).

The annotation results for the entire TM-Droplet dataset are summarized in the heatmap shown in Fig 5.3 (a). The diagonal bordered-boxes, indicating the expected annotations, are mostly filled with red, yellow or blue colors used to highlight the top 3 retrievals, which clearly demonstrates the effective annotation ability of CellMeSH. To see this more clearly, in Fig 5.3 (b) we focus on the annotation heatmap for only the immune cell types, where all query cells get their correct annotations within top 3 candidates.

(a) Tabula Muris Droplet



(b) Tabula Muris Droplet - Immune Cells



(c) Top 5 Results of Some Difficult Cells in Tabula Muris Droplet Dataset

	Rank	Promonocyte	Granulocyte	Alveolar Macrophage
Query				
Retrieved Top 5 Candidate MeSH Cells	1	Neutrophils	Neutrophils	Macrophages
	2	T-Lymphocytes	Myeloid Cells	Macrophages, Peritoneal
	3	Macrophages	Monocytes	Dendritic Cells
	4	Mesenchymal Stem Cells	Macrophages	Neutrophils
	5	Bone Marrow Cells	Granulocytes	Macrophages, Alveolar

Correct Candidate
Supercategory of Correct Candidate
Subcategory of Correct Candidate
Related Candidate

Figure 5.3: **Annotation results for Tabula Muris Droplet dataset. (a)** is the annotation heatmap for queries of all cells. The y-axis represents the query cells $\{r\}$ and x-axis represents the candidate cells $\{c\}$. A border-box for entry $(x = c, y = r)$ indicates c is the correct candidate for query r. The red, yellow or blue color indicates c has rank 1, rank 2, or rank 3 among retrieved results for r; the colors turn lighter if c is not the correct candidate. **(b)** is the annotation heatmap for queries of immune cells, all of which have their correct candidate cells within top 3 retrieved results. **(c)** shows the top 5 results for the three query cells that corresponding to the "Difficult Cases" in (a). They can still be relevant.

There are bordered-boxes forming vertical trajectories in Fig 5.3 (a). This is because we manually map several true cell types (e.g. Luminal Epithelial Cell of Mammary Gland, Kidney Collecting Duct Epithelial Cell, Bladder Urothelial Cell etc.) to the same MeSH cell

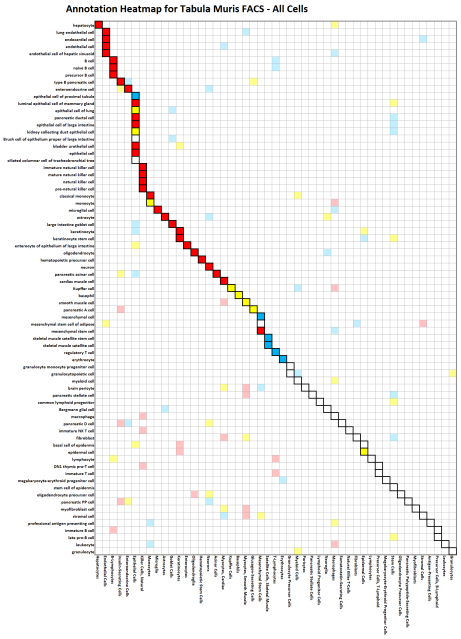
term (e.g. “Epithelial Cells”) due to the limited resolution of the MeSH cell types.

There are uncolored bordered-boxes in Fig 5.3 (a), implying the correct candidate may not exist due to a limit in the coverage of the MeSH cell types, or in many cases it is not within the top 3 retrieved results due to the noise in CellMeSH database. Still, the CellMeSH query results provide useful insights into the true cell types. To illustrate this, in Fig 5.3 (c), we show three queries that correspond to the “Difficult Cases” in Fig 5.3 (a). The first query Promonocyte does not have an exact same MeSH term; the closest term we could manually match is “Monocyte-Macrophage Precursor Cells”. However among the top 5 retrieved results there are “Monocytes” and “Bone Marrow Cells”, which are both closely relevant because a Promonocyte is a cell arising from a Monoblast (in Bone Marrow [108]) and developing into a Monocyte [109]. For the second query Granulocyte, the correct MeSH term “Granulocytes” is rank-5 in the retrieved results probably due to relatively fewer citations in the database. However the top 2 results “Neutrophils” and “Myeloid Cells” are respectively the subcategory and supercategory of “Granulocytes” in the MeSH tree. “Monocytes” and “Macrophages” are also related as Neutrophils can secrete products that stimulate Monocytes and Macrophages [110]. The third query Alveolar Macrophage has its correct MeSH candidate “Macrophages, Alveolar” with rank-5. The rank-1 result “Macrophages” is also close as it is a supercategory of “Macrophages, Alveolar”.

Mouse Cell Atlas Dataset

The Mouse Cell Atlas (MCA) dataset [107] consists of scRNA-seq data from 6- to 10-week-old mice, sampled from a large variety of tissues. It contains over 200,000 cells after batch effect filtering, and 840 annotated cell types. These cell types are further collapsed into 204 cell types in our evaluation. The MCA heatmap result is generally similar to the TM heatmaps, with mostly accurate annotations (Fig 5.4 (b)).

(a) Tabula Muris FACS



(b) Mouse Cell Atlas

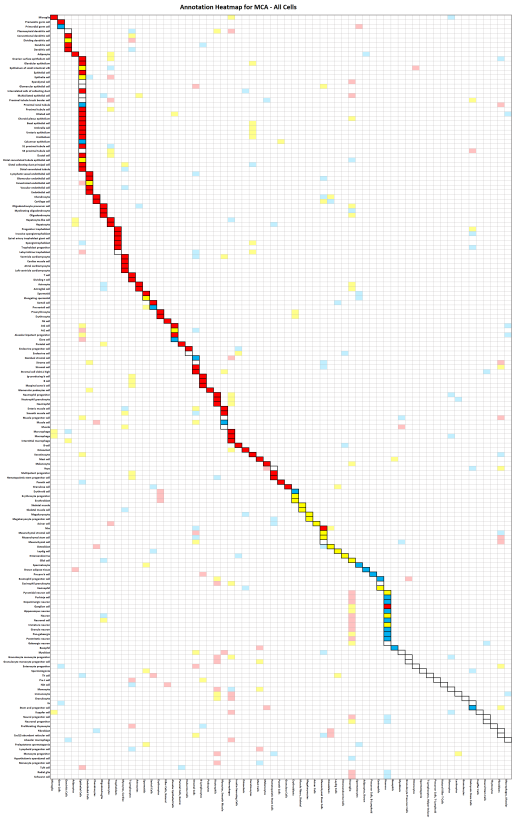


Figure 5.4: **Annotation results for (a) TM-FACS and (b) MCA datasets**

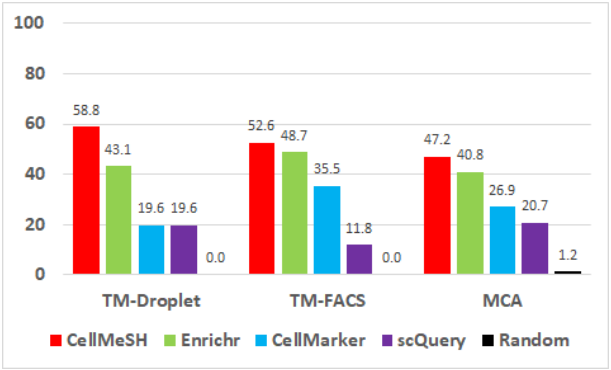
5.3.2 Performance of Top-k Accuracy

To quantitatively understand the detailed cell-type identification performance of CellMeSH, we continue to use the three mouse datasets (TM-Droplet, TM-FACS and MCA) to evaluate their top-k ($k = 1, 3$) accuracies.

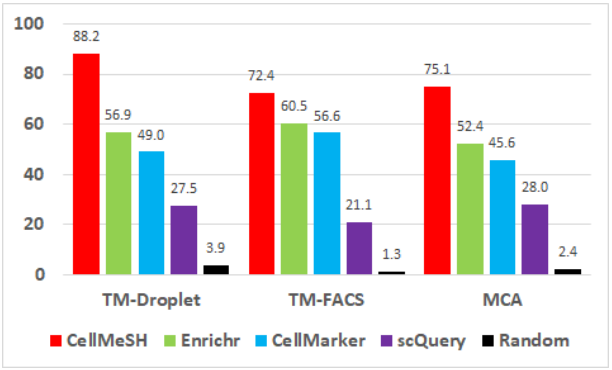
We compare the query results from CellMeSH to those from other approaches (Fig 5.5 (a)(b)) or from other query methods (Fig 5.5 (c)(d)) or from other databases (Fig 5.5 (e)(f)). The query procedure is similar to the procedure for heatmaps described above. For each annotated cell type, we use its top $n = 50$ differentially expressed genes as query genes. The other approaches or databases could use a different ontology and therefore have different cell type names from the MeSH terms. In order to calculate the accuracies of these other results,

we also manually created mappings between the given query cell types and the candidate cell types from other ontologies.

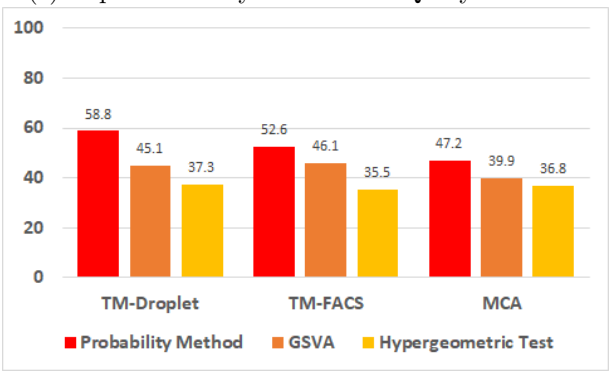
(a) Top-1 Accuracy of Different Approaches



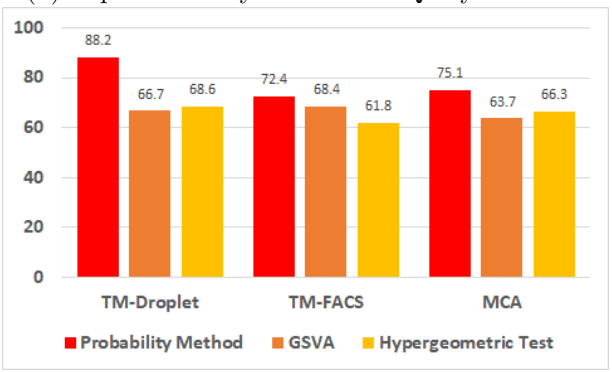
(b) Top-3 Accuracy of Different Approaches



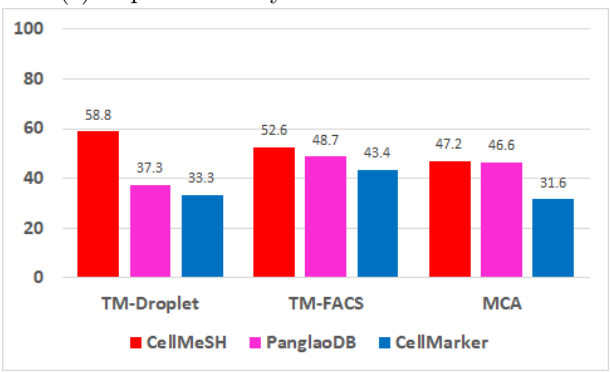
(c) Top-1 Accuracy of Different Query Methods



(d) Top-3 Accuracy of Different Query Methods



(e) Top-1 Accuracy of Different Databases



(f) Top-3 Accuracy of Different Databases

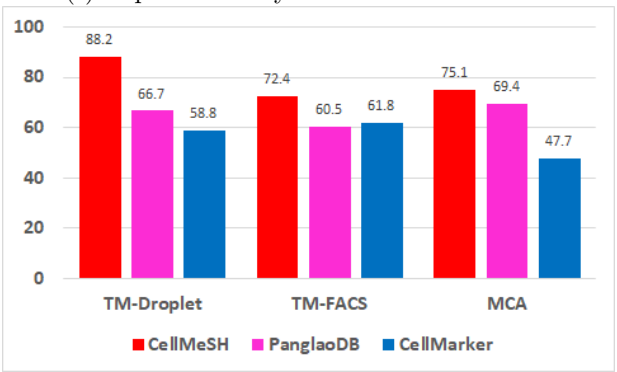


Figure 5.5: **Compare CellMeSH to other approaches (a)(b), to other query methods (c)(d) and to other databases (e)(f).** The bar plots have y-axis as top-k (k=1 or 3) accuracy (%) and x-axis as the different mouse datasets. Top-k accuracy refers to the percentage of queries where one of the candidate cells among the top k retrieved cells is accurate. CellMeSH demonstrates a consistent better top-1 and top-3 accuracy.

Top-k Accuracy Gain versus Existing Approaches

In Fig 5.5 (a)(b), we first compare CellMeSH with two other web servers: Enrichr [96] and scQuery [111]. We are unable to compare to the other existing web servers of PanglaoDB [99], CellMarker [98] and CellFinder [97] in an automatic way. Moreover, CellMarker and CellFinder do not rank their query results in statistical significance. Instead, we downloaded the existing CellMarker [98] database and queried it using the hypergeometric test. The CellFinder database is unavailable, and the PanglaoDB database will be discussed further in Fig 5.5 (e)(f). We also include the random retrieval results as a lower bound reference.

CellMeSH provides the most accurate results compared to all of the other approaches for all of the three mouse datasets. Specifically, in the TM-Droplet dataset, CellMeSH has achieved top-1 accuracy of 58.8%, meaning that in 58.8% of queries, the first retrieved candidate cell type is correct. The top-1 accuracy is 15.7% higher than that of Enrichr, the second best system. This is reasonable because the Enrichr cell types essentially come from the Mouse Gene Atlas (MGA) database [112], which contains only 96 cell types. Besides, some of the MGA cell types (such as "Heart", "Kidney", and "Stomach" etc.) actually refer to organs. Although CellMeSH itself has limitations, in terms of cell types, overall it still has much higher coverage and resolution than Enrichr does. For example, for query Classical Monocyte, while CellMeSH returns "Monocytes" as the first candidate, there is no monocyte term covered in MGA and Enrichr returns its first result as "Macrophage Bone Marrow 6hr LPS". For query Duct Epithelial Cell, while CellMeSH returns "Epithelial Cells" as the first result, Enrichr returns only the organ terms "Bladder", "Liver" and "Stomach" as its top 3 results. The top-3 accuracy of CellMeSH further increases to 88.2% (implying 88.2% of queries get at least one of the top 3 results correct), which is 31.3% higher than Enrichr.

CellMeSH is also consistently the best in the other two datasets. Its top-1 (or top-3) accuracy is 3.9% (or 11.9%) higher in the TM-FACS dataset, and 6.4% (or 22.7%) higher in the MCA dataset, than the second best method, Enrichr.

Top-k Accuracy Gain from Probabilistic Method

Both the CellMeSH database and the probabilistic cell-type identification method contribute to the overall performance gains of CellMeSH. To isolate the contribution of the probabilistic method to the overall query performance of CellMeSH, we have compared it to the hypergeometric test [47] and GSVA [51], as suggested in [52], on the CellMeSH database for the three mouse datasets.

As shown in Fig 5.5 (c)(d), the probabilistic method performs uniformly better than other methods. Compared to the best performance of GSVA and the hypergeometric test, the probabilistic query method has a top-1 accuracy gain of 13.7%, 6.6% and 7.3% in the TM-Droplet, TM-FACS and MCA datasets respectively. The numbers for top-3 accuracy gain are 19.6%, 3.9% and 8.8%.

We attribute these gains to the method's better utilization of gene weights. Indeed, we need to adjust ORA and GSVA to the setting where we take set of genes as input to query a noisy database. This requires ORA, which originally took set of genes to query a noise-free binary database, now to treat all genes in the database with the same weight. Consequently, some useful aggregated information could be lost, and some wrong associations could be amplified. As to GSVA, it essentially utilizes gene rankings as their weights to calculate an enrichment score. These integer rankings may not properly reflect the actual gene weights. Moreover, GSVA, which originally took gene expression to query a noise-free binary database and had its gene rankings come from the input expression, now has its "gene expression" and gene rankings come from the noisy database.

Top-k Accuracy Gain from CellMeSH Database

To isolate the contribution of the CellMeSH database, we have prepared gene-cell co-occurrence matrices from PanglaoDB [99] and CellMarker [98], both of which are manually compiled from the literature, and we compare the CellMeSH database to them.

We query the CellMeSH and CellMarker databases using the probabilistic query method,

since these databases are count-valued matrices and the probabilistic query method is effective in reduce their inherent noise, as illustrated in Fig 5.5 (c)(d). For PanglaoDB, the query method is the hypergeometric test, since the database is essentially a binary matrix.

As Fig 5.5 (e)(f) illustrates, using the CellMeSH database achieves a higher accuracy than using the PanglaoDB and CellMarker databases. Compared to the best performance out of PanglaoDB and CellMarker databases, the CellMeSH database has top-1 accuracy gain of 21.6%, 3.9% and 0.6% in the TM-Droplet, TM-FACS and MCA datasets respectively. The numbers for top-3 accuracy gain increase to 21.6%, 10.5% and 5.7%.

We attribute the overall gains to the rich gene-cell signals contained in the CellMeSH database, especially when the noise is properly handled using the probabilistic query method. Specifically, The CellMeSH database is larger scale than the hand-curated CellMarker [98] and PanglaoDB [99] databasets. CellMeSH contains 20164 genes, 570 cell types and 3.8% co-occurring gene-cell pairs for human; the numbers are 27322, 570 and 2.4% for mouse. CellMarker contains around 8900 genes, 340 cell types and less than 1% co-occurring gene-cell pairs for human; the numbers are around 7200, 310 and less than 1% for mouse. In PanglaoDB, there are around 4600 genes, 178 cell types and 1% co-occurring gene-cell pairs for both human and mouse.

5.3.3 Annotation Results of UNCURL-App

CellMeSH has its database and query methods open sourced as API, so it can be integrated into other scRNA-seq analysis as illustrated in Fig 5.1. We have integrated it into our recently developed web server UNCURL-App [38], which combines data preprocessing, dimensionality reduction, clustering, differential expression, and interactive data analysis within an online graphical user interface.

We show the UNCURL-App/CellMeSH analysis results, as initially presented in [38], using two datasets from the human peripheral blood mononuclear cells (PBMC) [113] and from mouse spinal cord [114]. The major difference from previous heatmap and top-k analysis is that we do not use the clusters and reference annotations obtained in the original papers.

Instead, the clusters are generated from the initial gene-cell matrix in the UNCURL-App. Each cluster may contain cells with different originally annotated cell types.

Human PBMC dataset

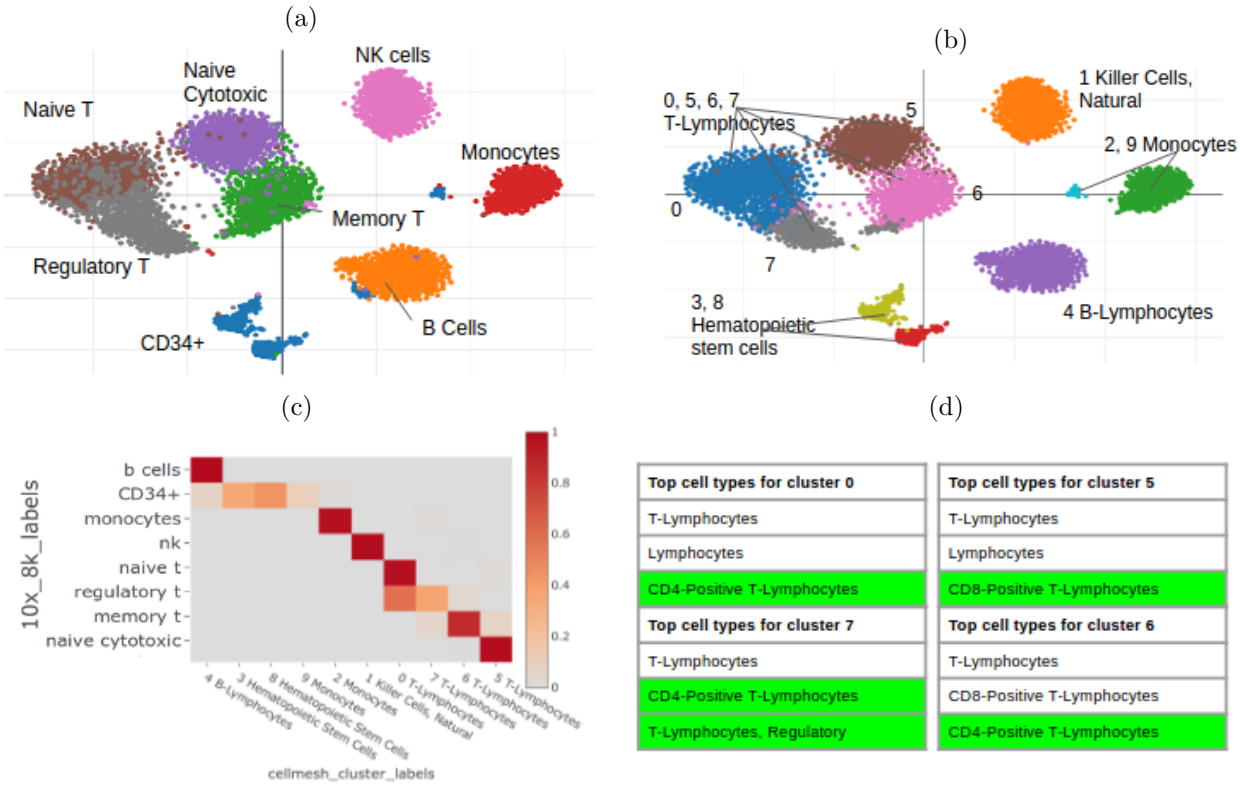


Figure 5.6: **UNCURL-APP/CellMeSH analysis results for PBMC dataset.** (a) Resulting clusters and their ground truth labels. (b) CellMeSH annotations. (c) Heatmap where y-axis represents ground truth labels and x-axis represents CellMeSH labels. Cells belonging to the same ground truth label may fall in different clusters and have different assigned labels (e.g. CD34+). (d) Top 3 CellMeSH labels for clusters labeled as "T-Lymphocytes" in (b). Green color indicates expected results.

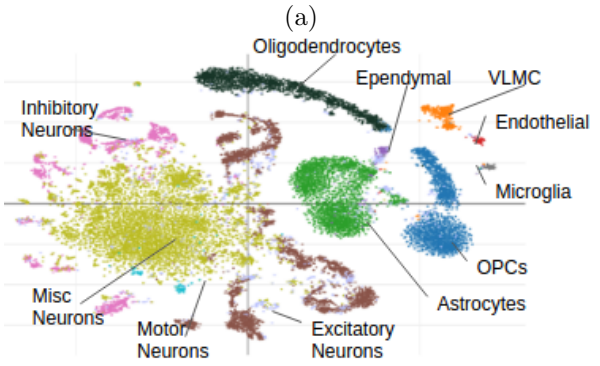
This dataset comprised of 8000 human peripheral blood mononuclear cells (PBMC) from [113]. It was created by randomly sampling 1000 cells from each of 8 scRNA-seq datasets comprised of cells that were flow-sorted based on known cell-type markers. Thus, the ground truth cell-type labels represent pure samples.

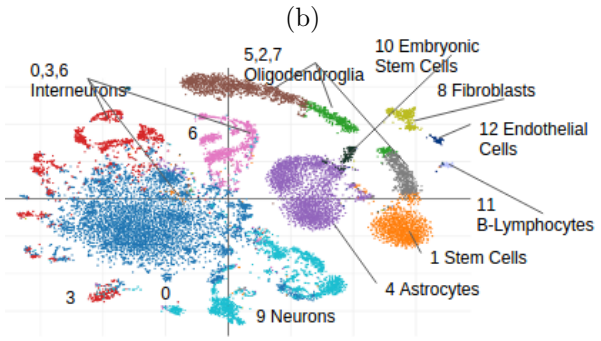
The dataset was processed by UNCURL-App with default settings to generate 10 initial clusters. Then the clustering and cell-type assignment are iteratively adjusted, by first visually inspecting the dimensionality reduction and assigned putative cell labels, and next merging some clusters appearing very similar as well as splitting off some cluster appearing to contain separate smaller clusters, and then rerunning clustering and cell-type assignment.

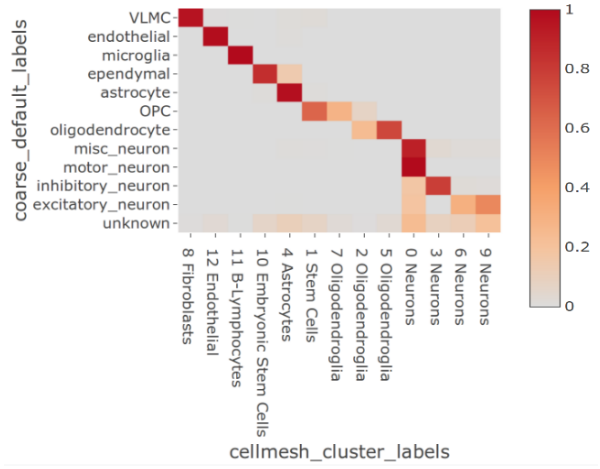
Fig 5.6 shows the resulting clusters and the cell-type assignments. There is good correspondence with the ground truth clusters and labels. Cells of the same ground truth type are generally assigned to the same cluster (Figure 5.6 (a)), and the cluster labels returned by CellMeSH generally correspond to the ground truth labels (Figure 5.6 (b)(c)). CD34+ cells are generally recognized as hematopoietic stem cells [115], so the CellMeSH label here seems to be accurate. One major difference is that CellMeSH labeled all four T cell subtypes as "T-Lymphocytes", even though they were clustered into distinct clusters. To investigate further, we looked at the top 3 CellMeSH labels for these clusters, not just the top one. These results are listed in Fig 5.6 (d), with the cell types most similar to the ground truth highlighted in green. For example, Cluster 0 corresponds to naive T-cells, which are selected as CD4+, and the "CD4-Positive T-Lymphocytes" label is the third highest label. Cluster 5 corresponds to naive cytotoxic T-cells, which are CD8+, and the "CD8-Positive T-Lymphocytes" label is the third highest label. Cluster 6 corresponds to memory T-cells, which can be either CD8+ or CD4+; the second and third labels are "CD8-Positive T-Lymphocytes" and "CD4-Positive T-Lymphocytes". Cluster 7 corresponds largely to regulatory T-cells, which are CD4+, and the second and third highest CellMeSH labels are "CD4-Positive T-Lymphocytes" and "T-Lymphocytes, Regulatory". The top 3 CellMeSH results show a good correspondence between the true and assigned labels at a more fine-grained level.

5.3.4 Mouse spinal cord dataset

This is a larger dataset comprised of 22,614 mouse spinal cord nuclei from 2 and 11-day old mice sequenced using SPLiT-seq [114]. This dataset has 44 annotated cell types, which is substantially more than the PBMC dataset. However, many of these annotated cell types are







(d)

Top cell types for cluster 8	Top cell types for cluster 10
Fibroblasts	Embryonic Stem Cells
Stromal Cells	Neurons, Afferent
Endothelial Cells	Olfactory Receptor Neurons
Mesenchymal Stem Cells	Stem Cells
Pericytes	Spermatozoa
Top cell types for cluster 11	**Top cell types for cluster 12**
B-Lymphocytes	Endothelial Cells
T-Lymphocytes	Hematopoietic Stem Cells
Bone Marrow Cells	Endothelial Progenitor Cells
Macrophages	Pericytes
Hematopoietic Stem Cells	Myocytes, Smooth Muscle

Figure 5.7: **UNCURL-App/CellMeSH analysis results for spinal cord dataset.** (a) Resulting clusters and their ground truth labels. (b) CellMeSH annotations. (c) Heatmap where y-axis represents ground truth labels and x-axis represents CellMeSH labels. Cells belonging to the same ground truth label may fall in different clusters and have different assigned labels (e.g. OPC). (d) Top 5 CellMeSH labels for some non-neuron clusters. Green color indicates expected results.

closely related (For example, there are 15 types of excitatory neurons.), so for the "ground truth" comparisons, we combine many of the annotated cell types into larger clusters of similar cells. Still, many cell types are similar, with many subtypes of neurons.

The clustering results are generated in a similar iterative way as those of the PBMC dataset. There is also generally good concordance between the cell types from the original paper and the annotated clusters generated by UNCURL-App/CellMeSH, as shown in Fig

5.7. Also, similarly to the PBMC dataset, the CellMeSH annotations are generally coarser grained than the original hand-annotated labels, with all of the neuron clusters being labeled as "Interneurons" or just "Neurons". For the non-neuronal results, interpreting the labels identified by CellMeSH is more challenging. Oligodendrocytes, astrocytes, and endothelial cells were correctly identified. For cluster 8, the ground-truth label was VLMC, or vascular and leptomeningeal cells. This is a highly specific category that does not appear in the MeSH ontology but was used as a cell label in [116]. Still, while coarse, the top labels suggested by CellMeSH ("Fibroblasts", "Stromal Cells", "Mesenchymal Stem Cells") seem consistent with cells derived from the meninges, the membrane enveloping the brain and spinal cord. In cluster 10, the ground-truth label Ependymal was not correctly identified by CellMeSH, and the returned results did not seem to relate to ependymal cells. This points to a paucity of annotated publications with gene markers for this cell type. For cluster 11, all of the top CellMeSH results were immune cells, a group which the published label, microglia, belongs to. In fact, "Microglia" was one of the top 10 cell types returned.

5.4 Conclusion

We have developed CellMeSH, a web server to identify cell types directly from literature, in order to make the scRNA-seq analysis more convenient. Essentially, CellMeSH contains a new database constructed by associating the genes and cells from existing indexed literature resources including Gene2pubmed and MEDLINE MeSH terms, and it queries the database using a newly designed probabilistic query method to better reduce noise. Through the experiments on several human and mouse scRNA-seq datasets, CellMeSH has demonstrated superior cell-type identification performance. We expect that the CellMeSH web server, together with its open-sourced database and the query method, could be complementary to existing approaches and assist community efforts for better automatic cell-type identification.

Nevertheless, there are still several limitations with CellMeSH. Particularly, the cell-type annotations provided by MeSH terms are somewhat coarse, and might not be enough to represent a comprehensive listing of all fine-grained (sub) cell types present in model

organisms such as human or mouse. For non-mammalian species such as C. elegans and plants, the gene-cell information could be limited due to a lack of indexed publications.

One way the CellMeSH database could be extended is through natural language processing on full text articles (including supplementary files). Ideally we would be able to identify new cell types in papers using unsupervised or semi-supervised named entity recognition.

Moreover, there are also terms within the MeSH ontology that might be useful but are not under the "Cell" heading, such as tissues, organs and diseases. Designing the query methods utilizing these informations is an interesting future direction. For instance, we could refine our search scope if we know the tissue information of the query; or if such information is missing, we could provide them from an extended Cell/Tissue-MeSH database.

Chapter 6

FUTURE DIRECTIONS

In this thesis, we have presented novel computational solutions for three computational problems for RNA-seq data analysis, including RefShannon for transcriptome assembly, abSNP for variant calling, and CellMeSH for single-cell RNA-seq annotation. In each chapter, we have also listed specific future directions for them. In this chapter, we will take a unified abstract view at them. Guided by this, we will list some additional open problems that could be interesting to investigate.

To begin with, we can view all of presented computational solutions as information processing systems, which process the source information by certain mechanism for some purpose. Such systems can be improved by either (1) using a richer information source, or (2) using a better information processing mechanism.

Our proposed solutions in this thesis have obtained improved performance mainly by utilizing a richer information source. Specifically, RefShannon explores the linking information from pair end reads as well as the varying abundances from RNA transcripts, abSNP explores the varying read mapping quality scores as well as the varying abundances from RNA transcripts, and CellMeSH explores the gene cell information from prior literature.

Both the assembly (e.g. RefShannon) and the variant calling (e.g. abSNP) problems shall be further improved by utilizing more information sources. For instance, the third-generation long read sequencing technologies have not been fully explored yet, but their long read sequences will definitely provide better linking information for these problems. We have conducted some study investigating long reads [117] whereas much remains to explore.

Both the assembly and variant calling problems shall also be further improved by utilizing a better information processing mechanism. The information processing systems are

inherently noisy due to internal procedures that involve heuristics. Actually many tools have their own default set of parameters, which could be sub-optimal options for a specific input of information source. It would be really interesting to design the systems to adapt their settings (parameters) to the specific input information source. We have also conducted some study for this, by using Bayesian Optimization to tune assembly parameters, which shows some promising results [118]. Still, much is remained for verification.

The downstream cell-type annotation problem will also be greatly benefited from a better source of gene and cell information. While we take an approach to harvest gene and cell information from literature, it is unlikely to produce an ultimate single best source that contains enough information. The annotation performance will always be better if more information sources are aggregated properly. Instead of harvesting information to create a new source, novel integration of existing knowledge in an automatic and easy-to-extend way could also be promising. Existing approaches either present results from separate sources [119] or require manual efforts for integration [120]. This is a challenging problem because many sources have different formats and levels of resolution.

Lastly, the cell-type annotation problem could also benefit from an improved information processing mechanism. While our current approach follows a typical single-cell analysis workflow to annotate clustered cells, it unavoidably contains many intrinsic processing noises. For instance, the number of differentially expressed genes could affect the annotation accuracy. Instead, an end-to-end solution (treating the processing as a black box, such as neural network) is appealing. There are existing works investigated in [101], however they could fail for unseen cell types. Connecting these approaches with literature resources could also be an interesting direction.

BIBLIOGRAPHY

[1] J.A Witkowski. The discovery of 'split' genes: a scientific revolution. *Trends in Biochemical Sciences*, 13(3):110–113, March 1988.

[2] D. Michael and L. Manyuan. Intron–exon structures of eukaryotic model organisms. *Nucleic Acids Research*, 27(15):3219–3228, August 1999.

[3] Walter Gilbert. Why genes in pieces? *Nature*, 271(5645):501–501, February 1978.

[4] R E Breitbart, A Andreadis, and B Nadal-Ginard. Alternative splicing: A ubiquitous mechanism for the generation of multiple protein isoforms from single genes. *Annual Review of Biochemistry*, 56(1):467–495, June 1987.

[5] Douglas L. Black. Mechanisms of alternative pre-messenger RNA splicing. *Annual Review of Biochemistry*, 72(1):291–336, June 2003.

[6] Gunter Meister. *RNA Biology: An Introduction*. Wiley-VCH, 2011.

[7] Wikipedia. Rna - wikipedia, 2020. `https://en.wikipedia.org/wiki/RNA`, Last accessed on 2020-05-08.

[8] Wikipedia. Protein - wikipedia, 2020. `https://en.wikipedia.org/wiki/Protein`, Last accessed on 2020-05-08.

[9] Genome News Network. Genetics and genomics timeline, 2020. `http://www.genomenewsnetwork.org/resources/timeline/1960_mRNA.php`, Last accessed on 2020-05-08.

[10] Wikipedia. De novo transcriptome assembly - wikipedia, 2020. `https://en.wikipedia.org/wiki/De_novo_transcriptome_assembly`, Last accessed on 2020-05-08.

[11] Wikipedia. Genetic disorder - wikipedia, 2020. `https://en.wikipedia.org/wiki/Genetic_disorder`, Last accessed on 2020-05-08.

[12] Mohammad Al-Haggar, Agnieszka Madej-Pilarczyk, Lukasz Kozlowski, Janusz M Bujnicki, Sohier Yahia, Dina Abdel-Hadi, Amany Shams, Nermin Ahmad, Sahar Hamed, and Monika Puzianowska-Kuznicka. A novel homozygous p.arg527leu LMNA mutation in two unrelated egyptian families causes overlapping mandibuloacral dysplasia and progeria syndrome. *European Journal of Human Genetics*, 20(11):1134–1140, May 2012.

[13] Rolf I. Skotheim and Matthias Nees. Alternative splicing in cancer: Noise, functional, or systematic? *The International Journal of Biochemistry & Cell Biology*, 39(7-8):1432–1449, July 2007.

[14] Amir Alavi, Matthew Ruffalo, Aiyappa Parvangada, Zhilin Huang, and Ziv Bar-Joseph. scQuery: a web server for comparative analysis of single-cell RNA-seq data. *bioRxiv*, page 323238, May 2018.

[15] David T. Szabo. Transcriptomic biomarkers in safety and risk assessment of chemicals. In *Biomarkers in Toxicology*, pages 1033–1038. Elsevier, 2014.

[16] Wikipedia. Rna-seq, 2020. `https://en.wikipedia.org/wiki/RNA-Seq`, Last accessed on 2020-05-08.

[17] Wikipedia. Real-time polymerase chain reaction, 2020. `https://en.wikipedia.org/wiki/Real-time_polymerase_chain_reaction`, Last accessed on 2020-05-08.

[18] Wikipedia. Dna microarray, 2020. `https://en.wikipedia.org/wiki/DNA_microarray`, Last accessed on 2020-05-08.

[19] Adam Roberts and Lior Pachter. Streaming fragment assignment for real-time analysis of sequencing experiments. *Nat Meth*, 10(1):71–73, Jan 2013. Brief Communication.

[20] Raj D. Maitra, Jungsuk Kim, and William B. Dunbar. Recent advances in nanopore sequencing. *ELECTROPHORESIS*, 33(23):3418–3428, November 2012.

[21] The long view on sequencing. *Nature Biotechnology*, 36(4):287–287, April 2018.

[22] Nicola De Maio, Liam P. Shaw, Alasdair Hubbard, Sophie George, Nicholas D. Sanderson, Jeremy Swann, Ryan Wick, Manal AbuOun, Emma Stubberfield, Sarah J. Hoosdally, Derrick W. Crook, Timothy E. A. Peto, Anna E. Sheppard, Mark J. Bailey, Daniel S. Read, Muna F. Anjum, A. Sarah Walker, and Nicole Stoesser and. Comparison of long-read sequencing technologies in the hybrid assembly of complex bacterial genomes. *Microbial Genomics*, 5(9), September 2019.

[23] D. A. Jaitin, E. Kenigsberg, H. Keren-Shaul, N. Elefant, F. Paul, I. Zaretsky, A. Mildner, N. Cohen, S. Jung, A. Tanay, and I. Amit. Massively parallel single-cell RNA-seq for marker-free decomposition of tissues into cell types. *Science*, 343(6172):776–779, feb 2014.

[24] Dmitry Usoskin, Alessandro Furlan, Saiful Islam, Hind Abdo, Peter Lönnerberg, Daohua Lou, Jens Hjerling-Leffler, Jesper Haeggström, Olga Kharchenko, Peter V Kharchenko, Sten Linnarsson, and Patrik Ernfors. Unbiased classification of sensory neuron types by large-scale single-cell RNA sequencing. *Nature Neuroscience*, 18(1):145–153, nov 2014.

[25] A. Zeisel, A. B. Munoz-Manchado, S. Codeluppi, P. Lonnerberg, G. La Manno, A. Jureus, S. Marques, H. Munguba, L. He, C. Betsholtz, C. Rolny, G. Castelo-Branco, J. Hjerling-Leffler, and S. Linnarsson. Cell types in the mouse cortex and hippocampus revealed by single-cell RNA-seq. *Science*, 347(6226):1138–1142, feb 2015.

[26] C. Trapnell, B. A. Williams, G. Pertea, A. Mortazavi, G. Kwan, M. J. van Baren, S. L. Salzberg, B. J. Wold, and L. Pachter. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nat. Biotechnol.*, 28(5):511–515, May 2010.

[27] Mihaela Pertea, Geo M Pertea, Corina M Antonescu, Tsung-Cheng Chang, Joshua T Mendell, and Steven L Salzberg. Stringtie enables improved reconstruction of a transcriptome from rna-seq reads. *Nat Biotech*, 33:290295, 03 2015.

[28] M.A. DePristo, E.Banks, R.Poplin, K.V. Garimella, J.R.Maguire, C.Hartl, A.A. Philippakis, G. Angel, M.A. Rivas, M. Hann, A. McKenna, T.J. Fennell, A.M. Kernytsky, A.Y. Sivachenko, K. Cibulskis, S.B. Gabriel, D. Altshuler, and M.J. Daly. A framework for variation discovery and genotyping using next-generation DNA sequencing data. *Nature Genetics*, 43(5):491–498, apr 2011.

[29] Abecasis Lab. *GlfMultiples*. `http://genome.sph.umich.edu/wiki/GlfMultiples`.

[30] H. Li. A statistical framework for SNP calling, mutation discovery, association mapping and population genetical parameter estimation from sequencing data. *Bioinformatics*, 27(21):2987–2993, sep 2011.

[31] Erik Garrison and Gabor Marth. Haplotype-based variant detection from short-read sequencing, 2012.

[32] D. C. Koboldt, Q. Zhang, D. E. Larson, D. Shen, M. D. McLellan, L. Lin, C. A. Miller, E. R. Mardis, L. Ding, and R. K. Wilson. VarScan 2: Somatic mutation and

copy number alteration discovery in cancer by exome sequencing. *Genome Research*, 22(3):568–576, 2012.

[33] X. Tang, S. Baheti, K. Shameer, K. J. Thompson, Q. Wills, N. Niu, I. N. Holcomb, S. C. Boutet, R. Ramakrishnan, J. M. Kachergus, J.-P. A. Kocher, R. M. Weinshilboum, L. Wang, E. A. Thompson, and K. R. Kalari. The eSNV-detect: a computational system to identify expressed single nucleotide variants from transcriptome sequencing data. *Nucleic Acids Research*, 42(22):e172–e172, oct 2014.

[34] Gokul Ramaswami Robert Piskol and Jin Billy Li. Reliable identification of genomic variants from RNA-seq data. *The American Journal of Human Genetics*, 93(4):641–651, 2013.

[35] R. Goya, M. G.F. Sun, R. D. Morin, G. Leung, G. Ha, K. C. Wiegand, J. Senz, A. Crisan, M. A. Marra, M. Hirst, D. Huntsman, K. P. Murphy, S. Aparicio, and S. P. Shah. SNVMix: predicting single nucleotide variants from next-generation sequencing of tumors. *Bioinformatics*, 26(6):730–736, feb 2010.

[36] Sreeram Kannan, Joseph Hui, Kayvon Mazooji, Lior Pachter, and David Tse. Shannon: An information-optimal de novo rna-seq assembler. *bioRxiv*, 2016.

[37] Shunfu Mao, Soheil Mohajer, Kannan Ramachandran, David Tse, and Sreeram Kannan. abSNP: RNA-Seq SNP Calling in Repetitive Regions via Abundance Estimation. In Russell Schwartz and Knut Reinert, editors, *17th International Workshop on Algorithms in Bioinformatics (WABI 2017)*, volume 88 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 15:1–15:14, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[38] Yue Zhang, Shunfu Mao, Sumit Mukherjee, Sreeram Kannan, and Georg Seelig. UNCURL-app: Interactive database-driven analysis of single cell RNA sequencing data. April 2020.

[39] Wikipedia. De bruijn graph - wikipedia, 2020. `https://en.wikipedia.org/wiki/De_Bruijn_graph`, Last accessed on 2020-05-10.

[40] Neil C. Jones. *An Introduction to Bioinformatics Algorithms (Computational Molecular Biology)*. The MIT Press, aug 2004.

[41] D. R. Zerbino and E. Birney. Velvet: Algorithms for de novo short read assembly using de bruijn graphs. *Genome Research*, 18(5):821–829, February 2008.

[42] Thomas Gatter and Peter F Stadler. Ryūtō: network-flow based transcriptome reconstruction. *BMC Bioinformatics*, 20(1), April 2019.

[43] E Cnlar. *Probability and stochastics*. Springer, New York, 2011.

[44] Wikipedia. Gene set enrichment analysis - wikipedia, 2020. `https://en.wikipedia.org/wiki/Gene_set_enrichment_analysis`, Last accessed on 2020-05-10.

[45] Michael Ashburner, Catherine A. Ball, Judith A. Blake, David Botstein, Heather Butler, J. Michael Cherry, Allan P. Davis, Kara Dolinski, Selina S. Dwight, Janan T. Eppig, Midori A. Harris, David P. Hill, Laurie Issel-Tarver, Andrew Kasarskis, Suzanna Lewis, John C. Matese, Joel E. Richardson, Martin Ringwald, Gerald M. Rubin, and Gavin Sherlock. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25–29, May 2000.

[46] A. Subramanian, P. Tamayo, V. K. Mootha, S. Mukherjee, B. L. Ebert, M. A. Gillette, A. Paulovich, S. L. Pomeroy, T. R. Golub, E. S. Lander, and J. P. Mesirov. Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences*, 102(43):15545–15550, September 2005.

[47] R. A. Fisher. The logic of inductive inference. *Journal of the Royal Statistical Society*, 98(1):39, 1935.

[48] Jing Cao and Song Zhang. A bayesian extension of the hypergeometric test for functional enrichment analysis. *Biometrics*, 70(1):84–94, December 2013.

[49] Jean Fred Fontaine and Miguel A Andrade-Navarro. Gene set to diseases (GS2d): disease enrichment analysis on human gene sets with literature data. *Genomics and Computational Biology*, 2(1):33, October 2016.

[50] N. Smirnov. Table for estimating the goodness of fit of empirical distributions. *The Annals of Mathematical Statistics*, 19(2):279–281, June 1948.

[51] Sonja Hänzelmann, Robert Castelo, and Justin Guinney. GSVA: gene set variation analysis for microarray and RNA-seq data. *BMC Bioinformatics*, 14(1):7, 2013.

[52] J. Javier Diaz-Mejia, Elaine C. Meng, Alexander R. Pico, Sonya A. MacParland, Troy Ketela, Trevor J. Pugh, Gary D. Bader, and John H. Morris. Evaluation of methods to assign cell type labels to cell clusters from single-cell RNA-sequencing data. *bioRxiv*, page 562082, February 2019.

[53] Zhong Wang, Mark Gerstein, and Michael Snyder. RNA-seq: a revolutionary tool for transcriptomics. *Nature Reviews Genetics*, 10(1):57–63, January 2009.

[54] Jeffrey A. Martin and Zhong Wang. Next-generation transcriptome assembly. *Nature Reviews Genetics*, 12(10):671–682, September 2011.

[55] Bruce Alberts, Karen Hopkin, Alexander D. Johnson, David Morgan, Martin Raff, Keith Roberts, and Peter Walter. *Essential Cell Biology (Fifth Edition)*. W. W. Norton & Company, 2019.

[56] Sabina Leonelli and Rachel A. Ankeny. What makes a model organism? *Endeavour*, 37(4):209–212, December 2013.

[57] Gordon Robertson, Jacqueline Schein, Readman Chiu, Richard Corbett, Matthew Field, Shaun D Jackman, Karen Mungall, Sam Lee, Hisanaga Mark Okada, Jenny Q Qian, Malachi Griffith, Anthony Raymond, Nina Thiessen, Timothee Cezard, Yaron S Butterfield, Richard Newsome, Simon K Chan, Rong She, Richard Varhol, Baljit Kamoh, Anna-Liisa Prabhu, Angela Tam, YongJun Zhao, Richard A Moore, Martin Hirst, Marco A Marra, Steven J M Jones, Pamela A Hoodless, and Inanc Birol. De novo assembly and analysis of RNA-seq data. *Nature Methods*, 7(11):909–912, October 2010.

[58] Manfred G Grabherr, Brian J Haas, Moran Yassour, Joshua Z Levin, Dawn A Thompson, Ido Amit, Xian Adiconis, Lin Fan, Raktima Raychowdhury, Qiandong Zeng, Zehua Chen, Evan Mauceli, Nir Hacohen, Andreas Gnirke, Nicholas Rhind, Federica di Palma, Bruce W Birren, Chad Nusbaum, Kerstin Lindblad-Toh, Nir Friedman, and Aviv Regev. Full-length transcriptome assembly from RNA-seq data without a reference genome. *Nature Biotechnology*, 29(7):644–652, May 2011.

[59] Marcel H. Schulz, Daniel R. Zerbino, Martin Vingron, and Ewan Birney. Oases: robust de novo RNA-seq assembly across the dynamic range of expression levels. *Bioinformatics*, 28(8):1086–1092, February 2012.

[60] Y. Xie, G. Wu, J. Tang, R. Luo, J. Patterson, S. Liu, W. Huang, G. He, S. Gu, S. Li, X. Zhou, T.-W. Lam, Y. Li, X. Xu, G. K.-S. Wong, and J. Wang. SOAPdenovo-trans: de novo transcriptome assembly with short RNA-seq reads. *Bioinformatics*, 30(12):1660–1666, February 2014.

[61] Mitchell Track changes is on Add comment Recompile 26 Guttman, Manuel Garber, Joshua Z. Levin, Julie Donaghey, James Robinson, Xian Adiconis, Lin Fan, Magdalena J. Koziol, Andreas Gnirke, Chad Nusbaum, John L. Rinn, Eric S. Lander, and Aviv Regev. Ab initio reconstruction of cell type-specific transcriptomes in mouse

reveals the conserved multi-exonic structure of lincrnas. *Nat Biotech*, 28(5):503–510, May 2010.

[62] Lin Liu, Lin Tang, Wen Dong, Shaowen Yao, and Wei Zhou. An overview of topic modeling and its current applications in bioinformatics. *SpringerPlus*, 5(1), September 2016.

[63] Li Song, Sarven Sabunciyan, and Liliana Florea. CLASS2: accurate and efficient splice variant annotation from RNA-seq reads. *Nucleic Acids Research*, 44(10):e98–e98, mar 2016.

[64] Ruolin Liu and Julie Dickerson. Strawberry: Fast and accurate genome-guided transcript reconstruction and quantification from RNA-seq. *PLOS Computational Biology*, 13(11):e1005851, November 2017.

[65] Tamara Steijger, Josep F. Abril, Par G. Engstrom, Felix Kokocinski, The RGASP Consortium, Tim J. Hubbard, Roderic Guigo, Jennifer Harrow, and Paul Bertone. Assessment of transcript reconstruction methods for rna-seq. *Nat Meth*, 10(12):1177–1184, Dec 2013. Analysis.

[66] Katharina E. Hayer, Angel Pizarro, Nicholas F. Lahens, John B. Hogenesch, and Gregory R. Grant. Benchmark analysis of algorithms for determining and quantifying full-length mRNA splice forms from RNA-seq data. *Bioinformatics*, page btv488, sep 2015.

[67] A. Dobin, C. A. Davis, F. Schlesinger, J. Drenkow, C. Zaleski, S. Jha, P. Batut, M. Chaisson, and T. R. Gingeras. STAR: ultrafast universal RNA-seq aligner. *Bioinformatics*, 2012.

[68] Daehwan Kim, Geo Pertea, Cole Trapnell, Harold Pimentel, Ryan Kelley, and Steven L Salzberg. TopHat2: accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions. *Genome Biology*, 14(4):R36, 2013.

[69] Daehwan Kim, Ben Langmead, and Steven L. Salzberg. Hisat: a fast spliced aligner with low memory requirements. *Nat Meth*, 12(4):357–360, Apr 2015. Article.

[70] T. D. Wu and C. K. Watanabe. GMAP: a genomic mapping and alignment program for mRNA and EST sequences. *Bioinformatics*, 21(9):1859–1875, February 2005.

[71] Heng Li. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*, 34(18):3094–3100, May 2018.

[72] Kangueane P. Sakharkar MK, Chow VT. Distributions of exons and introns in the human genome. *In Silico Biol.*, 4:387–93, 2004.

[73] K. F. Au, V. Sebastiano, P. T. Afshar, J. D. Durruthy, L. Lee, B. A. Williams, H. van Bakel, E. E. Schadt, R. A. Reijo-Pera, J. G. Underwood, and W. H. Wong. Characterization of the human ESC transcriptome by hybrid sequencing. *Proc. Natl. Acad. Sci. U.S.A.*, 110(50):E4821–4830, Dec 2013.

[74] H. Tilgner, F. Grubert, D. Sharon, and M. P. Snyder. Defining a personal, allele-specific, and single-molecule long-read transcriptome. *Proc. Natl. Acad. Sci. U.S.A.*, 111(27):9869–9874, Jul 2014.

[75] Krešimir Križanović, Amina Echchiki, Julien Roux, and Mile Šikić. Evaluation of tools for long read RNA-seq splice-aware alignment. *Bioinformatics*, 34(5):748–754, October 2017.

[76] Mark A DePristo, Eric Banks, Ryan Poplin, Kiran V Garimella, Jared R Maguire, Christopher Hartl, Anthony A Philippakis, Guillermo del Angel, Manuel A Rivas, Matt Hanna, Aaron McKenna, Tim J Fennell, Andrew M Kernytsky, Andrey Y Sivachenko, Kristian Cibulskis, Stacey B Gabriel, David Altshuler, and Mark J Daly. A framework for variation discovery and genotyping using next-generation DNA sequencing data. *Nature Genetics*, 43(5):491–498, April 2011.

[77] Elizabeth T Cirulli, Abanish Singh, Kevin V Shianna, Dongliang Ge, Jason P Smith, Jessica M Maia, Erin L Heinzen, James J Goedert, and David B Goldstein. Screening the human exome: a comparison of whole genome and whole transcriptome sequencing. *Genome Biology*, 11(5):R57, 2010.

[78] Justin M. Zook, Brad Chapman, Jason Wang, David Mittelman, Oliver Hofmann, Winston Hide, and Marc Salit. Integrating human sequence data sets provides a resource of benchmark snp and indel genotype calls. *Nat Biotech*, 32(3):246–251, Mar 2014. Computational Biology.

[79] Daniel F Simola and Junhyong Kim. Sniper: improved SNP discovery by multiply mapping deep sequenced reads. *Genome Biology*, 12(6):R55, 2011.

[80] Kristal Curtis, Ameet Talwalkar, Matei Zaharia, Armando Fox, and David A. Patterson. *SiRen: Leveraging Similar Regions for Efficient and Accurate Variant Calling*, 2015. `http://www.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-159.html`.

[81] Maryam Ghareghani, Seyed Abolfazl Motahari, Shahram Khazaei, and Mostafa Tavassolipour. Gw-call: Accurate genome-wide variant caller. *bioRxiv*, 2016.

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

[83] H. Li, J. Ruan, and R. Durbin. Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome Research*, 18(11):1851–1858, nov 2008.

[84] The SAM/BAM Format Specification Working Group. *Sequence Alignment/Map Format Specifiation*. `https://samtools.github.io/hts-specs/SAMv1.pdf`.

[85] Wei Li. *RNASeqReadSimulator: A Simple RNA-Seq Read Simulator*. `http://alumni.cs.ucr.edu/~liw/rnaseqreadsimulator.html`.

[86] The Genome Reference Consortium. *Human Genome Assembly GRCh37*. `https://www.ncbi.nlm.nih.gov/grc/human/data?asm=GRCh37`.

[87] UCSC Genome Informatics Group. *UCSC Genome Browser*. `https://genome.ucsc.edu/cgi-bin/hgTables`.

[88] Broad Institute. *GATK Best Practices workflow for SNP and indel calling on RNAseq data*. `https://software.broadinstitute.org/gatk/guide/article?id=3891`.

[89] Amie J. Radenbaugh, Singer Ma, Adam Ewing, Joshua M. Stuart, Eric A. Collisson, Jingchun Zhu, and David Haussler. RADIA: RNA and DNA integrated analysis for somatic mutation detection. *PLoS ONE*, 9(11):e111516, nov 2014.

[90] Lisa Neums, Seiji Suenaga, Peter Beyerlein, Sara Anders, Devin Koestler, Andrea Mariani, and Jeremy Chien. VaDiR: an integrated approach to variant detection in RNA. *GigaScience*, 7(2), dec 2017.

[91] NIH National Cancer Institute. Using tcga data, resources, and materials, 2019. `https://www.cancer.gov/about-nci/organization/ccg/research/structural-genomics/tcga/using-tcga`, Last accessed on 2019-04-20.

[92] Grace X. Y. Zheng, Jessica M. Terry, Phillip Belgrader, Paul Ryvkin, Zachary W. Bent, Ryan Wilson, Solongo B. Ziraldo, Tobias D. Wheeler, Geoff P. McDermott, Junjie Zhu, Mark T. Gregory, Joe Shuga, Luz Montesclaros, Jason G. Underwood, Donald A. Masquelier, Stefanie Y. Nishimura, Michael Schnall-Levin, Paul W. Wyatt, Christopher M. Hindson, Rajiv Bharadwaj, Alexander Wong, Kevin D. Ness, Lan W. Beppu, H. Joachim Deeg, Christopher McFarland, Keith R. Loeb, William J. Valente,

Nolan G. Ericson, Emily A. Stevens, Jerald P. Radich, Tarjei S. Mikkelsen, Benjamin J. Hindson, and Jason H. Bielas. Massively parallel digital transcriptional profiling of single cells. *Nature Communications*, 8:14049, jan 2017.

[93] Andrew Butler, Paul Hoffman, Peter Smibert, Efthymia Papalexi, and Rahul Satija. Integrating single-cell transcriptomic data across different conditions, technologies, and species. *Nature Biotechnology*, 36(5):411–420, April 2018.

[94] F. Alexander Wolf, Philipp Angerer, and Fabian J. Theis. SCANPY: large-scale single-cell gene expression data analysis. *Genome Biology*, 19:15, February 2018.

[95] David Lhnemann, Johannes Kster, Ewa Szczurek, Davis J. McCarthy, Stephanie C. Hicks, Mark D. Robinson, Catalina A. Vallejos, Kieran R. Campbell, Niko Beerenwinkel, Ahmed Mahfouz, Luca Pinello, Pavel Skums, Alexandros Stamatakis, Camille Stephan-Otto Attolini, Samuel Aparicio, Jasmijn Baaijens, Marleen Balvert, Buys de Barbanson, Antonio Cappuccio, Giacomo Corleone, Bas E. Dutilh, Maria Florescu, Victor Guryev, Rens Holmer, Katharina Jahn, Thamar Jessurun Lobo, Emma M. Keizer, Indu Khatri, Szymon M. Kielbasa, Jan O. Korbel, Alexey M. Kozlov, Tzu-Hao Kuo, Boudewijn P.F. Lelieveldt, Ion I. Mandoiu, John C. Marioni, Tobias Marschall, Felix Mlder, Amir Niknejad, Lukasz Raczkowski, Marcel Reinders, Jeroen de Ridder, Antoine-Emmanuel Saliba, Antonios Somarakis, Oliver Stegle, Fabian J. Theis, Huan Yang, Alex Zelikovsky, Alice C. McHardy, Benjamin J. Raphael, Sohrab P. Shah, and Alexander Schnhuth. Eleven grand challenges in single-cell data science. *Genome Biology*, 21(1):31, February 2020.

[96] Edward Y Chen, Christopher M Tan, Yan Kou, Qiaonan Duan, Zichen Wang, Gabriela Meirelles, Neil R Clark, and Avi Ma'ayan. Enrichr: interactive and collaborative HTML5 gene list enrichment analysis tool. *BMC Bioinformatics*, 14(1):128, 2013.

[97] Harald Stachelscheid, Stefanie Seltmann, Fritz Lekschas, Jean-Fred Fontaine, Nancy Mah, Mariana Neves, Miguel A. Andrade-Navarro, Ulf Leser, and Andreas Kurtz. CellFinder: a cell data repository. *Nucleic Acids Research*, 42(D1):D950–D958, December 2013.

[98] Xinxin Zhang, Yujia Lan, Jinyuan Xu, Fei Quan, Erjie Zhao, Chunyu Deng, Tao Luo, Liwen Xu, Gaoming Liao, Min Yan, Yanyan Ping, Feng Li, Aiai Shi, Jing Bai, Tingting Zhao, Xia Li, and Yun Xiao. CellMarker: a manually curated resource of cell markers in human and mouse. *Nucleic Acids Research*, 2018.

[99] Oscar Franzén, Li-Ming Gan, and Johan L M Björkegren. PanglaoDB: a web server for exploration of mouse and human single-cell RNA sequencing data. *Database*, 2019, January 2019.

[100] Hannah A. Pliner, Jay Shendure, and Cole Trapnell. Supervised classification enables rapid annotation of cell atlases. *bioRxiv*, page 538652, February 2019.

[101] Tamim Abdelaal, Lieke Michielsen, Davy Cats, Dylan Hoogduin, Hailiang Mei, Marcel J. T. Reinders, and Ahmed Mahfouz. A comparison of automatic cell identification methods for single-cell RNA sequencing data. *Genome Biology*, 20(1), September 2019.

[102] Donna Maglott, Jim Ostell, Kim D. Pruitt, and Tatiana Tatusova. Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Research*, 35(Database issue):D26–D31, January 2007.

[103] MEDLINE Indexing Online Training Course, 2015.

[104] Medical Subject Headings, 2019.

[105] Wikipedia. tf-idf - wikipedia, 2020. https://en.wikipedia.org/wiki/Tfidf, Last accessed on 2020-02-21.

[106] The Tabula Muris Consortium. Single-cell transcriptomics of 20 mouse organs creates a Tabula Muris. *Nature*, 562(7727):367, October 2018.

[107] Xiaoping Han, Renying Wang, Yincong Zhou, Lijiang Fei, Huiyu Sun, Shujing Lai, Assieh Saadatpour, Zimin Zhou, Haide Chen, Fang Ye, Daosheng Huang, Yang Xu, Wentao Huang, Mengmeng Jiang, Xinyi Jiang, Jie Mao, Yao Chen, Chenyu Lu, Jin Xie, Qun Fang, Yibin Wang, Rui Yue, Tiefeng Li, He Huang, Stuart H. Orkin, Guo-Cheng Yuan, Ming Chen, and Guoji Guo. Mapping the Mouse Cell Atlas by Microwell-Seq. *Cell*, 172(5):1091–1107.e17, February 2018.

[108] Wikipedia. Monoblast - wikipedia, 2020. `https://en.wikipedia.org/wiki/Monoblast`, Last accessed on 2020-01-20.

[109] Wikipedia. Promonocyte - wikipedia, 2020. `https://en.wikipedia.org/wiki/Promonocyte`, Last accessed on 2020-01-20.

[110] Wikipedia. Granulocyte - wikipedia, 2020. `https://en.wikipedia.org/wiki/Granulocyte`, Last accessed on 2020-01-18.

[111] Amir Alavi, Matthew Ruffalo, Aiyappa Parvangada, Zhilin Huang, and Ziv Bar-Joseph. A web server for comparative analysis of single-cell RNA-seq data. *Nature Communications*, 9(1):4768, November 2018.

[112] A. I. Su, T. Wiltshire, S. Batalov, H. Lapp, K. A. Ching, D. Block, J. Zhang, R. Soden, M. Hayakawa, G. Kreiman, M. P. Cooke, J. R. Walker, and J. B. Hogenesch. A gene atlas of the mouse and human protein-encoding transcriptomes. *Proceedings of the National Academy of Sciences*, 101(16):6062–6067, April 2004.

[113] Grace X. Y. Zheng, Jessica M. Terry, Phillip Belgrader, Paul Ryvkin, Zachary W. Bent, Ryan Wilson, Solongo B. Ziraldo, Tobias D. Wheeler, Geoff P. McDermott, Junjie Zhu, Mark T. Gregory, Joe Shuga, Luz Montesclaros, Jason G. Underwood, Donald A. Masquelier, Stefanie Y. Nishimura, Michael Schnall-Levin, Paul W. Wyatt, Christopher M. Hindson, Rajiv Bharadwaj, Alexander Wong, Kevin D. Ness, Lan W. Beppu, H. Joachim Deeg, Christopher McFarland, Keith R. Loeb, William J. Valente, Nolan G. Ericson, Emily A. Stevens, Jerald P. Radich, Tarjei S. Mikkelsen, Benjamin J. Hindson, and Jason H. Bielas. Massively parallel digital transcriptional profiling of single cells. *Nature Communications*, 8:14049, January 2017.

[114] Alexander B. Rosenberg, Charles M. Roco, Richard A. Muscat, Anna Kuchina, Paul Sample, Zizhen Yao, Lucas Gray, David J. Peeler, Sumit Mukherjee, Wei Chen, Suzie H. Pun, Drew L. Sellers, Bosiljka Tasic, and Georg Seelig. Single-cell profiling of the developing mouse brain and spinal cord with split-pool barcoding. *Science*, page eaam8999, March 2018.

[115] S. Siena, M. Bregni, B. Brando, F. Ravagnani, G. Bonadonna, and A. M. Gianni. Circulation of CD34+ hematopoietic stem cells in the peripheral blood of high-dose cyclophosphamide-treated patients: enhancement by intravenous recombinant human granulocyte-macrophage colony-stimulating factor. *Blood*, 74(6):1905–1914, November 1989.

[116] Sueli Marques, Amit Zeisel, Simone Codeluppi, David van Bruggen, Ana Mendanha Falco, Lin Xiao, Huiliang Li, Martin Hring, Hannah Hochgerner, Roman A. Romanov, Daniel Gyllborg, Ana B. Muoz-Manchado, Gioele La Manno, Peter Lnnerberg, Elisa M. Floriddia, Fatemah Rezayee, Patrik Ernfors, Ernest Arenas, Jens Hjerling-Leffler, Tibor Harkany, William D. Richardson, Sten Linnarsson, and Gonalo Castelo-Branco. Oligodendrocyte heterogeneity in the mouse juvenile and adult central nervous system. *Science*, 352(6291):1326–1329, June 2016. Publisher: American Association for the Advancement of Science Section: Report.

[117] Dhaivat Joshi, Shunfu Mao, Sreeram Kannan, and Suhas Diggavi. QAlign: Aligning nanopore reads accurately using current-level modeling. December 2019.

[118] Shunfu Mao, Yihan Jiang, Edwin Basil Mathew, and Sreeram Kannan. BOAssembler: A bayesian optimization framework to improve RNA-seq assembly performance. In *Al-*

gorithms for Computational Biology, pages 188–197. Springer International Publishing, 2020.

[119] Edward Y. Chen, Christopher M. Tan, Yan Kou, Qiaonan Duan, Zichen Wang, Gabriela Vaz Meirelles, Neil R. Clark, and Avi Maayan. Enrichr: interactive and collaborative HTML5 gene list enrichment analysis tool. *BMC Bioinformatics*, 14:128, April 2013.

[120] Christopher J Mungall, Carlo Torniai, Georgios V Gkoutos, Suzanna E Lewis, and Melissa A Haendel. Uberon, an integrative multi-species anatomy ontology. *Genome Biology*, 13(1):R5, 2012.