RNA-BLOOM: *DE NOVO* RNA-SEQ ASSEMBLY WITH BLOOM FILTERS

by

Ka Ming Nip

B.Sc., The University of British Columbia, 2011

B.Sc., The University of British Columbia, 2007

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

(Bioinformatics)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

August 2017

Abstract

High-throughput RNA sequencing (RNA-seq) is primarily used in measuring gene expression, quantifying transcript abundance, and building reference transcriptomes. Without bias from a reference sequence, *de novo* RNA-seq assembly is particularly useful for building new reference transcriptomes, detecting fusion genes, and discovering novel spliced transcripts. This is a challenging problem, and to address it at least eight approaches, including Trans-ABySS and Trinity, were developed within the past decade. For instance, using Trinity and 12 CPUs, it takes approximately one and a half day to assemble a human RNA-seq sample of over 100 million read pairs and requires up to 80 GB of memory. While the high memory usage typical of *de novo* RNA-seq assemblers may be alleviated by distributed computing, access to a high-performance computing environment is a requirement that may be limiting for smaller labs. In my thesis, I present a novel *de novo* RNA-seq assembler, "RNA-Bloom," which utilizes compact data structures based on Bloom filters for the storage of *k*-mer counts and the de Bruijn graph in memory. Compared to Trans-ABySS and Trinity, RNA-Bloom can assemble a human transcriptome with comparable accuracy using nearly half as much memory and half the wall-clock time with 12 threads.

Lay Summary

High-throughput RNA sequencing (RNA-seq) is used for the identification and quantification of transcripts (RNA molecules) in cells, and it has a variety of applications in genomics research. RNA-seq assembly is the process of reconstructing full-length transcripts from short but highly accurate sequences. For well-studied species, transcript reconstruction can be guided by a reference genome. In the absence of a suitable reference genome, transcripts can be reconstructed with *de novo* assembly methods. Because the cost of sequencing experiments has drastically decreased since its introduction, RNA-seq data becomes much more readily available than previous years, resulting in a demand for fast and resource-efficient RNA-seq analysis methods. Although multiple algorithms for *de novo* RNA-seq assembly were developed within the past decade, the current state-of-the-art methods are slow and require sizable amount of computing resources. I describe in my thesis a competitive, fast, and lightweight approach that uses a compact data structure called Bloom filter.

Preface

The concept and design of my research was conceived by my supervisor, Dr. Inanc Birol, and myself. All software programming and analyses were carried out by myself with very helpful discussions with Mr. Justin Chu, Mr. Readman Chiu, Dr. Hamid Mohamadi, and Mr. Ben Vandervalk. Mr. Readman Chiu created Figure 3.1 and provided the fusion detection results for Tophat-Fusion in Chapter 3.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

bp	Base pair
c-Bf	Counting Bloom filter
CPU	Central processing unit
DBG	De Bruijn graph
DBG-Bf	De Bruijn graph Bloom filter
DNA	Deoxyribonucleic acid
ENCODE	Encyclopedia of DNA Elements
FPR	False-positive rate
Gbp	Giga-base pair (one billion base pairs)
GB	Gigabyte (1,073,741,824 bytes)
GRCh38	Genome Reference Consortium Human Build 38
k-mer	Sequence of *k* letters
MCF-7	A breast cancer cell line
Q1	First quartile
RNA	Ribonucleic acid
RNA-seq	High-throughput RNA sequencing
RT-PCR	Reverse-transcription polymerase chain reaction
TB	Terabyte (1,099,511,627,776 bytes)
UHRR	Universal Human Reference RNA

List of Genes

ABL1	ABL proto-oncogene 1, non-receptor tyrosine kinase
APPBP2	amyloid beta precursor protein binding protein 2
ARL17A	ADP ribosylation factor like GTPase 17A
ARFGEF2	ADP Ribosylation Factor Guanine Nucleotide Exchange Factor 2
BAG6	BCL2 associated athanogene 6
BCAS3	BCAS3, microtubule associated cell migration factor
BCAS4	breast carcinoma amplified sequence 4
BCR	BCR, RhoGEF and GTPase activating protein
CA4	carbonic anhydrase 4
GAS6	growth arrest specific 6
FGFR1	fibroblast growth factor receptor 1
HOMEZ	homeobox and leucine zipper encoding
KANSL1	KAT8 regulatory NSL complex subunit 1
MYH6	myosin heavy chain 6
NSD3	nuclear receptor binding SET domain protein 3
NUP214	nucleoporin 214
RASA3	RAS p21 protein activator 3
RPS6KB1	ribosomal protein S6 kinase B1
SLC44A4	solute carrier family 44 member 4
SULF2	Sulfatase 2

TANC2	tetratricopeptide repeat, ankyrin repeat and coiled-coil containing 2
USP32	ubiquitin specific peptidase 32
VMP1	vacuole membrane protein 1
XKR3	XK related 3

Acknowledgements

First, I would like to thank Dr. Inanc Birol for providing me the opportunity to develop my research skills and communication skills as a graduate student under his guidance. I also would like to thank him for his patience, his constructive criticisms, and providing me multiple opportunities to present my research at various international conferences.

I would like to thank my thesis committee members, Dr. Sara Mostafavi and Dr. Wyeth Wasserman, for their very valuable feedbacks and time throughout my degree.

My sincerest gratitude goes to the Canadian Institutes of Health Research, for providing financial support for my thesis project.

I would like to thank all members of the Bioinformatics Technology Lab at the Genome Sciences Centre, especially Mr. Readman Chiu, Mr. Justin Chu, Mr. Ben Vandervalk, and Dr. Hamid Mohamadi for their valuable discussions on various aspects of my thesis project.

Finally, I would like to thank my parents and my sister for their time, encouragement, and unconditional support for me throughout my life.

Chapter 1: Introduction

1.1 High throughput RNA sequencing

High-throughput RNA sequencing (RNA-seq) is used for the identification and quantification of transcripts in cells. RNA-seq enables the study of a spectrum of RNA species and it has a variety of applications in genomics research such as, gene expression analysis, building new reference transcriptomes, and splice variant detection and discovery [1]. Moreover, RNA-seq has the potential to be translated into clinical diagnostics due to its ability to profile transcript expression and detect aberrant transcription and gene fusions in human diseases [2].

RNA sequencing has been dominated by Illumina's sequencing technologies, due to their highly accurate reads with a small percentage of substitution errors, their relatively high throughput, and the ability to identify and quantify transcripts from a single sequencing experiment [3]. Consequently, "RNA-seq" has been typically referred to Illumina's RNA sequencing technologies. Although long-read sequencing technologies from Pacific Biosciences and Oxford Nanopore Technologies have shown premise in transcriptome analyses [4-6], they have a much higher sequencing error rate, more indel errors, higher cost, and lower throughput than Illumina sequencing technologies. Since the cost of Illumina sequencing experiments has drastically decreased over the years, RNA-seq data becomes much more readily available than previous years, resulting in a demand for resource-efficient RNA-seq analysis software.

1.2 *De novo* RNA-seq assembly

RNA-seq assembly is the process of reconstructing full-length transcripts from short but highly accurate RNA-seq read sequences ranging from 100 to 300 bp. For well-studied species, transcript reconstruction can be guided by the alignment of read sequences against the reference genome. In the absence of a suitable reference genome, transcripts may still be reconstructed with *de novo* RNA-seq assembly. Without bias from a reference sequence, *de novo* RNA-seq assembly is particularly useful for building reference transcriptomes [7-10] and detecting gene fusions [11-14]. Consequently, *de novo* RNA-seq assembly has been applied for structural variant detection in cancer cohort studies [11-14] and personalized oncogenomic profiling [15].

De novo RNA-seq assembly is a challenging problem and at least eight algorithms [16-23] were developed within the past decade (Table 1.1). Trans-ABySS, Oases, SOAP-denovo-Trans, IDBA-Tran, and SSP use *de novo* genome assemblers for building the initial set of contiguous sequences, which are then further connected to form transcript sequences. Trinity clusters overlapping reads into closely-related gene groups where transcript sequences are derived. Bridger and BinPacker produce transcript sequences based on splice graphs derived from RNA-seq reads and they borrow ideas from Trinity, SOAP-denovo-Trans, and Cufflinks [24], which is a reference-based RNA-seq assembler. However, the performance of these methods varies.

In a recent comprehensive benchmarking study [25], Trans-ABySS [16] was ranked the best in full-length transcript reconstruction and gene coverage. Trinity [17], another assembler, was also ranked relatively high and had received several updates since the benchmarking study was published. However, these two state-of-the-art methods require sizable amount of computational resources (Table 1.2) and thus are typically run on specialized hardware. For instance, using Trinity and 12 CPUs, it takes approximately one and a half day to assemble a human RNA-seq sample of over 100 million read pairs and requires up to 80 GB of memory. While the high memory usage typical of *de novo* RNA-seq assemblers may be alleviated by distributed computing, access to a high-performance computing environment is a requirement that may be limiting for smaller labs.

Table 1.1 *De novo* RNA-seq assemblers published within the past decade.

Assembler	Year Published	Reference
Trans-ABySS	2010	[16]
Trinity	2011	[17]
Oases	2012	[18]
IDBA-Tran	2013	[19]
SSP	2013	[20]
SOAP-denovo-Trans	2014	[21]
Bridger	2015	[22]
BinPacker	2016	[23]

Table 1.2 Performance of current state-of-the-art methods.

Assemblers were run in 12 threads. The RNA-seq dataset is an ENCODE MCF-7 sample (ENCLB555AVR) that has 148 million strand-specific read pairs of length 2 x 100 bp.

Assembler	Peak Memory (GB)	Wall-clock Time (hours)
Trans-ABySS	32	23
Trinity	84	34

1.3 Inspirations for a lightweight *de novo* RNA-seq assembly algorithm

In this section, I describe recent advances in sequence assembly algorithms and other sequence analysis methods that provide insights for a fast and memory-efficient algorithm for *de novo* RNA-seq assembly.

1.3.1 Bloom filter de Bruijn graph

In a de Bruijn graph (DBG), the nodes represent *k*-mers (sequences of *k* letters) and the edges represent the overlap of $k - 1$ letters between *k*-mers. Overlapping reads and the sequences of their underlying DNA/RNA molecules may be represented by paths in a DBG, where *k*-mers are sub-sequences of reads. Hence, DBG has been widely adopted in *de novo* assembly algorithms for genomes, metagenomes, and transcriptomes [26].

A typical hash-table implementation of DBG for *de novo* sequence assembly, such as the one in ABySS [27], uses two-bit encoding to represent four bases, “A,” “C,” “G,” and “T,” hence requires a minimum of two bits in memory for every base in a *k*-mer. For $k = 28$, a practical hash table DBG representation of the human genome, which is over 3 Gbp in length, (assuming every 28-mer in the genome is unique) would require at least 21 GB of memory, where each unique *k*-mer occupies exactly 7 bytes. Since DBG representation of larger genomes and noisy experimental data would require even more memory, reducing the memory usage of DBG has been an active area of research for *de novo* sequence assembly [26].

A Bloom filter, which is a compact probabilistic data structure for set-membership tests [28], can be used as a memory-efficient representation of DBG for sequence assembly [29]. A Bloom filter is initialized as an empty binary array of zero-bits; select bits are set to denote the presence of set elements, where bit positions are assigned by one or more hash functions (Figure

1.1). To represent a DBG in a Bloom filter, k-mers are first stored as set-bits in the Bloom filter, and then the DBG is traversed from a given k-mer by testing each possible neighboring k-mer with the Bloom filter (Figure 1.2). However, Bloom filters may have false-positives due to hash collisions in assigning bit positions within the binary array. The probability of false-positives, also known as false-positive rate (FPR), for a Bloom filter can be estimated with the formula, $\left(1 - e^{-hn/m}\right)^h$, where *h* is the number of hash functions, *n* is the number of set elements stored in the Bloom filter, and m is the size of the Bloom filter. As the formula indicates, inserting more set elements always increases the FPR and increasing the Bloom filter size always decreases the FPR. Increasing the number of hash functions decreases the FPR to some extent before detrimental effects are observed. Essentially, the memory efficiency of a Bloom filter comes at the cost of false-positives.

Furthermore, Birol *et al.* recently introduced the concept of a counting Bloom filter for counting *k*-mers in *de novo* assembly applications [30]. The k-mer counts are represented compactly as 8-bit minifloats, which are exact from 0 to 16 but are non-exact from 17 up to 122,880. For comparison, the smallest exact representation of integers in this range would require a 32-bit integer data type, which uses four times more memory. Although higher counts are non-exact in their concept, Birol *et al.* showed that counts at different orders of magnitude rarely overlap. This property is especially important for RNA-seq assembly because transcript abundance has a dynamic range across six orders of magnitude in RNA-seq data.

Several methods have adapted Bloom filters for the *de novo* assembly of genomes and metagenomes [29, 31, 32], but Bloom filters remained to be explored in *de novo* RNA-seq assembly.

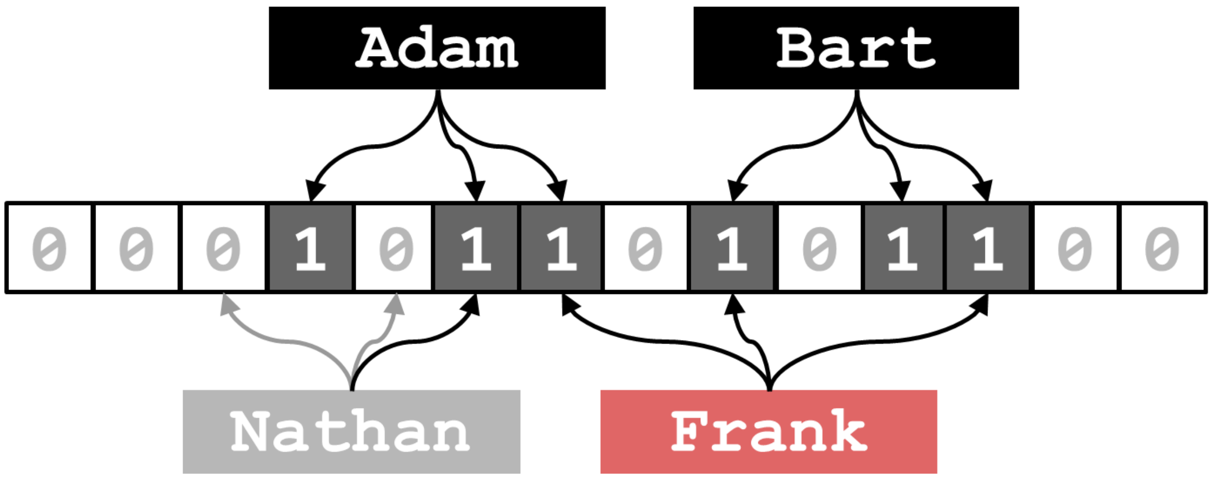


Figure 1.1 An example of Bloom filter.

This is an example of a Bloom filter, which uses 3 bits to store each item in the set containing only "Adam" and "Bart." "Nathan" does not exist in the set, as indicated by the presence of 0-bits at its hash locations. "Frank," which is not in the set, is a Bloom filter false-positive because the bits at its hash locations have been set previously for "Adam" and "Bart".

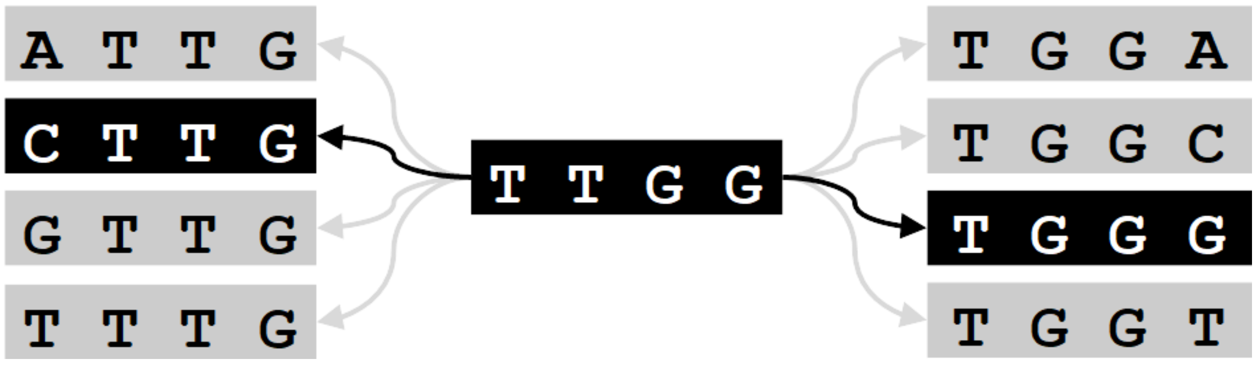


Figure 1.2 A Bloom filter representation of de Bruijn graph.

A *k*-mer has four possible neighbors on each side. Testing for the presence of each neighbor is equivalent to a set-membership test, which is the lookup operation of a Bloom filter.

1.3.2 Synthetic reads

The quality of *de novo* assembly usually improves with longer input reads because longer reads permits the use of longer *k*-mers, resulting in a smaller and simpler DBG. Several methods aim to generate synthetic reads by extending read sequences before assembly. Overlapping paired reads may be merged if there is sufficient overlap [33]. Non-overlapping paired reads can be joined with methods such as Konnector [34] and MaSuRCa [35], which aim to find a unique path in the DBG between each read pair. Since the fragment length for paired-end sequencing are much shorter than the underlying nucleotide sequence (eg. chromosome), joining paired reads is a much less challenging task than reconstructing the entire nucleotide sequence. Consequently, Konnector and MaSuRCa were instrumental to creating the draft assemblies of conifer tree genomes over 20 Gbp in length [36, 37]. Konnector uses a Bloom filter DBG in its implementation and thus has a much lower memory footprint than MaSuRCa. StringTie, a reference-guided RNA-seq assembler, has the option to include MaSuRCa's output reads for assembling transcripts [38]. However, existing *de novo* RNA-seq assemblers do not include synthetic read construction in their algorithms.

1.3.3 Alignment-free methods

Alignment of high-throughput sequencing reads is an important but relatively time-consuming step in methods for processing high-throughput sequencing data; reducing the runtime of this operation in genomics applications has been an active area of research [39]. Although *de novo* assembly algorithms, by definition, do not utilize alignment of reads against a reference genome or transcriptome, a typical *de novo* assembly algorithm such as ABySS still requires two rounds of read alignments against the assembled sequences for contiging and scaffolding. The current state-of-the-art *de novo* RNA-seq assemblers (Table 1.1) also rely on alignment of reads throughout the assembly process. An alignment-free assembly approach would likely have a much quicker runtime. To the best of my knowledge, LINKS is the only alignment-free method for scaffolding genome assemblies so far, and it uses a Bloom filter for the storage of *k*-mer pairs in long reads [40].

In recent years, several alignment-free methods have emerged for RNA-seq quantification [41-44], and their runtimes are over 100 times faster than the fastest alignment-based method. In the past, RNA-seq quantification methods, such as RSEM [45], deduce the transcript expression levels based on the alignment of reads against a reference genome or transcriptome. However, these alignment-free RNA-seq quantification methods are not *de novo* methods because they still rely on given reference sequences (and reference annotations in some cases).

1.4 Thesis objectives

Recent advances in high-throughput sequencing analyses methods have motivated the development of a fast and memory-efficient *de novo* RNA-seq assembler. Compared to the current state-of-the-art RNA-seq assemblers, my assembly algorithm, RNA-Bloom, was developed with the following objectives:

1. lower memory usage,
2. quicker runtime, and
3. similar or better accuracy.

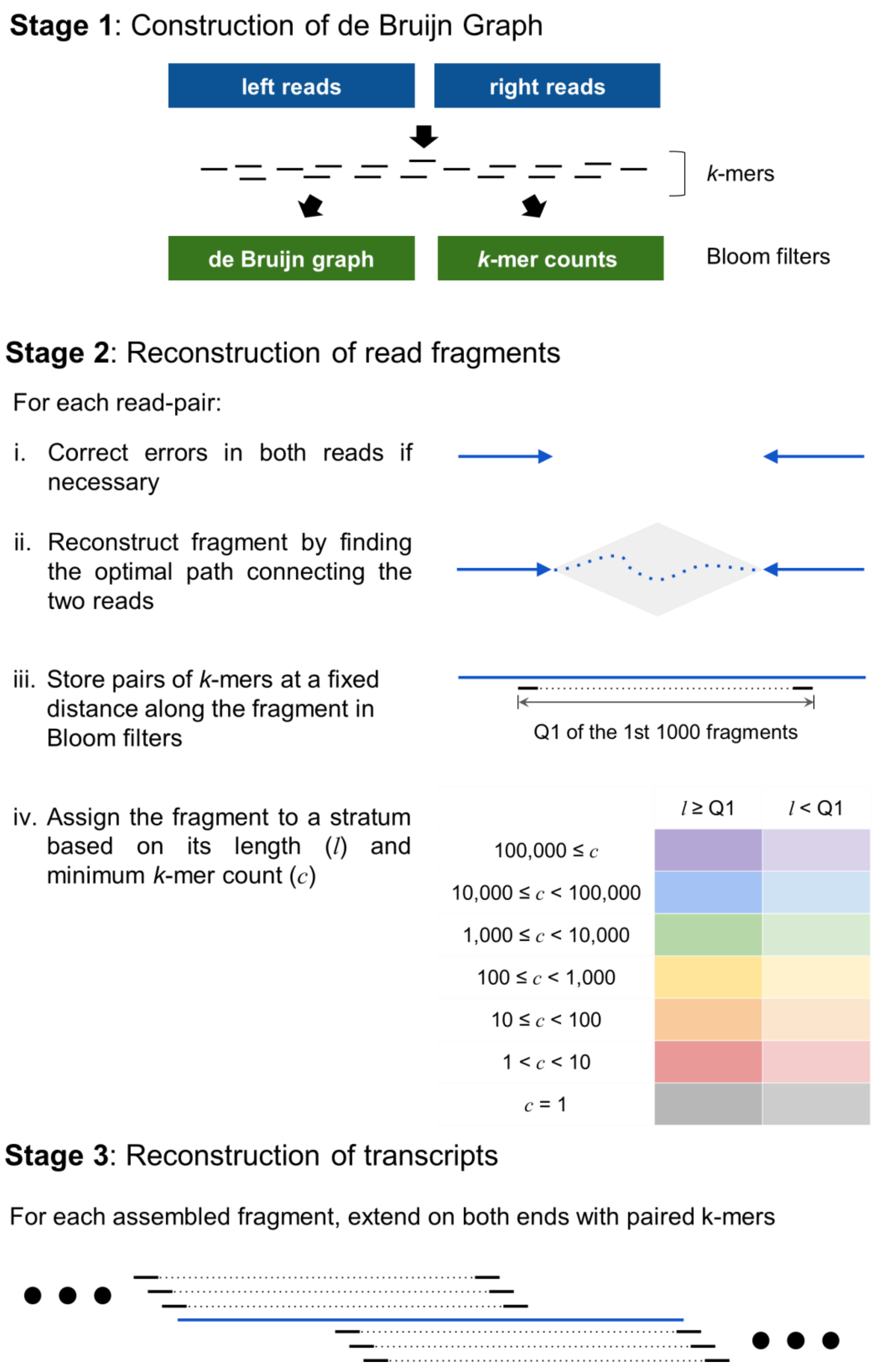
The following chapters illustrate how these objectives were achieved.

Chapter 2: Methods

RNA-Bloom consists of three stages: construction of de Bruijn graph, reconstruction of synthetic long reads representing sequencing library fragments, and reconstruction of transcripts. Each stage is described in the following sections. An overview of the workflow of RNA-Bloom is illustrated in Figure 2.1.

2.1 Stage 1: Construction of de Bruijn graph

In this stage, all sequencing reads are parsed and all *k*-mers containing only “A,” “C,” “G,” or “T” characters are loaded into a Bloom filter, representing an implicit DBG. The counts of *k*-mers are stored in a counting Bloom filter based on the concept discussed in the previous chapter. The count of each *k*-mer is incremented every time the *k*-mer is observed in the sequencing reads. To account for hash collisions, only the hash positions with the smallest value are incremented during update. Figure 2.2 illustrates how hash collisions in the c-Bf are handled. The DBG Bloom filter (DBG-Bf) and the counting Bloom filter (c-Bf) co-operate to reduce false-positives in each other. For example, a *k*-mer found in the DBG-Bf is deemed as “present” in the DBG only if it has a positive count in the c-Bf. Similarly, a non-zero count from the c-Bf is reported only if the *k*-mer is found in the DBG-Bf; otherwise, a zero-count is reported. To ensure quick update and lookup of Bloom filters, all *k*-mers are hashed with a custom hashing function based on an ultrafast nucleotide hashing algorithm called ntHash [46]. Once all sequencing reads have been parsed, the DBG-Bf and the c-Bf are ready for use in subsequent stages.



Stage 1: Construction of de Bruijn Graph

Stage 2: Reconstruction of read fragments

For each read-pair:

i. Correct errors in both reads if necessary

ii. Reconstruct fragment by finding the optimal path connecting the two reads

iii. Store pairs of *k*-mers at a fixed distance along the fragment in Bloom filters

iv. Assign the fragment to a stratum based on its length (l) and minimum *k*-mer count (c)

	$l \geq$ Q1	$l <$ Q1
$100{,}000 \leq c$	🟪	🟪
$10{,}000 \leq c < 100{,}000$	🟦	🟦
$1{,}000 \leq c < 10{,}000$	🟩	🟩
$100 \leq c < 1{,}000$	🟨	🟨
$10 \leq c < 100$	🟧	🟧
$1 < c < 10$	🟥	🟥
$c = 1$	⬜	⬜

Stage 3: Reconstruction of transcripts

For each assembled fragment, extend on both ends with paired k-mers

Figure 2.1 An overview of RNA-Bloom.

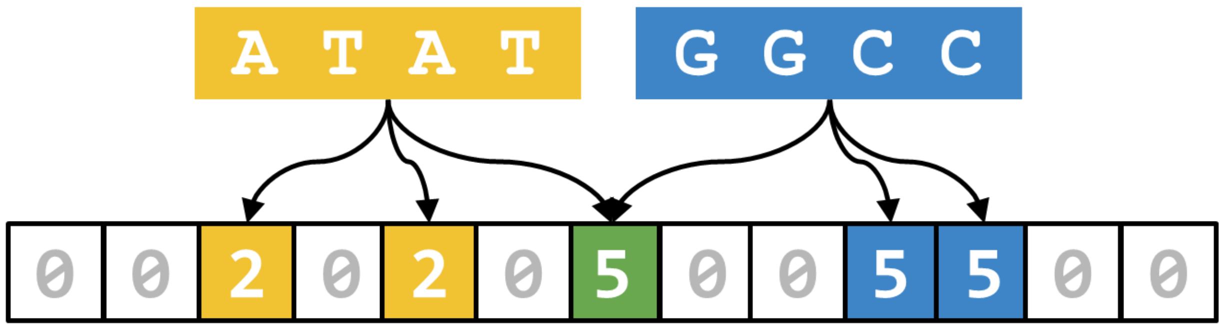


Figure 2.2 Hash collision in counting Bloom filter.

This counting Bloom filter uses three hash functions and it has a hash collision at the green cell. To account for hash collisions, only the minimum value of all three hash positions of a *k*-mer is reported or updated. Hence, "ATAT" would be reported to have a count of 2 despite "5" is stored at one of its hash positions. If the count for "ATAT" were incremented, the yellow cells would be updated to "3" and the green cell would remain untouched. However, if the yellow cells become "5," both yellow and green cells would be updated to "6" during an increment.

2.2 Stage 2: Reconstruction of read fragments

In this stage, paired-end reads are first corrected for errors and are then connected to reconstruct the read fragment. Pairs of *k*-mers at a fixed distance along each reconstructed fragment are stored for use in the next stage. Each fragment is assigned a bin according to its length and its lowest *k*-mer count.

2.2.1 Error correction of paired end reads

Since RNA-seq reads may contain errors due to sequencer miscalls, paired-end reads are corrected before they are connected to reconstruct the underlying fragment sequence. Sequencing errors can be corrected based on *k*-mer counts because sequencer miscalls tend to be rarer than the correct calls. Since transcript abundance has a range across six orders of magnitude, applying a global threshold on *k*-mer counts, as done in genome assembly [32], may remove low expressed transcripts and thus is not preferable for RNA-seq assembly.

In RNA-Bloom, sequencing errors are corrected based on an algorithm called Rcorrector [47], which is specifically designed for *de novo* correction of RNA-seq data. For each read in a read-pair, *k*-mer counts are first sorted in descending order and then a local threshold is set at the first sharp (over 50% by default) decrease (Figure 2.3). However, only the smaller of the two thresholds from both reads is used. This is done to account for the read-pair potentially spanning a low-expressed alternative isoform or paralogous transcript. *k*-mers with counts less than the local threshold are replaced with an alternative branch in the DBG with a higher median *k*-mer count while retaining a high percent sequence identity, if such a branch exists (Figure 2.4).

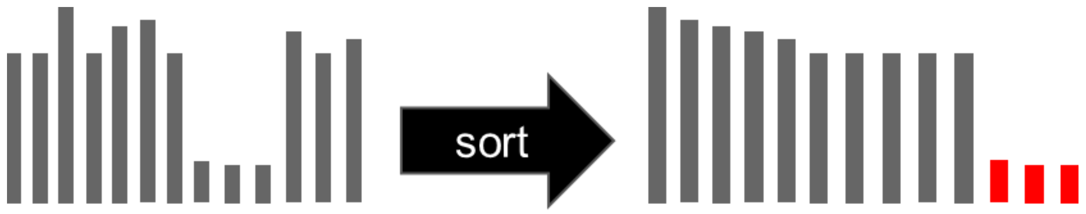


Figure 2.3 Marking candidate error *k*-mers using a count-based method.

k-mer counts within a read are sorted in descending order and the first sharp decrease determines a local threshold for the *k*-mer counts in the read. *k*-mers with counts below the threshold (shown in red) are potentially erroneous and are marked for inspection.

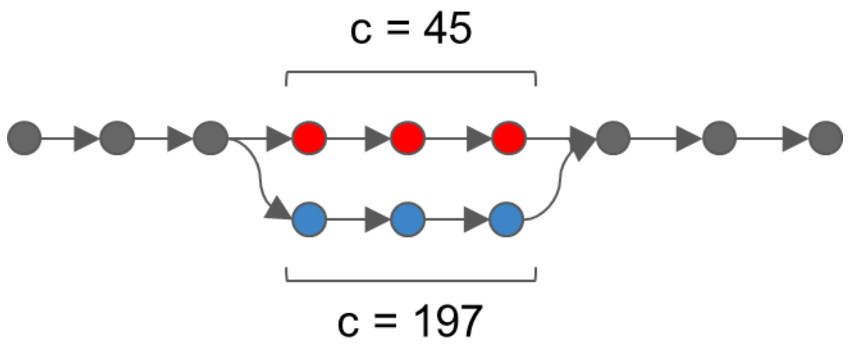


Figure 2.4 Replacing error *k*-mer candidates using a graph-based method.

Error *k*-mer candidates (shown in red) may be replaced by finding in the de Bruijn graph an alternative branch (shown in blue) with higher median *k*-mer count but high sequence identity and nearly identical length.

2.2.2 Connecting paired-end reads

For each read pair, the left and right reads may be connected by one or more paths in the DBG. The optimal path connecting the reads should be free from sequencing errors and should not introduce chimeras. In RNA-Bloom, this optimal path is approximated by extending the paired reads towards each other, while ignoring short dead-end branches and branches that fall below a local *k*-mer count threshold. This process begins at the right-most *k*-mer in the left read, extending to the right. Each step in the extension consists of a look-ahead, which prunes short dead-end branches assumed to be introduced by Bloom filter false-positives. The local *k*-mer count threshold is initially set to 10% of the minimum *k*-mer count in the read pair. As the extension progresses, the threshold is dynamically adjusted to 10% of the minimum *k*-mer count in the read pair and the currently traversed path to allow fluctuation in read depth. The extension terminates when either the left-most *k*-mer in the right read is reached, a dead-end is reached, the maximum search depth is reached, or an ambiguous branch is reached. If the reads are not connected at this point, another attempt is made by extending to the left, starting at the left-most *k*-mer in the right read. The extension of the second attempt terminates when either the right-most *k*-mer in the left read is reached, a *k*-mer in the extension path from the previous attempt is reached, a dead-end is reached, the maximum search depth is reached, or an ambiguous branch is reached. The process of connecting paired reads is illustrated in Figure 2.5.

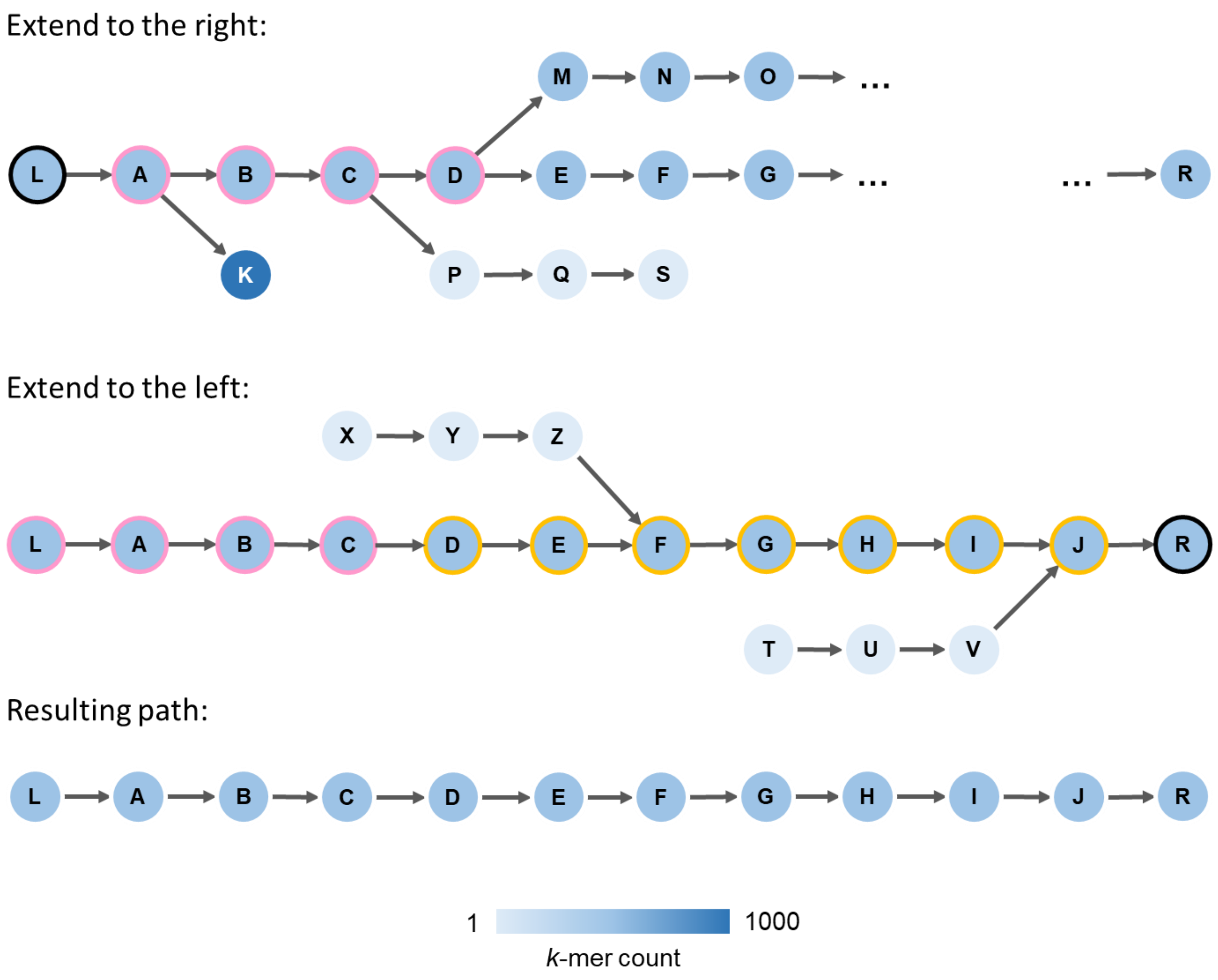


Figure 2.5 Connecting paired reads.

"L" is the right-most *k*-mer in the left read and "R" is the left-most *k*-mer in the right read. During extension to the right from "L," the short dead-end branch "K" and the branch with low *k*-mer count "P-Q-S" are ignored. The extension to the right (highlighted in pink) terminates at "D" because the look-ahead discovers more than one possible extension. Since "R" has not been reached, the second attempt of extension is performed in the opposite direction from "R." During the extension to the left from "R," the branches with low *k*-mer count, "V-U-T" and "Z-Y-X," are ignored. The extension to the left (highlighted in orange) terminates at "D," a *k*-mer from the previous extension path. As a result, both extension paths are joined together to connect "L" to "R."

2.2.3 Storage of *k*-mer pairs

The distribution of fragment lengths is inferred from the first 1,000 fragments assembled, and Q1 is set as the first quartile of this set. The sample size of 1,000 is arbitrary and can be set by the user. Pairs of *k*-mers at a fixed distance along fragments are stored for the assembly of transcript sequences in Stage 3. The distance between paired *k*-mers is set to Q1 – 2*k*. Assuming fragment lengths are normally distributed, approximately 75% of fragments would be longer than or equal to Q1 and thus are represented by *k*-mer pairs. On the other hand, the remaining 25% of fragments would not be represented by *k*-mer pairs, because they are likely to be too short to provide enough long-range information for assembling transcripts. The *k*-mer pairs are stored in three Bloom filters, separately representing left *k*-mers, right *k*-mers, and the pairing between *k*-mers. These Bloom filters are crucial to guiding graph traversal for the assembly of transcripts in Stage 3, which will be discussed later.

2.2.4 Stratification of assembled fragments

Assembled fragments are divided into strata (14 by default) according to their lengths and their minimum *k*-mer counts (Figure 2.6). Fragments are stratified based on their minimum *k*-mer count to separate fragments representing isoforms expressed in different order of magnitude. Fragments are also divided into “long” and “short” groups. “Long” fragments are longer than or equal to the first quartile of the initial sample of assembled fragments, and therefore they are represented by at least one *k*-mer pair. Hence, “long” fragments, compared to “short” fragments, are more extendable with *k*-mer pairs. Strata are stored in separate FASTA files and are retrieved in a specific order in Stage 3.

		Fragment length (*l*)	
		$l \geq$ Q1	**$l <$ Q1**
Minimum *k*-mer count (*c*)	**$100{,}000 \leq c$**	long.e5.fa	short.e5.fa
	$10{,}000 \leq c < 100{,}000$	long.e4.fa	short.e4.fa
	$1{,}000 \leq c < 10{,}000$	long.e3.fa	short.e3.fa
	$100 \leq c < 1{,}000$	long.e2.fa	short.e2.fa
	$10 \leq c < 100$	long.e1.fa	short.e1.fa
	$1 < c < 10$	long.e0.fa	short.e0.fa
	$c = 1$	long.01.fa	short.01.fa

Figure 2.6 Stratification of assembled fragments.

Each assembled fragment is written to a specific FASTA file according to its length (*l*) class and its minimum *k*-mer count (*c*). Q1 is the first quartile of fragment lengths in the initial sample of assembled fragments.

2.3 Stage 3: Reconstruction of transcripts

2.3.1 Complexity reduction of de Bruijn graph

Owing to the error correction in Stage 2, the entire set of assembled fragments would contain less *k*-mers than the original input reads. In other words, the DBG of assembled fragments would be smaller and thus less complex than the original DBG constructed from input reads. Since a less complex DBG would increase the efficiency of graph search, the DBG-Bf is emptied and re-populated by *k*-mers of all assembled fragments. The updated DBG-Bf would have a lower false-positive rate than before due to the reduction in the number of *k*-mers in the DBG.

2.3.2 Extension of fragment sequences with paired *k*-mers

Transcripts are reconstructed by extending assembled fragments with *k*-mer pairs extracted during Stage 2. All "long" fragments are evaluated before "short" fragments. Fragments are retrieved from the highest minimum *k*-mer count stratum down to the second lowest stratum. Fragments in the lowest stratum are not extended because fragments with singleton *k*-mers (count = 1) tend to introduce misassemblies. Fragments are extended on both sides by finding paths supported by consecutive *k*-mer pairs using a depth-first-search approach, as illustrated in Figure 2.7. Consecutive *k*-mer pairs are needed to account for false-positives in the Bloom filters for *k*-mer pairs. Each step of the search is advanced by choosing the *k*-mer with the highest median look-ahead *k*-mer count. The extension terminates when no more paths supported by *k*-mer pairs are found.

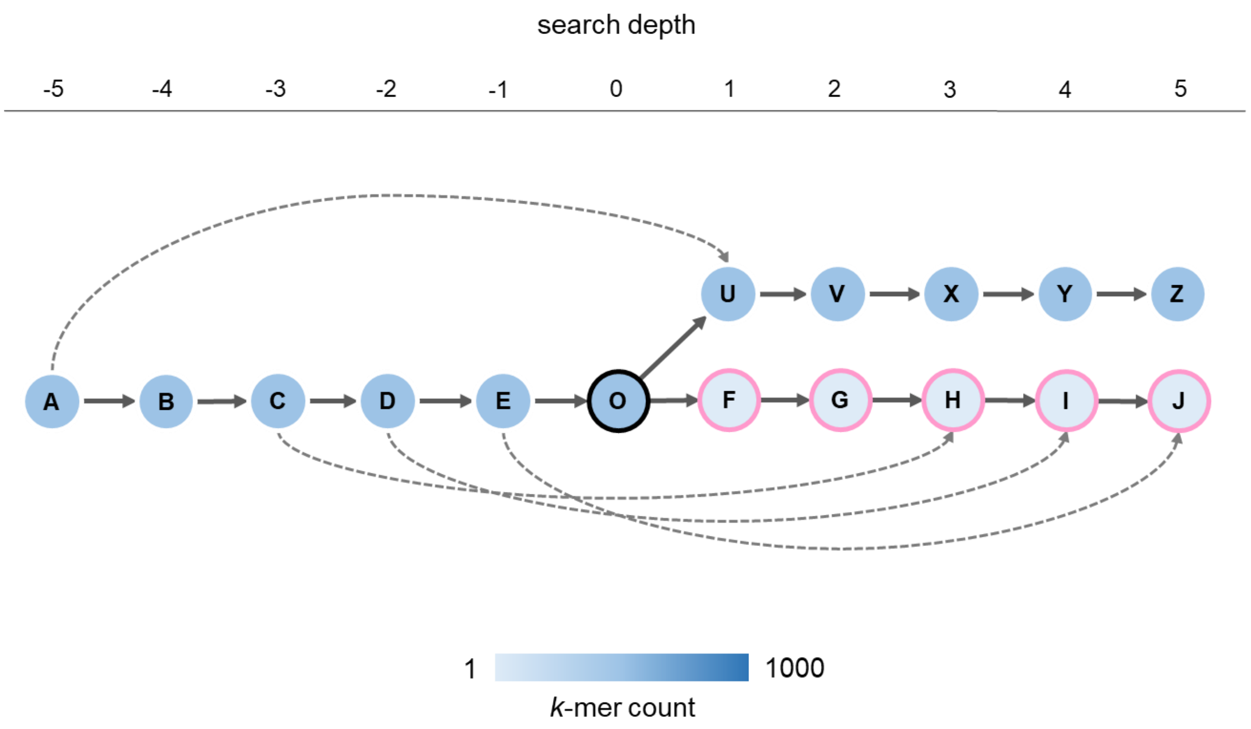


Figure 2.7 Extension with *k*-mer pairs.

In this example, paired *k*-mers are separated by 6 steps in the graph. An extension to the right at "O" is made by searching for a path supported by *k*-mer pairs (dotted lines). At "O," more than one extension is possible. With a look-ahead of 3 steps, the search would visit the "U" branch first because the median look-ahead *k*-mer count of "U-V-X-Y" is higher (as indicated by the darker blue color of the nodes) than that of the "F-G-H-I." Although "U" forms a *k*-mer pair with "A," there are not enough consecutive *k*-mers supporting the extension into the "U" branch. When the maximum search depth is reached, the search backtracks to "O" and advances to "F." Since sufficient (eg. 3) consecutive *k*-mer pairs (dotted lines) are found, an extension into the "F" branch is made (highlighted in pink).

Chapter 3: Results

3.1 Benchmarking Specification

3.1.1 Choosing *de novo* RNA-seq assemblers

To avoid repetition of results presented in a recent benchmarking study for *de novo* RNA-seq assemblers [25], I chose to compare the performance of RNA-Bloom with the two best performing methods that were most recently updated, Trans-ABySS v1.5.5 (August 2nd, 2016) and Trinity v2.4.0 (February 5th, 2017).

3.1.2 Datasets

Benchmarking was performed on three paired-end strand-specific RNA-seq datasets: a simulated human transcriptome, a healthy human blood transcriptome, and a sample of Universal Human Reference RNA. The simulated human transcriptome was generated with FluxSimulator v1.4.0 [48]. The human blood transcriptome is publicly available on Sequence Read Archive (SRR1957705) and is an Illumina RNA-seq sample of blood pooled from five healthy males. The simulated and real datasets were primarily used to compare the memory usage, runtime, sensitivity, and specificity of the assemblers. The Universal Human Reference RNA (UHRR) dataset comprises of the transcriptomes of ten different cancer cell lines and it is publicly available on Illumina BaseSpace (mRNA-UHRR-C1_S1_L001). Since a variety of fusion transcripts in UHRR and its constituent cell lines have been previously reported in the literature [49-55], the UHRR data in this benchmark was used to assess the performance in assembling fusion transcripts. All three datasets were preprocessed with Trimmomatic v0.36 [56] to remove Illumina adaptor sequences and trim poor-quality nucleotides with Phred scores less than 5, as recommended in an earlier study [57]. The specifications of the datasets are summarized in the Table 3.1.

Table 3.1 Data Specification.

Dataset	**Read Length (bp)***	**Size#**
Simulated human transcriptome	2 x 100	41 million
Healthy human blood transcriptome	2 x 100	44 million
Universal Human Reference RNA	2 x 75	41 million

* standard read length before trimming with Trimmomatic
number of read pairs after trimming with Trimmomatic

3.1.3 Assembly assessment tools

Assembly quality was assessed by rnaQUAST v1.4 [58], which is a software for evaluating the completeness and correctness of RNA-seq assemblies. As the name implies, rnaQUAST is based on QUAST [59], a popular tool for genome assembly evaluation. For all benchmarking performed, rnaQUAST was provided with the human reference genome build 38 (GRCh38) and the Ensembl annotation. Since sequencing artifacts often reside in short contigs, only contigs longer than or equal to double the read length were considered.

Fusion transcripts were detected from the assemblies of UHRR using Post-Assembly Variant Finder (PAVFinder, https://github.com/bcgsc/pavfinder) v0.2.0. To the best of my knowledge, PAVFinder is the only currently known method that detects fusion transcripts from any given RNA-seq assembly. PAVFinder detects fusion transcripts by finding split-alignments of assembled transcripts to the reference genome whereas older fusion detection methods, such as Tophat-Fusion [60] and deFuse [61], mainly rely on discovering discordant alignments of reads to the reference genome. PAVFinder was developed with the assumption that assembled sequences, compared to individual sequence reads, would yield more sensitive and accurate alignments to the reference genome. Compared to Tophat-Fusion and deFuse, PAVFinder (when used with Trans-ABySS), indeed, has a higher true-positive rate and lower false-positive rate in detecting gene fusions in a simulated dataset (Figure 3.1). For comparison with PAVFinder, the UHRR dataset was also processed with Tophat-Fusion, which is the next best method behind PAVFinder.

3.1.4 Computational specification

Each assembly was generated on a 64-bit high-performance computing machine running CentOS 7.1.1503 with 2.5 TB of RAM and 128 Intel Xeon E7-8867 processors at 2.5 GHz. All assemblers used a *k*-mer size of 25 and were run in strand-specific mode using 12 threads. Particularly, Trans-ABySS was set to use a coverage threshold of 1 (default is 2) such that extremely low-expressed transcripts can still be assembled. RNA-Bloom and Trinity do not have such a setting. The peak memory usage of each assembly was tracked by a custom Python script.

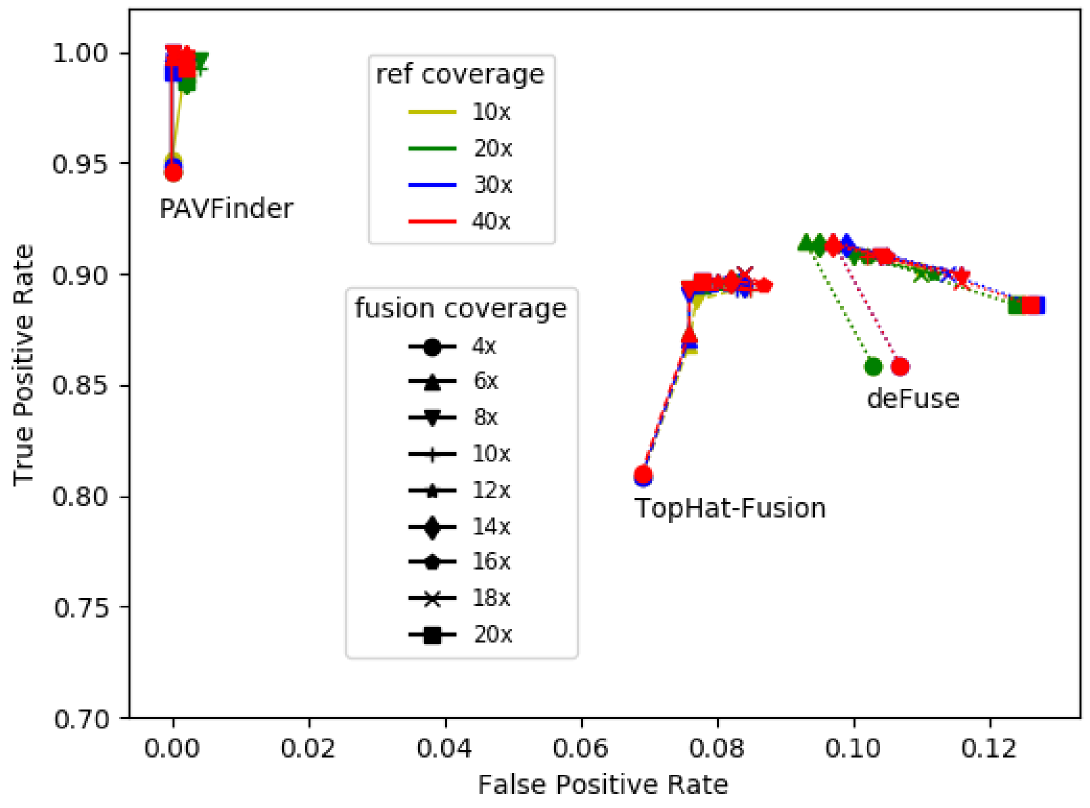


Figure 3.1 Comparison of PAVFinder, Tophat-Fusion, and deFuse.

A comparison of PAVFinder (coupled with Trans-ABySS), Tophat-Fusion, and deFuse in detecting gene fusions from a simulated dataset. Gene fusions are simulated at coverages from 4x to 20x (step size of 2) with varying background levels from 10x to 40x (step size of 10).

3.2 Benchmark 1: Simulated Human Transcriptome

3.2.1 Computational performance

RNA-Bloom has the fastest runtime and the smallest peak memory usage in assembling the simulated human transcriptome. RNA-Bloom used 39 minutes and had a peak memory usage less than 3 GB. Trinity had the highest peak memory usage, 4.6 times more than RNA-Bloom. Trans-ABySS was the slowest, 3.97 times slower than RNA-Bloom. The computational performance of the three assemblers is summarized in Table 3.2.

Table 3.2 Performance in assembling simulated RNA-seq using 12 threads.

Assembler	Peak Memory (GB)	Wall-clock Time (hour)
RNA-Bloom	**2.84**	**0.65**
Trans-ABySS	4.66	2.58
Trinity	13.18	1.43

The best results are highlighted in bold.

3.2.2 Assembly sensitivity

The number of annotated isoforms recovered and the number of exons recovered from the three assemblies are shown in Figure 3.2 and Figure 3.3, respectively. The Trans-ABySS assembly has the largest number of annotated isoforms covered 90% or less, followed by RNA-Bloom and Trinity. The RNA-Bloom assembly has the most full-length annotated isoforms, marginally higher than Trinity and Trans-ABySS.

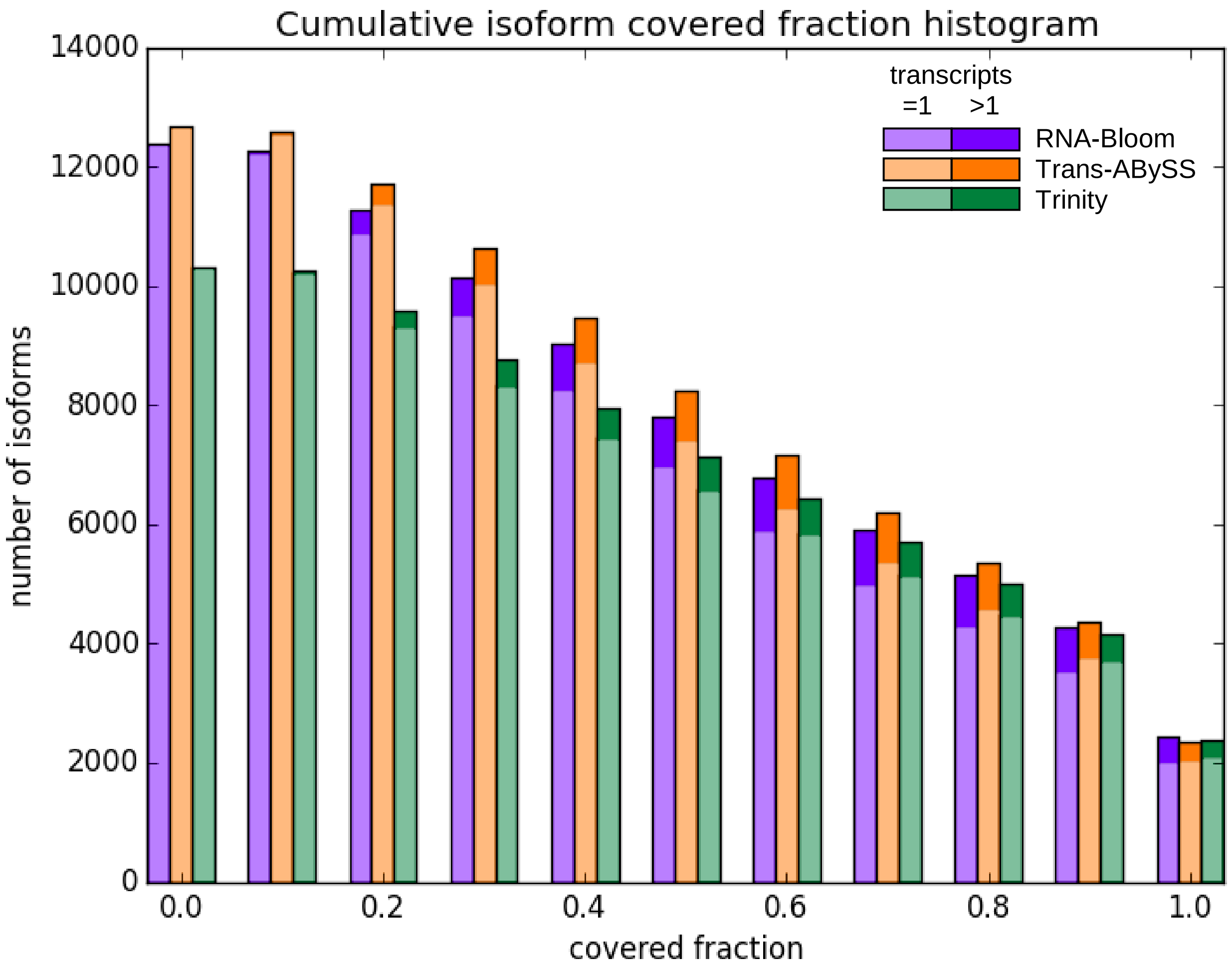


Figure 3.2 Number of isoforms covered by assembled transcripts.

The histogram bars are two-toned. The lighter tone represents the number of isoforms covered at or above the threshold fraction by one assembled transcript. If multiple assembled transcripts are covering the isoform, the largest covered fraction is selected. The darker tone represents the number of isoforms covered at or above the threshold fraction by more than one assembled transcripts.

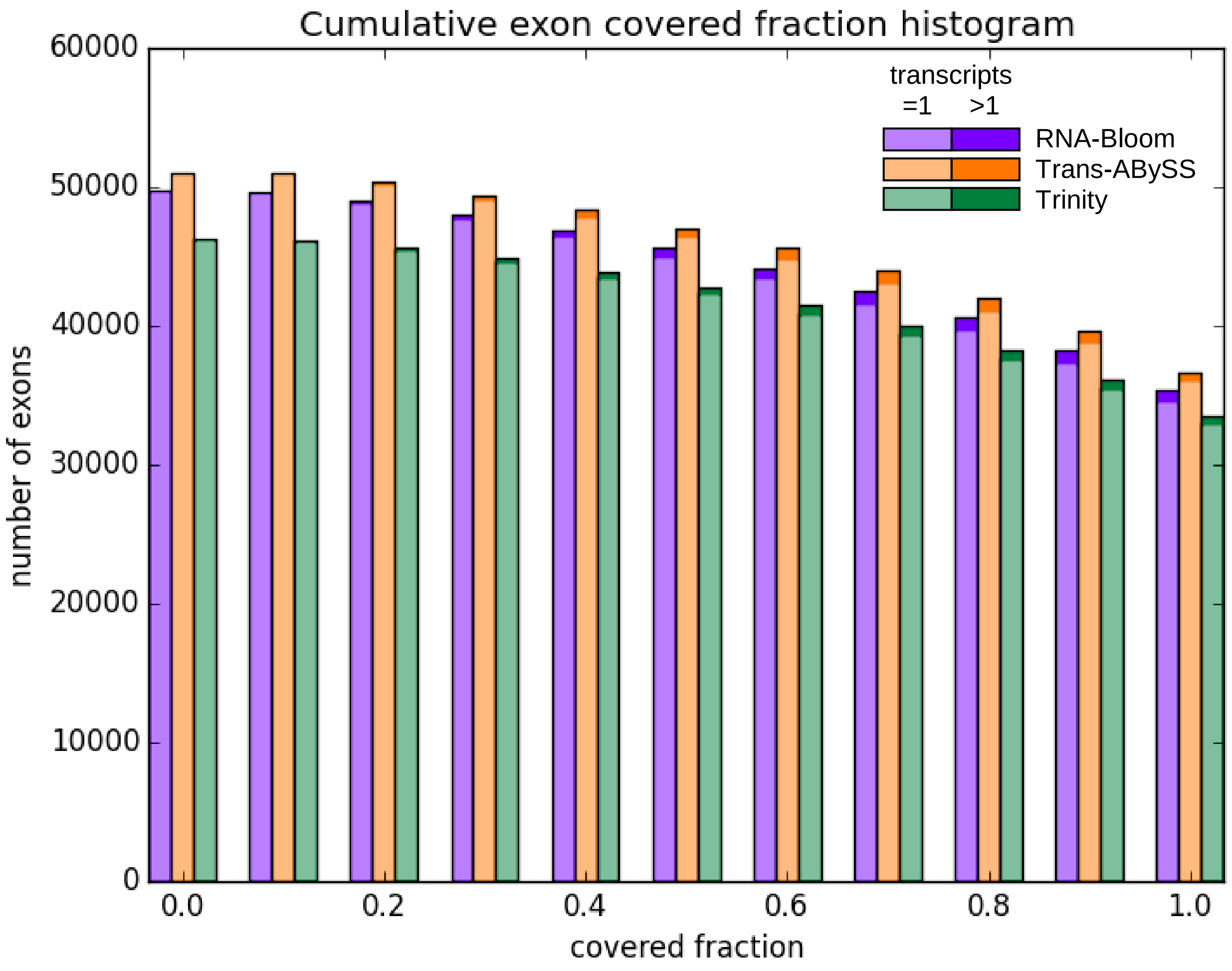


Figure 3.3 Number of exons covered by assembled transcripts.

The histogram bars are two-toned. The lighter tone represents the number of exons covered at or above the threshold fraction by one assembled transcript. If multiple assembled transcripts are covering the isoform, the largest covered fraction is selected. The darker tone represents the number of exons covered at or above the threshold fraction by more than one assembled transcripts.

3.2.3 Assembly specificity

Among the three assemblers, RNA-Bloom has the highest specificity; it has the least misassemblies and the lowest average mismatches per assembled transcript. Trinity created the most misassemblies, whereas Trans-ABySS has the highest average mismatches per assembled transcript. The specificity metrics are summarized in Table 3.3.

Table 3.3 Specificity metrics for assembling simulated RNA-seq.

Assembler	Average mismatches per transcript	Misassemblies
RNA-Bloom	**0.213**	**13**
Trans-ABySS	0.522	21
Trinity	0.499	66

The best results are highlighted in bold.

3.3 Benchmark 2: Human Blood Transcriptome

3.3.1 Computational performance

RNA-Bloom has the fastest runtime and the smallest peak memory usage in assembling the human blood transcriptome. RNA-Bloom used 1.11 hour and has a peak memory usage of 5.69 GB. Trinity has the highest peak memory usage of 20.05 GB, which is 3.52 times more than RNA-Bloom. Trinity also has the slowest runtime of 11.96 hours, which is 10.77 times slower than RNA-Bloom. The computational performance of the three assemblers is summarized in Table 3.4.

Table 3.4 Performance in assembling human blood transcriptome using 12 threads.

Assembler	Peak Memory (GB)	Wall-clock Time (hour)
RNA-Bloom	**5.69**	**1.11**
Trans-ABySS	10.16	6.00
Trinity	20.05	11.96

The best results are highlighted in bold.

3.3.2 Assembly sensitivity

As shown in Figure 3.4 and 3.5, the Trinity assembly has the largest number of isoforms covered 50% or more in length and the largest number of exons covered 90% or more in length. This is completely different from the benchmarking with simulated data where Trans-ABySS performed the best throughout the entire range of isoform coverage fractions (Figure 3.2) and exon coverage fractions (Figure 3.3). Despite of this, the RNA-Bloom assembly has the largest number of isoforms above 0 ~ 40% coverage in length and the largest number of exons above 0 ~ 70% coverage in length. This suggests that RNA-Bloom has the best capability of reconstructing transcripts that are partially represented in the presence of experimental noise.

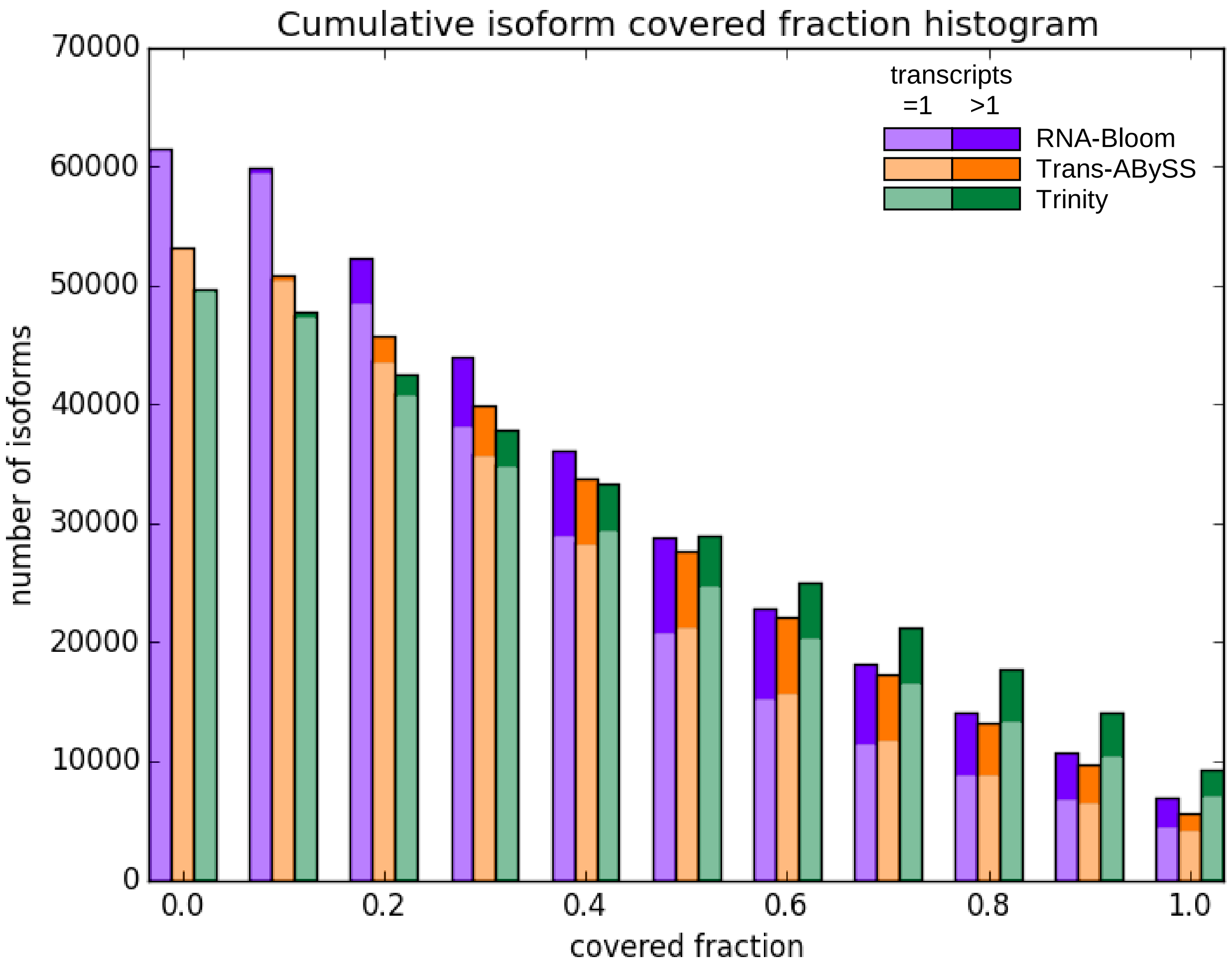


Figure 3.4 Number of isoforms covered by assembled transcripts.

The histogram bars are two-toned. The lighter tone represents the number of isoforms covered at or above the threshold fraction by one assembled transcript. If multiple assembled transcripts are covering the isoform, the largest covered fraction is selected. The darker tone represents the number of isoforms covered at or above the threshold fraction by more than one assembled transcripts.

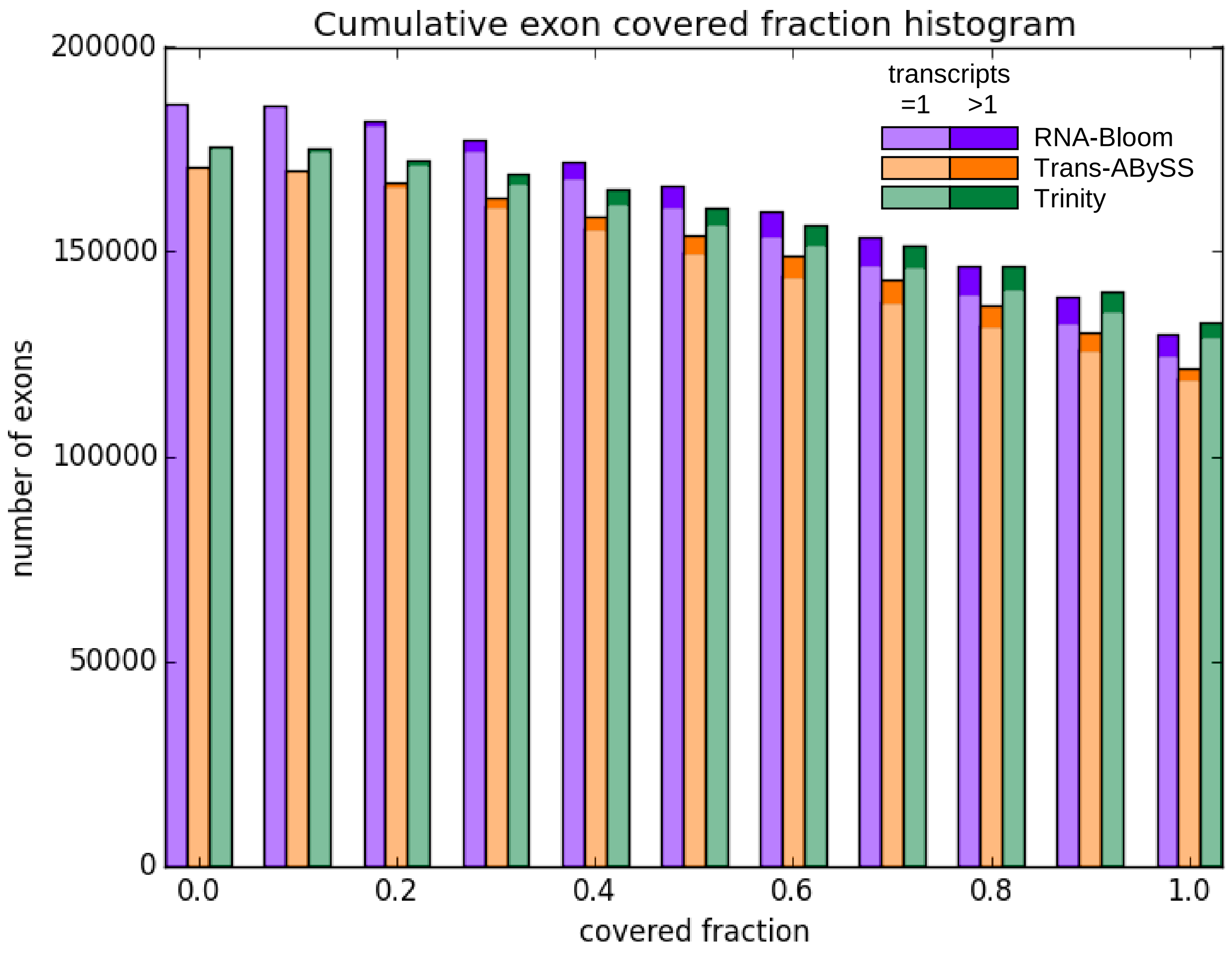


Figure 3.5 Number of exons covered by assembled transcripts.

The histogram bars are two-toned. The lighter tone represents the number of exons covered at or above the threshold fraction by one assembled transcript. If multiple assembled transcripts are covering the isoform, the largest covered fraction is selected. The darker tone represents the number of exons covered at or above the threshold fraction by more than one assembled transcripts.

3.3.3 Assembly specificity

Trans-ABySS has the best specificity while Trinity has the worst in assembling the human blood transcriptome. RNA-Bloom produced 1.49 times more misassemblies than Trans-ABySS whereas Trinity produced nearly 7 times more misassemblies than Trans-ABySS. The specificity of the three assemblers is summarized in Table 3.5. Trinity created the most misassembled transcripts in both simulated and real data. Although Trinity has the largest average mismatches per transcripts for real data, it has less average mismatches per transcript than Trans-ABySS for the simulated data. However, Trans-ABySS and RNA-Bloom are very comparable in terms of specificity for both simulated and real data.

Table 3.5 Specificity metrics for assembling human blood transcriptome.

Assembler	Average mismatches per transcript	Misassemblies
RNA-Bloom	0.676	556
Trans-ABySS	**0.599**	**373**
Trinity	1.162	2598

The best results are highlighted in bold.

3.4 Benchmark 3: Universal Human Reference RNA

3.4.1 Memory usage and runtime

RNA-Bloom has the fastest runtime and the smallest peak memory usage in assembling the UHRR dataset. RNA-Bloom used 22 minutes and has a peak memory usage of 4.15 GB. Trinity has the highest peak memory usage of 19.93 GB, which is 4.8 times more than RNA-Bloom. Trinity also has the slowest runtime of 16.73 hours, which is 45.22 times slower than RNA-Bloom. The computational performance of the three assemblers is summarized in Table 3.6.

Table 3.6 Performance in assembling UHRR using 12 threads.

Assembler	Peak Memory (GB)	Wall-clock Time (hour)
RNA-Bloom	**4.15**	**0.37**
Trans-ABySS	7.31	1.95
Trinity	19.93	16.73

The best results are highlighted in bold.

3.4.2 Fusion transcripts detected

PAVFinder detected 49 fusions in the RNA-Bloom assembly, 46 fusions in the Trans-ABySS assembly, and 89 fusions in the Trinity assembly (Table 3.7). On the other hand, Tophat-Fusion detected 77 fusions in total. RNA-Bloom assembled the most fusion transcripts supported by the literature, whereas Trinity assembled the least (Table 3.8). PAVFinder, when used with RNA-Bloom or Trans-ABySS, detected more literature-supported gene fusions than Tophat-

Fusion. This appears to agree with the higher true positive rate of PAVFinder shown in Figure 3.1. A large portion of fusions detected by PAVFinder and Tophat-Fusion are not supported by the literature and there is little overlap among the four sets of fusions detected (Figure 3.5). This suggests that the fusions detected lacking literature support may be false-positives, but the validity of these fusions may be verified with RT-PCR.

Table 3.7 Number of fusion transcripts detected.

	RNA-Bloom	Trans-ABySS	Trinity	Tophat-Fusion
Supported by literature	12	10	4	6
Not supported by literature	37	36	85	71
Total	49	46	89	77

Table 3.8 Fusion transcripts supported by literature.

5' Gene	3' Gene	RNA-Bloom	Trans-ABySS	Trinity	Tophat-Fusion	Literature
BCR	*ABL1*	✓	✕	✓	✓	[49, 50]
BAG6	*SLC44A4*	✓	✓	✕	✕	[51]
RPS6KB1	*VMP1*	✓	✓	✕	✓	[49]
GAS6	*RASA3*	✓	✓	✕	✕	[49]
NUP214	*XKR3*	✓	✓	✓	✓	[49, 50]
FGFR1	*NSD3*	✓	✕	✕	✕	[50]
BCAS4	*BCAS3*	✓	✓	✓	✓	[49]
HOMEZ	*MYH6*	✓	✓	✕	✕	[52]
KANSL1	*ARL17A*	✓	✓	✕	✕	[52, 55]
USP32	*APPBP2*	✓	✓	✕	✕	[53]
TANC2	*CA4*	✓	✓	✕	✓	[54]
ARFGEF2	*SULF2*	✓	✓	✓	✓	[49]

'✓' indicates that the fusion was found in the assembly.
'✕' indicates that the fusion was not found in the assembly.

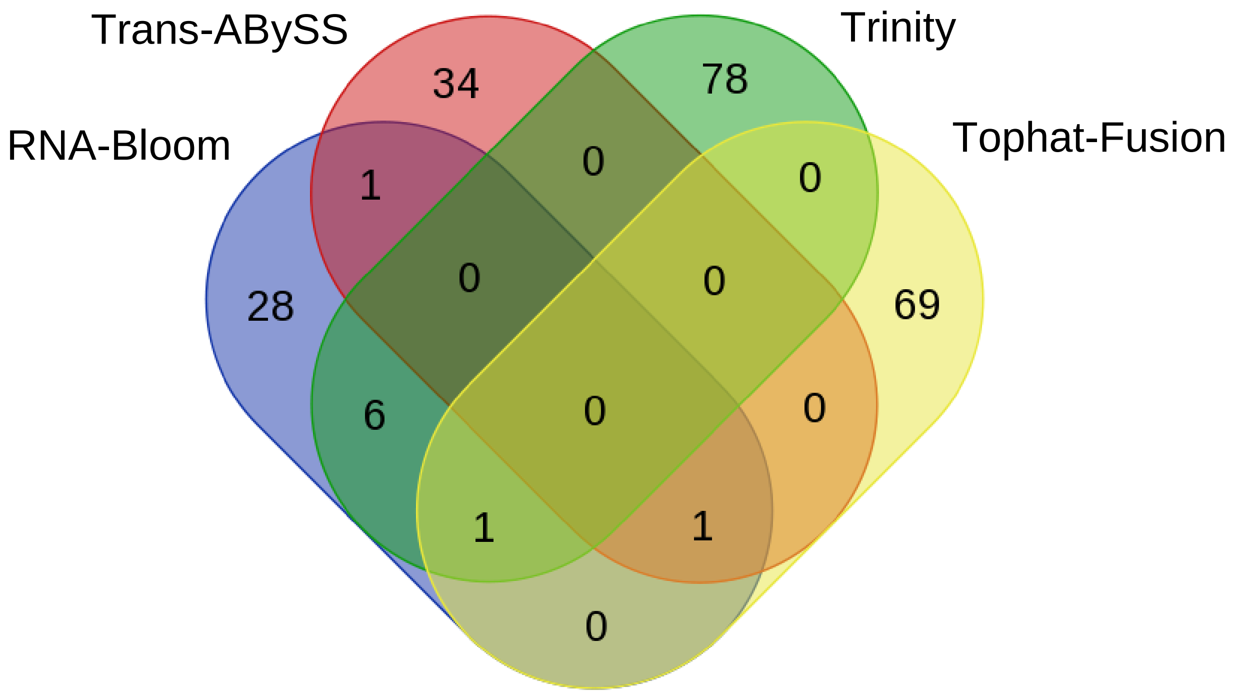


Figure 3.6 Venn diagram of fusion transcripts not supported by literature.

Chapter 4: Discussion

RNA-Bloom has superior runtime and memory usage in the three benchmarks performed, while achieving comparable sensitivity and specificity to Trans-ABySS and Trinity in assembling simulated and real RNA-seq data. Compared to Trans-ABySS, RNA-Bloom was 3.97 ~ 5.41 times quicker using 12 threads and used 55.9 ~ 60.0% of Trans-ABySS' peak memory usage. Compared to Trinity, RNA-Bloom was 2.20 ~ 45.22 times quicker using 12 threads and used 20.8 ~ 28% of Trinity's peak memory usage. RNA-Bloom has made the least misassemblies for the simulated data. Its sensitivity in assembling simulated data and real data is between Trans-ABySS and Trinity. RNA-Bloom's significantly faster runtime, lower memory usage, and superior reconstruction of fusion transcripts present its utility in a potential clinical setting, where resources and accuracy are of utmost importance.

While it has the best sensitivity in assembling the simulated data, Trans-ABySS has the worst sensitivity in assembling the real data. However, Trans-ABySS reconstructed only two less known fusion transcripts in UHRR when compared to RNA-Bloom. On the other hand, Trinity recovered the most annotated isoforms in the human blood transcriptome at the cost of significantly more misassemblies than those of Trans-ABySS and RNA-Bloom combined. Since misassemblies are inherently chimeric transcripts, one would expect Trinity to have the best reconstruction of known fusion transcripts in UHRR. Surprisingly, Trinity performed the worst in the benchmark with UHRR. Trinity's extremely slow runtime (over 45 times slower than RNA-Bloom) and poor reconstruction of known fusion transcripts in UHRR suggest that Trinity is not a robust tool for assembling complex transcriptomes in cancer cells. PAVFinder, when used with RNA-Bloom or Trans-ABySS, appeared to have a higher sensitivity in detecting gene fusions than Tophat-Fusion.

Although it is unanimous that RNA-Bloom has the best resource utilization in RNA-seq assembly, it would be worthwhile to investigate in the higher sensitivity of Trinity observed in the human blood transcriptome data and the higher sensitivity of Trans-ABySS observed in the simulated data. A potential starting point would be to understand why certain read-pairs could not be connected in Stage 2. Moreover, RNA-Bloom currently only uses pairs of *k*-mers separated by the first quartile of fragment lengths. Since longer range linkage information may help assembly of long transcripts, RNA-Bloom can perhaps incorporate pairs of *k*-mers separated by the third quartile of fragment lengths at the cost of slight increase in memory consumption.

RNA-Bloom currently only supports Illumina RNA-seq data, but long noisy reads from Pacific Biosciences or Oxford Nanopore can potentially be utilized to resolve ambiguous branching in the DBG due to alternative splicing and paralogous transcripts. In addition, 10x Genomics' linked-reads technology, which emerged recently, provides long range linkage information for assembling genomes [62] and single-cell 3' profiling of transcripts [63]. Since the linked-reads technology is based on Illumina's sequencing platform, this new data type can be adapted in RNA-Bloom once it gains traction in transcriptome research.

In this thesis, I have presented the work on my lightweight *de novo* RNA-seq assembler, which has fulfilled the objective of achieving lower runtimes and lower memory usage and retaining similar or better accuracy than current state-of-the-art methods. I foresee that RNA-Bloom would be used by many researchers, in both large and small labs, that study transcriptomics and RNA biology.

Bibliography

1. Martin, J.A. and Z. Wang, *Next-generation transcriptome assembly.* Nat Rev Genet, 2011. **12**(10): p. 671-82.
2. Byron, S.A., et al., *Translating RNA sequencing into clinical diagnostics: opportunities and challenges.* Nat Rev Genet, 2016. **17**(5): p. 257-71.
3. Glenn, T.C., *Field guide to next-generation DNA sequencers.* Molecular Ecology Resources, 2011. **11**(5): p. 759-769.
4. Au, K.F., et al., *Characterization of the human ESC transcriptome by hybrid sequencing.* Proc Natl Acad Sci U S A, 2013. **110**(50): p. E4821-30.
5. Tilgner, H., et al., *Defining a personal, allele-specific, and single-molecule long-read transcriptome.* Proc Natl Acad Sci U S A, 2014. **111**(27): p. 9869-74.
6. Bolisetty, M.T., G. Rajadinakaran, and B.R. Graveley, *Determining exon connectivity in complex mRNAs by nanopore sequencing.* Genome Biol, 2015. **16**: p. 204.
7. Birol, I., et al., *De novo Transcriptome Assemblies of Rana (Lithobates) catesbeiana and Xenopus laevis Tadpole Livers for Comparative Genomics without Reference Genomes.* PLoS One, 2015. **10**(6): p. e0130720.
8. Singh, R., et al., *Improved annotation with de novo transcriptome assembly in four social amoeba species.* BMC Genomics, 2017. **18**(1): p. 120.
9. Duan, J., et al., *Optimizing de novo common wheat transcriptome assembly using short-read RNA-Seq data.* BMC Genomics, 2012. **13**: p. 392.
10. Iorizzo, M., et al., *De novo assembly and characterization of the carrot transcriptome reveals novel genes, new markers, and genetic diversity.* BMC Genomics, 2011. **12**: p. 389.
11. Northcott, P.A., et al., *Subgroup-specific structural variation across 1,000 medulloblastoma genomes.* Nature, 2012. **488**(7409): p. 49-56.
12. Kasaian, K., et al., *Complete genomic landscape of a recurring sporadic parathyroid carcinoma.* J Pathol, 2013. **230**(3): p. 249-60.
13. Morin, R.D., et al., *Mutational and structural analysis of diffuse large B-cell lymphoma using whole-genome sequencing.* Blood, 2013. **122**(7): p. 1256-65.
14. Cancer Genome Atlas Research, N., *Comprehensive molecular characterization of gastric adenocarcinoma.* Nature, 2014. **513**(7517): p. 202-9.
15. Thibodeau, M.L., et al., *Genomic profiling of pelvic genital type leiomyosarcoma in a woman with a germline CHEK2:c.1100delC mutation and a concomitant diagnosis of metastatic invasive ductal breast carcinoma.* Cold Spring Harb Mol Case Stud, 2017.
16. Robertson, G., et al., *De novo assembly and analysis of RNA-seq data.* Nat Methods, 2010. **7**(11): p. 909-12.
17. Grabherr, M.G., et al., *Full-length transcriptome assembly from RNA-Seq data without a reference genome.* Nat Biotechnol, 2011. **29**(7): p. 644-52.
18. Schulz, M.H., et al., *Oases: robust de novo RNA-seq assembly across the dynamic range of expression levels.* Bioinformatics, 2012. **28**(8): p. 1086-92.
19. Peng, Y., et al., *IDBA-tran: a more robust de novo de Bruijn graph assembler for transcriptomes with uneven expression levels.* Bioinformatics, 2013. **29**(13): p. i326-34.

20. Safikhani, Z., et al., *SSP: an interval integer linear programming for de novo transcriptome assembly and isoform discovery of RNA-seq reads.* Genomics, 2013. **102**(5-6): p. 507-14.
21. Xie, Y., et al., *SOAPdenovo-Trans: de novo transcriptome assembly with short RNA-Seq reads.* Bioinformatics, 2014. **30**(12): p. 1660-6.
22. Chang, Z., et al., *Bridger: a new framework for de novo transcriptome assembly using RNA-seq data.* Genome Biol, 2015. **16**: p. 30.
23. Liu, J., et al., *BinPacker: Packing-Based De Novo Transcriptome Assembly from RNA-seq Data.* PLoS Comput Biol, 2016. **12**(2): p. e1004772.
24. Trapnell, C., et al., *Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation.* Nat Biotechnol, 2010. **28**(5): p. 511-5.
25. Wang, S. and M. Gribskov, *Comprehensive evaluation of de novo transcriptome assembly programs and their effects on differential gene expression analysis.* Bioinformatics, 2017. **33**(3): p. 327-333.
26. Nagarajan, N. and M. Pop, *Sequence assembly demystified.* Nat Rev Genet, 2013. **14**(3): p. 157-67.
27. Simpson, J.T., et al., *ABySS: a parallel assembler for short read sequence data.* Genome Res, 2009. **19**(6): p. 1117-23.
28. Bloom, B.H., *Space/time trade-offs in hash coding with allowable errors.* Commun. ACM, 1970. **13**(7): p. 422-426.
29. Pell, J., et al., *Scaling metagenome sequence assembly with probabilistic de Bruijn graphs.* Proc Natl Acad Sci U S A, 2012. **109**(33): p. 13272-7.
30. Birol, I., et al., *Spaced Seed Data Structures for De Novo Assembly.* Int J Genomics, 2015. **2015**: p. 196591.
31. Chikhi, R. and G. Rizk, *Space-efficient and exact de Bruijn graph representation based on a Bloom filter.* Algorithms Mol Biol, 2013. **8**(1): p. 22.
32. Jackman, S.D., et al., *ABySS 2.0: resource-efficient assembly of large genomes using a Bloom filter.* Genome Res, 2017. **27**(5): p. 768-777.
33. Magoc, T. and S.L. Salzberg, *FLASH: fast length adjustment of short reads to improve genome assemblies.* Bioinformatics, 2011. **27**(21): p. 2957-63.
34. Vandervalk, B.P., et al., *Konnector v2.0: pseudo-long reads from paired-end sequencing data.* BMC Med Genomics, 2015. **8 Suppl 3**: p. S1.
35. Zimin, A.V., et al., *The MaSuRCA genome assembler.* Bioinformatics, 2013. **29**(21): p. 2669-77.
36. Birol, I., et al., *Assembling the 20 Gb white spruce (Picea glauca) genome from whole-genome shotgun sequencing data.* Bioinformatics, 2013. **29**(12): p. 1492-7.
37. Zimin, A., et al., *Sequencing and assembly of the 22-gb loblolly pine genome.* Genetics, 2014. **196**(3): p. 875-90.
38. Pertea, M., et al., *StringTie enables improved reconstruction of a transcriptome from RNA-seq reads.* Nat Biotechnol, 2015. **33**(3): p. 290-5.
39. Li, H. and N. Homer, *A survey of sequence alignment algorithms for next-generation sequencing.* Brief Bioinform, 2010. **11**(5): p. 473-83.
40. Warren, R.L., et al., *LINKS: Scalable, alignment-free scaffolding of draft genomes with long reads.* Gigascience, 2015. **4**: p. 35.

41. Patro, R., S.M. Mount, and C. Kingsford, *Sailfish enables alignment-free isoform quantification from RNA-seq reads using lightweight algorithms.* Nat Biotechnol, 2014. **32**(5): p. 462-4.

42. Zhang, Z. and W. Wang, *RNA-Skim: a rapid method for RNA-Seq quantification at transcript level.* Bioinformatics, 2014. **30**(12): p. i283-i292.

43. Bray, N.L., et al., *Near-optimal probabilistic RNA-seq quantification.* Nat Biotechnol, 2016. **34**(5): p. 525-7.

44. Patro, R., et al., *Salmon provides fast and bias-aware quantification of transcript expression.* Nat Methods, 2017. **14**(4): p. 417-419.

45. Li, B. and C.N. Dewey, *RSEM: accurate transcript quantification from RNA-Seq data with or without a reference genome.* BMC Bioinformatics, 2011. **12**: p. 323.

46. Mohamadi, H., et al., *ntHash: recursive nucleotide hashing.* Bioinformatics, 2016. **32**(22): p. 3492-3494.

47. Song, L. and L. Florea, *Rcorrector: efficient and accurate error correction for Illumina RNA-seq reads.* Gigascience, 2015. **4**: p. 48.

48. Griebel, T., et al., *Modelling and simulating generic RNA-Seq experiments with the flux simulator.* Nucleic Acids Res, 2012. **40**(20): p. 10073-83.

49. Maher, C.A., et al., *Chimeric transcript discovery by paired-end transcriptome sequencing.* Proc Natl Acad Sci U S A, 2009. **106**(30): p. 12353-8.

50. Scolnick, J.A., et al., *An Efficient Method for Identifying Gene Fusions by Targeted RNA Sequencing from Fresh Frozen and FFPE Samples.* PLoS One, 2015. **10**(7): p. e0128916.

51. Berger, M.F., et al., *Integrative analysis of the melanoma transcriptome.* Genome Res, 2010. **20**(4): p. 413-27.

52. Kinsella, M., et al., *Sensitive gene fusion detection using ambiguously mapping RNA-Seq read pairs.* Bioinformatics, 2011. **27**(8): p. 1068-75.

53. Yoshihara, K., et al., *The landscape and therapeutic relevance of cancer-associated transcript fusions.* Oncogene, 2015. **34**(37): p. 4845-54.

54. Qu, K., J. Baker, and Y. Ma, *Applications of NGS to Screen FFPE Tumours for Detecting Fusion Transcripts*, in *Next Generation Sequencing in Cancer Research, Volume 2: From Basepairs to Bedsides*, W. Wu and H. Choudhry, Editors. 2015, Springer International Publishing: Cham. p. 155-177.

55. Wen, H., et al., *New fusion transcripts identified in normal karyotype acute myeloid leukemia.* PLoS One, 2012. **7**(12): p. e51203.

56. Bolger, A.M., M. Lohse, and B. Usadel, *Trimmomatic: a flexible trimmer for Illumina sequence data.* Bioinformatics, 2014. **30**(15): p. 2114-20.

57. Macmanes, M.D., *On the optimal trimming of high-throughput mRNA sequence data.* Front Genet, 2014. **5**: p. 13.

58. Bushmanova, E., et al., *rnaQUAST: a quality assessment tool for de novo transcriptome assemblies.* Bioinformatics, 2016. **32**(14): p. 2210-2.

59. Gurevich, A., et al., *QUAST: quality assessment tool for genome assemblies.* Bioinformatics, 2013. **29**(8): p. 1072-5.

60. Kim, D. and S.L. Salzberg, *TopHat-Fusion: an algorithm for discovery of novel fusion transcripts.* Genome Biol, 2011. **12**(8): p. R72.

61. McPherson, A., et al., *deFuse: an algorithm for gene fusion discovery in tumor RNA-Seq data.* PLoS Comput Biol, 2011. **7**(5): p. e1001138.

62. Weisenfeld, N.I., et al., *Direct determination of diploid genome sequences.* Genome Res, 2017. **27**(5): p. 757-767.
63. Zheng, G.X., et al., *Massively parallel digital transcriptional profiling of single cells.* Nat Commun, 2017. **8**: p. 14049.