Date of acceptance Grade

Instructor

Comparison of spliced alignment software in analyzing RNA-Seq data

Anna Kuosmanen

Helsinki October 16, 2013

M.Sc. thesis

UNIVERSITY OF HELSINKI
Department of Computer Science

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Faculty of Science	Department of Computer Science

Tekijä — Författare — Author
Anna Kuosmanen

Työn nimi — Arbetets titel — Title

Comparison of spliced alignment software in analyzing RNA-Seq data

Oppiaine — Läroämne — Subject
Bioinformatics

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
M.Sc. thesis	October 16, 2013	65 pages + 2 appendices

Tiivistelmä — Referat — Abstract

A recently developed protocol for sequencing RNA in a cell in a high-throughput manner, RNA-seq, generates from hundreds of thousands to a few billion short sequence fragments from each RNA sample. Aligning these fragments, or "reads", to the reference genome in a fast and accurate manner is a challenging task that has been tackled by many researchers over the past five years.

In this thesis I review the process of RNA-seq data creation and analysis, and introduce and compare some of the popular alignment software. As part of the thesis, I implemented an alignment software based on the novel idea of a limited range BWT-transformed index. This software, called SpliceAligner, is also introduced in detail. In addition to my own software, I chose for comparison Tophat, SpliceMap, MapSplice, SOAPsplice and SHRiMP2.

I tested the chosen software on simulated data sets with read lengths of 50, 100, 150 and 250 base pairs, as well as with data from a real RNA-seq experiment. I ranked the software based on the running time, number of reads mapped and the accuracy of the alignments. I also predicted transcripts from the alignments of the simulated data, and measured the correctness of the predictions.

With read lengths of 50 base pairs, 100 base pairs and 150 base pairs, speed, alignment accuracy and ease of use make Tophat a solid top choice. MapSplice is a comparable choice in speed and alignment accuracy, and SOAPsplice is only slightly behind, but their user interfaces are much more complicated. However, Tophat slowed down significantly as the read length increased to 250 base pairs and SOAPsplice completely failed to run with 250 base pairs long reads. This leaves MapSplice as the top choice for long reads in most cases.

My software SpliceAligner was competitive in the alignment accuracy with the top choices, but there still remains work to be done on the running speed as well as on multiple small optimizations.

ACM Computing Classification System (CCS):

Applied computing ˜ Bioinformatics[300],
Applied computing ˜ Molecular sequence analysis[500],
Applied computing ˜ Computational transcriptomics[300]

Avainsanat — Nyckelord — Keywords
bioinformatics, RNA-seq, short read mapping

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Contents

1 Introduction

Gene expression controls the functions of all the cells in an organism, and therefore has direct correlation with the condition of each cell. Different types of cells, under different conditions, can express different sets of genes, or different transcripts for same genes [DDM+12]. Therefore studying gene expression can give valuable insights to what is happening inside a cell, and even more importantly, why the cell functions that way.

Gene expression can be studied with many different techniques. Hybridization-based approaches, such as microarrays, are popular because of their low price and high throughput, but they have several limitations [WGS09, CADC10]. For example they can only identify known sequences in the sample. On the other hand, sequencing approaches such as Sanger sequencing can find novel expressed sequences, but have very limited throughput and are generally expensive [WGS09, CADC10].

The invention of next-generation sequencing led to a new high-throughput sequencing-based way of studying gene expression, RNA-seq. In an RNA-seq experiment the RNA is converted to cDNA (complementary DNA) and hundreds of thousands to a few billion of these molecules can be sequenced in parallel. Depending on the next-generation sequencing technology, the sequences, or "reads", from an RNA-seq experiment can be 35-1000 bases long, with the number of reads inversely correlating with their length. For example while 454 pyrosequencing creates hundreds of thousands to a few million of reads with length in 300-1000 bases[1], Illumina's sequencing-by-synthesis creates hundreds of millions to a few billion reads with lengths of 35-250 bases[2].

Differential expression analysis, discovering fusion genes and studying differential splicing are just a few examples of the topics that RNA-seq data is used for [CADC10, OM11, THS+13, SHP+10]. As the RNA-seq data is distinctly different from the data from both hybridization based approaches and earlier sequencing based approaches, completely new techniques had to be developed to handle the data.

Majority of RNA-seq analysis pipelines start with aligning the reads to the reference [CADC10, WL09, NPP+12, VHPV13]. Aligning one read to the reference is trivial, but the sheer number of them created from a single sequencing run makes many approaches infeasible. The spliced nature of RNA poses another challenge to

[1] 454.com/products

[2] www.illumina.com/systems

the alignment to the reference genome [MW11, CADC10]. Pseudogenes and repetitive regions also add to the problem [WL09]. Many tools using various approaches for solving these problems have been published in the past five years, one of the most well-known being Tophat [TPS09], by Trapnell *et al.*.

As the correct alignment is important for the downstream analysis, this thesis focuses on analyzing and comparing the performance of short read aligners designed for aligning RNA-seq reads to the genome. The analysis is based mostly on simulated data, as the *ground truth*, that is, the knowledge where the reads originated from, is not available for real RNA-seq experiment data. To test the scalability of the approaches, I also use one real RNA-seq data set in the analysis.

In Chapter 2 I introduce some biological concepts necessary for understanding the rest of the thesis, and in Chapter 3 I talk about methods for studying gene expression in general, and obtaining and analyzing RNA-seq data in particular. In Chapter 4 I describe the software chosen for this comparison, as well as the test setup. And finally the results of the comparisons can be found in Chapter 5, with summary and discussion following in Chapter 6.

As part of the thesis work, I implemented a short read aligner based on the novel idea of a limited range index. This tool, called SpliceAligner, is included in the comparison, and I describe the ideas behind it in detail in Chapter 4.3.

2 Biological background

The human genome contains approximately 21,000 protein-coding *genes* (Ensembl GRCh37.p10/NCBI Hg19 assembly), which control all the functions of a cell. In addition there exist several thousand non-coding RNA genes, which for example help regulate the protein-coding genes.

To create a functional product from the information stored in the gene, the DNA sequence of the gene needs to be *transcribed* to RNA. The RNA can then be *translated* to protein, or it can act in the RNA form by binding to other RNA or proteins (e.g. small interfering RNAs and micro RNAs that regulate gene expression by repressing translation of certain transcripts).

In transcription, the DNA double-helix opens, RNA polymerase attaches to the gene area and creates a single-stranded copy of the gene sequence, called the *primary transcript*, after which the double-helix closes again. In eukaryotes (e.g. humans), primary transcripts that will be used in protein synthesis (also called pre-mRNAs) undergo a series of modifications before they exit the cell nucleus and enter the cytoplasm for translation. The modifications are adding a 5'-cap and polyadenylation (adding a poly-A tail to the 3' end), both of which protect the mRNA from degradation and help in the translation initiation, and *RNA splicing*.

Eukaryotic genes consists of *exons* (for "expressed") and *introns* (for "intervening sequence"). In addition there are some nucleotides at both 5' and 3' ends, which are transcribed but not translated to proteins. These regions, which are not strictly introns, are called 5' UTR (untranslated region) and 3' UTR. In RNA splicing the introns are removed from the pre-mRNA transcript and the exons are spliced together to form the mature mRNA transcript, which can then be translated into protein (see Figure 1).

The pre-mRNA can be spliced in multiple ways, which results in multiple distinctly different mature transcripts (isoforms), and therefore multiple proteins being created from one gene. This phenomenon is called *alternative splicing* (see Figure 2 for some of the common variants). Alternative splicing is the reason why humans can have only a fraction of the number of genes you would expect based on organism complexity. Initial estimates for the number of protein-coding genes in humans were actually in the range of 35,000-150,000 [EG00, RJB$^+$00, LHP$^+$00].

Genes express more than one transcript simultaneously in an organism, and a higher number of possible transcripts correlates with more transcripts expressed

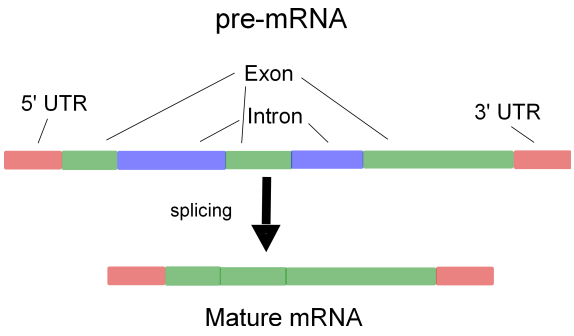


Figure 1: RNA splicing (exons and introns are not to scale). To form a mature mRNA transcript, introns are removed from the pre-mRNA and the exons are spliced together.

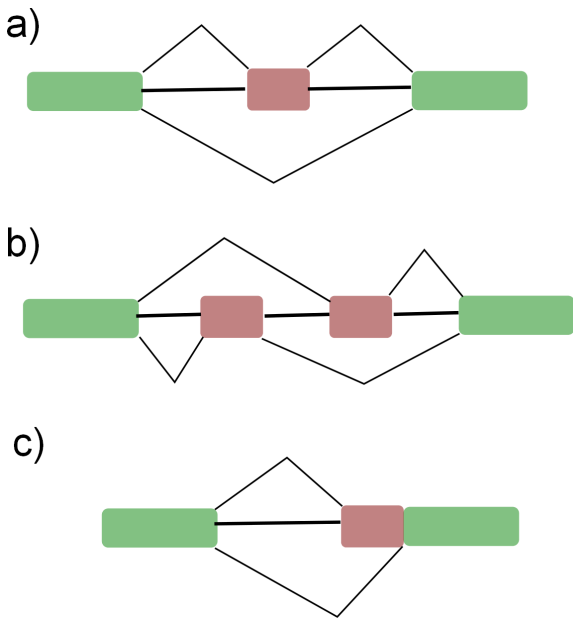


Figure 2: A few examples of alternative splicing. a) Exon skipping (cassette exon). Exon is either spliced out or retained. b) Mutually exclusive exons. Only one is included in the mature transcript. c) Alternative 3' acceptor site.

at a time [DDM+12]. However, for each cell type and condition, expression levels of one or two transcripts dominate over the others [DDM+12]. As the differently expressed transcripts control the functions of a cell differently, studying gene expression can lead to a better understanding of those functions as well as the causes leading to a specific behavior of a cell (e.g. uncontrolled cell growth in cancer).

3 Gene expression analysis

Many techniques exist to study the different transcripts expressed in a cell. They can be roughly divided into two categories: hybridization based approaches and sequencing based approaches [WGS09]. Hybridization based approaches, for example microarrays, have become popular because they are high-throughput and inexpensive [WGS09]. But they depend on the annotated genome for the probe design, and are therefore unable to find novel transcripts [WGS09, CADC10, NPP+12]. They also have limited dynamic detection range due to signal saturation at high end and noise at low end. [WGS09, CADC10, NPP+12].

Of the sequencing based approaches, Sanger sequencing was the first, but it was low throughput, expensive and generally not quantitative [WGS09]. Multiple tag-based methods were invented to provide data in a high-throughput manner, with more precise expression levels [WGS09, CADC10]. However, as they were all based on Sanger sequencing, price was still an issue [WGS09, CADC10]. The development of next-generation sequencing (NGS) technologies created a new method for analyzing the transcriptome, RNA-seq.

3.1 RNA-seq

In RNA-seq the RNA is converted to complementary DNA (cDNA), and each molecule is sequenced in high-throughput manner (either with or without amplification). For the next-generation sequencing technologies (also known as second-generation), the reads can be 35-1000 base pairs (bp) long[3][4][5], whereas the newer third-generation platforms can produce reads with lengths in several thousands [LLL+12]. However, even though third-generation systems are being rapidly developed, they are not yet commonly used [CCHD13, Gle11]. This thesis will focus on the next-generation sequencing and its data analysis.

Unlike hybridization-based approaches, RNA-seq methods are not limited to finding only annotated transcripts, and can find transcript boundaries at single-base resolution [WGS09, WL09, NPP+12]. If the reads are sequenced from both ends of the fragment (paired-end), the pair information can be used to reveal connectivity between exons. RNA-seq has a low level of background noise compared to

[3] 454.com/products

[4] www.illumina.com/systems

[5] https://products.appliedbiosystems.com

hybridization-based approaches, and has a high dynamic range of detectable expression levels thanks to the absence of the upper limit for number of reads sequenced [WGS09, WL09, NPP+12].

However, as with any method, RNA-seq has its weaknesses as well. As the reads are sequenced randomly from the transcriptome, some of the high abundance transcripts (e.g. housekeeping genes such as ubiquitin) can dominate the results [LLL+11]. Because of this, lowly expressed transcripts often cannot be quantified as reliably [LLL+11]. In addition, while sequencing is not affected by certain distortions that plague microarray experiments (e.g.chemical saturation for hybridization), the sample preparation protocols for the next-generation systems contain many steps where experimental conditions can introduce systematic biases [HIW12].

3.1.1 Sample preparation and library construction

Compared to microarrays, a relatively small amount of RNA is required for an RNA-seq experiment [KQGW12]. Generally the required amount for sequencing ranges from a few hundred nanograms to a few micrograms of RNA [WL09, VHPV13]. However, the majority ($> 90\%$) of the RNA present in the cell is ribosomal RNA (rRNA), which is not very informative for studying the transcriptome [WL09]. Therefore approximately 100 micrograms of total RNA are required for RNA-seq experiment, and the first step of the sample preparation is to enrich mRNA content by using oligo(dT) beads [WL09, HBD10] or deplete rRNAs with rRNA removal kits designed for that purpose [WL09]. The former approach enriches strictly mRNAs with poly-A tails, whereas in the latter case other types of non-ribosomal RNA (e.g. small regulatory RNAs) and mRNAs without poly-A tails are also enriched [CADC10, MW11].

As the next-generation sequencing platforms were designed for double-stranded DNA, after preparing the sample a required step in the library construction is to reverse transcribe the RNA to complementary DNA (cDNA) [CADC10]. For reverse transcription either oligo(dT) primers or random hexamer primers can be used [WL09, CADC10, HBD10]. Oligo(dT) primers have the advantage of targeting mRNAs, but the reverse transcriptase randomly falling off the template during reverse transcription creates a bias towards the 3'-end [WL09, CADC10]. Random hexamer primers create fragments starting from all along the transcripts, but they show clear sequence-specific bias, resulting in a distribution that is far from uniform [HBD10, RTD+11].

Either before or after reverse transciption, the RNA or the cDNA is broken into short fragments [WL09, CADC10]. These fragments are then size-separated, usually with agarose gel, and the fragments with the desired length can be separated for sequencing [CADC10]. As final steps before the sequencing, adapters are ligated to the fragments [WL09, CADC10], and optionally the fragments can be PCR amplified to achieve the desired coverage [WGS09].

Mistakes in the amplification step can cause PCR artifacts, that is, an overabundance of copies created from a small number of fragments [WGS09]. Determining whether an abundance of identical short reads really represents an abundant transcript or if it is merely a PCR artifact is one of the challenges of RNA-seq [WGS09].

3.1.2 Sequencing and imaging

After the library preparation, the fragments of RNA-converted-to-cDNA can be sequenced either from one end of the fragment (single-end reads) or from both ends (paired-end reads). The specifics of the sequencing process depend on the platform, but the basic principle is the same for all next-generation sequencing platforms [WL09, CCHD13]: the library of cDNA fragments is diluted to the point that each sequencing "unit" contains a single molecule, which is clonally amplified using PCR. Amplification is required to produce a signal strong enough to measure with the imaging instruments [Met10, CCHD13]. The resulting populations of identical fragments can then be sequenced in parallel by either using fluorescent nucleotides or measuring the reaction when nucleotides are added in some way (e.g. measuring pH changes) [Met10, Mar08, VDD09, CCHD13]. As nucleotides bind to their complements (A to T and C to G), the incorporation of a particular nucleotide reveals the nucleotide present at a given position in the read.

The first commercial sequencer, the 454 FLX Pyrosequencer introduced in 2005, uses a sequencing technology called *pyrosequencing* [Met10, Mar08, VDD09, CCHD13, WL09]. For pyrosequencing, nucleotides are added to the wells one type (A, C, G or T) at a time. Incorporation of a matching nucleotide starts a chemical chain reaction that produces light. The number of incorporated nucleotides can be deduced from the intensity of the light.

As pyrosequencing has only one type of nucleotide added to the process at a time but can incorporate multiple nucleotides to the growing chain in one cycle, mismatch errors are very rare, whereas homopolymer run errors (for example interpreting

"AA" as "A" or "AAA") are the common error type [Mar08, VDD09]. Deletions not involving homopolymers are also possible, as a single nucleotide might fail to be incorporated to the chain. The quality scores of 454 reads for homopolymer runs represent, for each of the same nucleotide after the first one, the probability that the homopolymer was of that length or longer.

Solexa released their Genome Analyzer the following year. Solexa was acquired by Illumina soon after, so this technology is commonly referred to as Illumina technology. Compared to 454, this sequencer was a "short read" sequencer: Where the first 454 machine produced hundreds of thousands of reads of lengths in hundreds of nucleotides, Illumina/Solexa Genome Analyzer produced billions of reads of length approximately 36 nucleotides (nt) [Met10, Mar08, VDD09]. Illumina technologies use an approach called *sequencing-by-synthesis* [Met10, Mar08, VDD09, CCHD13, WL09].

Sequencing-by-synthesis uses nucleotides combined with fluorescent reversible terminators. All four types of nucleotides (A, C, G and T) are added into the flowcell where the read clusters reside, matching nucleotides incorporate and the reaction terminates. Excess nucleotides are washed off, fluorescence is imaged and the terminators are cleaved off, then the next cycle can start. The primary error type on Illumina platforms is substitution [Gle11]. Illumina also provides quality scores for the reads that represents the probability of the base called in a given position to be correct.

The third major competitor in the NGS market is Applied Biosystems's SOLiD (Supported Oligonucleotide Ligation and Detection) system. SOLiD's sequencing approach is radically different from the other two systems: instead of DNA polymerase adding single nucleotides to the growing chain, SOLiD uses DNA ligase to bind probes to the chain [Met10, Mar08, VDD09]. A probe consists of a dinucleotide pair, six *degenerate nucleotides*, that is, nucleotides that pair with any nucleotide, and a fluorescent marker. A cDNA fragment is sequenced five times with varying starting positions for the first probe, so that every position is covered by two overlapping dinucleotides [Mar08].

SOLiD uses a four color system to encode bases, with each color coding for four different dinucleotides. As each nucleotide is interrogated by two dinucleotides, with SOLiD data it is easier to distinguish sequencing errors from true single nucleotide variations (SNVs) in aligned data [Met10, Mar08]. Sequencing error occurs only in one of the dinucleotides, whereas for an SNV both of the dinucleotides have to

be different, and the sequence must match the reference around the dinucleotide pair [Met10, Mar08]. However, this sequence interpretation that depends on the previous nucleotide makes SOLiD data unsuitable for *de novo* assembly, as calling one base incorrectly results to the misinterpretation of the rest of the read.

As each sequencing platform has its own unique characteristics, (e.g. read length, throughput and error profile), there is no best platform for all purposes [MW11, Met10, Gle11]. For example, long 454 reads are often more suitable for *de novo* assembly of small transcriptomes than shorter Illumina reads [MW11], and SOLiD's ability to differentiate base call errors from SNVs makes it excellent for SNV calling on low coverages [Met10]. The different characteristics of the platforms have to also be taken into account in the development of the analysis tools.

3.2 Analysing RNA-seq data

Because of weak or mixed signals during sequencing, not all the reads are of good quality [WL09]. Therefore before beginning the data analysis, the data should be ran through quality filters [WL09, VHPV13]. For example, reads containing many N's (which mean that the base could have been any of the A, C, G or T) [WL09] as well as very short reads featuring sequence repeats [MW11] can be removed to make the analysis faster. Most platforms also give each read a quality score which measures their confidence of each base being correct, and there are multiple software available for visualization of read qualities for easy quality control (e.g. FastQC [And10] and RSeQC [WWL12]). For an example of viewing quality scores in FastQC see Figure 3.

After removing the low-quality reads, the remaining short reads are mapped to a reference or assembled *de novo* into contigs [CADC10, MW11]. As *de novo* assembly for higher eukaryotes is very challenging because of the complexity of the transcriptome [MW11] and the differences in the expression levels of transcripts that usually vary by five orders of magnitude [RLSG12], in most cases *de novo* assembly is only done when there exists no suitable reference, or the reference is not of high enough quality.

Following the alignment of the reads, a common downstream analysis task is to identify the various genes and transcripts present in the sample and to quantify their expression levels [MW11, VHPV13, RPTP11, LFJ11]. The expression levels can then be compared between samples to study the changes in gene and transcript expression between two or more conditions [VHPV13, RKL$^+$13, THS$^+$13, RMS10,

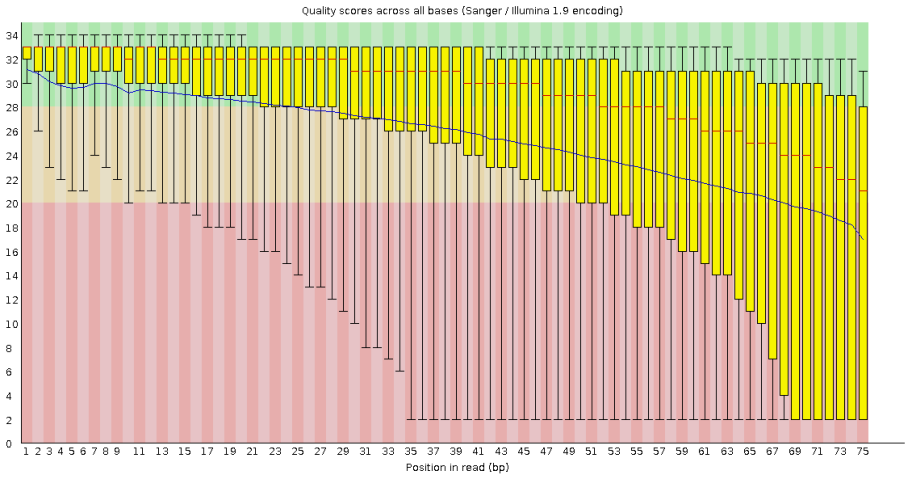


Figure 3: Screenshot of quality scores in FastQC. This example has bad base qualities towards the end of the reads, which suggests that trimming the ends of the reads might be appropriate before attempting to map them to the reference.

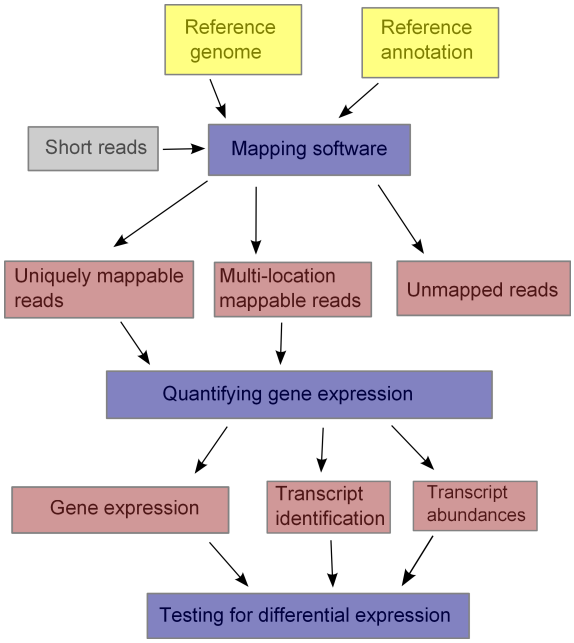


Figure 4: Example of RNA-seq analysis pipeline. Figure based on [CADC10].

GHR12].

In addition to transcript prediction, the aligned data can also be examined to detect fusion genes [SHP+10, GLJ+11, LCSM11] or allele-specific expression [TSG+11, RAW+11, PFFS13], to name just a few other examples for analysing RNA-seq data. One example of an RNA-seq data analysis pipeline is shown in Figure 4.

3.2.1 Read mapping

While mapping one read to a reference using for example BLAST or BLAT may be trivial, the sheer number of the reads (from tens of millions to a few billions) makes such strategies infeasible [WL09]. As this step is critical for many RNA-seq applications, much research has gone into developing efficient mapping algorithms [GFP+11, LH10].

For efficient searching, the short read mapping algorithms create auxiliary data structures called indexes from the reference, the reads or both [GGGT11, LH10]. Depending on the properties of the index, most algorithms can be classified to two types [GGGT11, LH10]: those using *hash tables* (for example, Bfast [HMN09], MAQ [LRD08] and GSNAP [WN10]) and those using compressed variants of *suffix trees* (for example, Bowtie [LTPS09], BWA [LD09] and readaligner [Mäk10]).

A hash table for a reference sequence is created by indexing all the k-mers, that is, segments of the reference of length k, so that the hash values of these k-mers act as keys that point to all the positions where the given k-mer occurs in the reference [LH10]. An example of a hash table is shown in Figure 5).

The reference sequence from which the hash table is created can be either the reference genome or the reads, while the query sequence is the other [LH10]. However, as the number and length of the short reads has been steadily increasing with the improvements in the sequencing technology, the approaches that index the reads have become infeasible for many applications because the size of the index grows too large [LH10].

The query sequence can then be split into seeds of length k and the hash table queried with these seeds to find the positions for exact match in the reference [LH10]. When a match for the seed is found, it can be extended to check if the whole query sequence matches the reference around the given position [LH10].

A suffix tree is a data structure that stores all the suffixes of a given string. The structure of the suffix tree allows for fast exact string matching [LH10] (see Figure 6 for an example). Inexact string matching in a suffix tree can be thought of as enumerating over all the possible positions for mismatches and doing an exact match search around them [LH10]. This causes the search to branch, of which an example for one mismatch case is shown in Figure 7.

However, suffix tree, as well as its close relatives suffix trie and enhanced suffix array, are impractical choices for indexing large genomes, as they require several

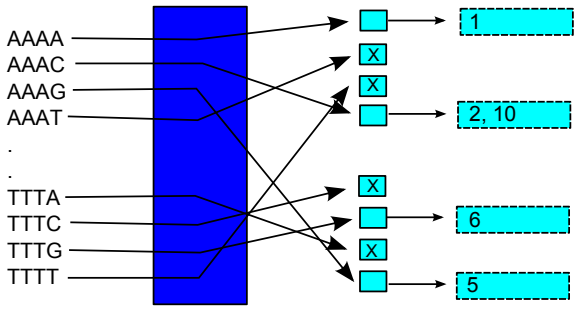


Figure 5: A hash table for string "AAAATTTTCAAAT" using 4-mers. All substrings of length 4 are hashed (blue box being the hashing function), and the hash values are used as pointers to lists containing all the positions where the 4-mer occurred. (All 4-mers are not shown in the figure.)

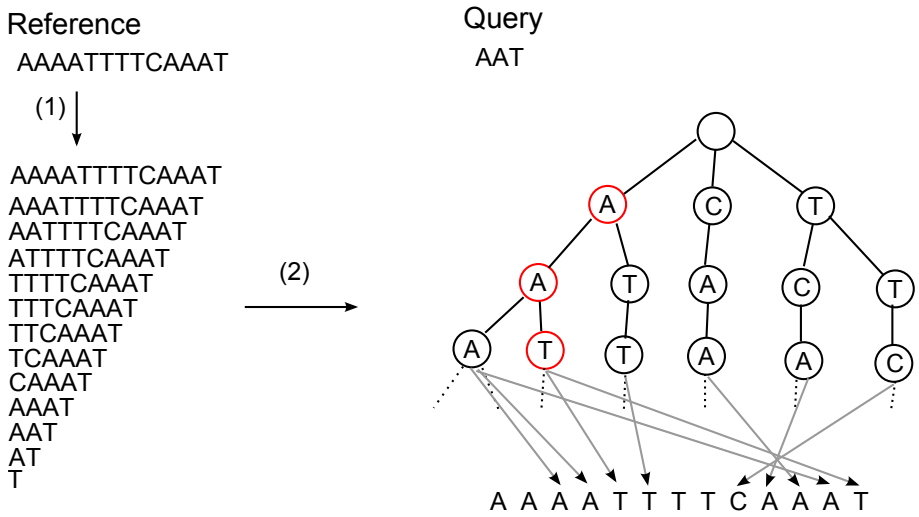


Figure 6: Searching for exact match for string "AAT" in string "AAAATTTTCAAAT" using suffix tree. All the suffixes of the reference string (1) are inserted into the tree (2). Each node points to the location(s) in the reference where the string spelled out by the path to the node ends. Exact matches for string "AAT" in the reference are found by finding the path corresponding to the string and checking the pointers to the reference positions (red-colored path, with end node pointing to positions 5 and 13).

bytes or more of memory for each base [LH10]. The invention of FM-index, a compact Burrows-Wheeler transform -based data structure, by Paolo Ferragina and Giovanni Manzini [FM00] improved the memory efficiency of suffix-tree variant -based approaches to 0.5-2 bytes per base [LH10]. In principle many algorithms using a suffix tree work with a suffix trie, a suffix array or an FM-index, and vice

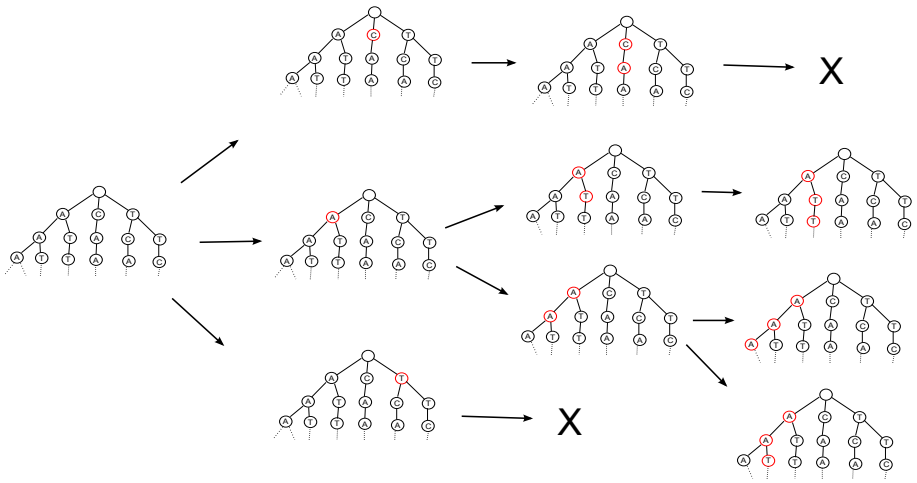


Figure 7: Searching for match for string "AAT" in string "AAAATTTTCAAAT" while allowing one mismatch. Search branches at every node as long as there are allowed mismatches left. Branches are rejected once the mismatch limit is exceeded (marked with X). A total of three matches are found for string "AAT": "AAT", "AAA" and "ATT".

versa [LH10].

For more information about hash tables see for example [MS08], and for more information about suffix trees see for example [Gus97].

Compared to hash tables, algorithms based on suffix trees have the advantage that identical sub-sequences in the reference collapse to a single path in the tree [LH10, GGGT11]. This means the alignment only needs to be handled once, whereas in the hash based approaches every exact match to the seed has to be checked separately for complete alignment [LH10]. This simple fact makes suffix tree based approach many times faster, when no mismatches are allowed [GGGT11].

The performance of suffix tree based approaches decrease when more errors are allowed in the search, as the search space grows exponentially [GGGT11, YMW+12]. Hash based methods that require an exact match for the seed are not affected in this way [GGGT11]. The speed of extending the alignment decreases very little as the number of errors allowed increases. However, aligners based on suffix trees can use various heuristics to limit the search space and thus make the alignment process faster at the expense of possibly missing some alignments [LD09, LTPS09, YMW+12].

Independent of the implementation details of the index, alignment software can

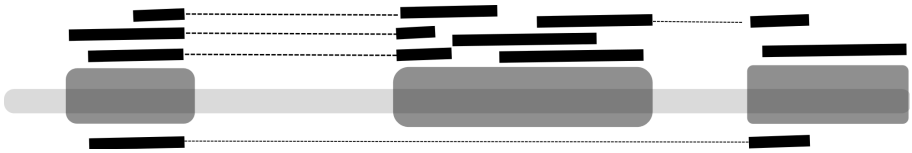


Figure 8: When aligning an RNA-seq read to a reference genome, some reads will map fully, but others will contain a splice junction, resulting to a gap in the alignment (dark gray blocks are exons, black blocks are reads, and dashed line signifies a gap in the alignment).

be classified as either *unspliced read aligners* or *spliced read aligners* [GGGT11, CADC10]. Unspliced read aligners simply align the read to the reference without allowing large gaps, and as such are suited for aligning DNA reads to the genome or aligning RNA reads to the transcriptome [GGGT11]. Spliced read aligners on the other hand allow gaps and are suited for aligning RNA reads to the reference genome [GGGT11].

In this thesis I concentrate on the problem of aligning reads to the reference genome, therefore my software comparison will include only spliced read aligners. However, many of the spliced read aligners are built on top of unspliced (core) aligners, so that the spliced read aligner modifies the reads by trimming or segmenting them in different ways and uses the core aligner to map these segments to the reference genome [GGGT11, CADC10]. While mapping, some mismatches or indels (insertions or deletions) have to be allowed because of genomic variation as well as sequencing errors, but there is a fine balance between allowing as many reads as possible to map while keeping the quality of the alignments high [CADC10].

Repetitive regions, which make up nearly 50% of the genomes of higher eukaryotes (for example human and mouse), present an added difficulty, as the reads can map perfectly to multiple positions in the genome. [WL09, RLSG12]. This difficulty can be partially alleviated with paired-end reads [WL09, ORY10, RLSG12]. If one read of the pair maps to a repetitive region, but the other does not, the uniquely mapping read gives an anchor to find the correct location of its mate [WL09, ORY10]. Reads that align to the proximity of the candidate alignment positions can also be used to locate the correct position, as the flanking areas should have roughly the same amount of aligned reads [BCZF12].

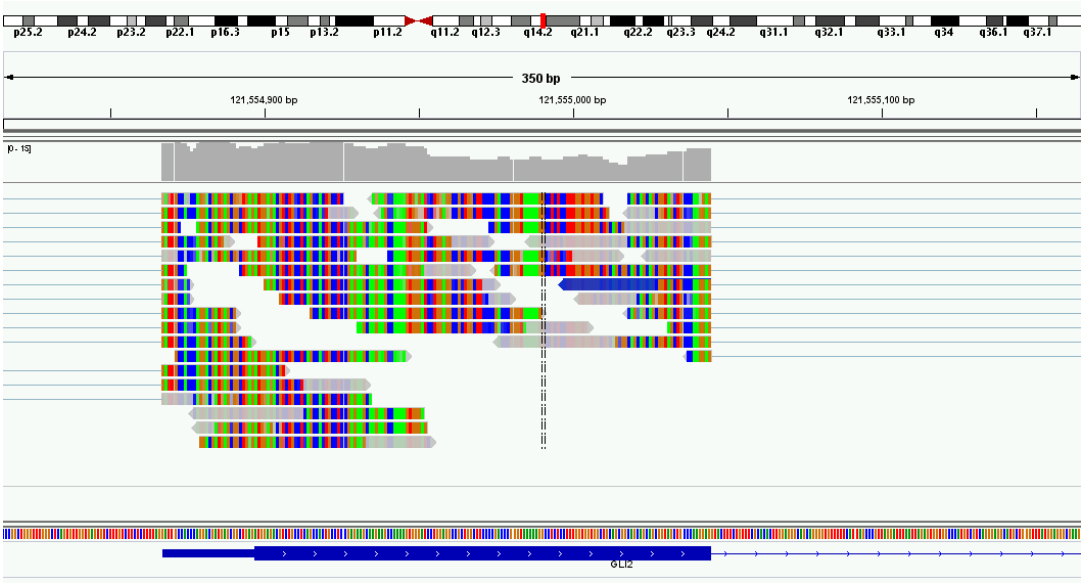


Figure 9: Screenshot of genomic view in IGV genome browser. Each colored block is one read. Chromosome positions are shown at the top, and known RefSeq genes at the bottom.

3.2.2 Transcript identification and other downstream applications

After aligning the reads to the reference, a common next step in the RNA-seq analysis is to identify the different features present and to quantify their expression levels [VHPV13, MW11, CADC10, RLSG12]. Genome browsers, for example UCSC Genome Browser and IGV (see Figure 9), allow for visual inspection of the alignments, which can be a useful starting point for inspecting specific features based on annotation and mapped reads, as well as checking the alignment quality [CADC10]. However, visual inspection can only give a qualitative picture of the feature of interest and the sheer amount of data makes it difficult, if not impossible, to find all the relevant features.

Computationally quantifying the expression level of a single non-overlapping exon (i.e. an exon that does not share any genomic positions with another exon in either strand) from RNA-seq data aligned to the reference genome is in principle a simple process. For example, the number of reads falling between the start and end of each exon can be counted, and this value can then be normalized by dividing by the length of the exon [VHPV13, YMW$^+$12]. But as transcripts can be formed from various combinations of the exons (see Chapter 2), identifying which exons belong to which transcript(s), and consequently dividing the reads correctly among

them is a much more difficult task [VHPV13, YMW+12]. This task is made even more difficult by the presence of multimapping reads, as the uncertainty for their true location is high. Once the transcripts have been identified and their expression levels quantified, the expression level of the gene can be calculated as the sum of the expression levels of its transcripts [WWZ10].

Because of the ease of calculating exon expression, and the gene expression level following from the transcript expressions levels, a lot of the software development has gone into identifying transcripts and/or quantifying their expression levels [RPTP11, LFJ11, LJB+11, TKRM13b, LD11]. At their core, many transcript identification and quantification methods solve some variation of a graph problem, whether it be an overlap graph with each read as a node (e.g. Cufflinks [RPTP11]); or splicing graph (also known as connectivity graph) with each node being a continuous stretch of DNA and each edge derived from overlaps or spliced read alignments (e.g. Isolasso [LFJ11], Traph [TKRM13b]). The identification and quantification problem then reduces to finding paths in this graph and splitting the coverage of nodes and edges between the paths, according to various error models (see Figure 10 for an example of a splicing graph).

After identifying all the transcripts present in the samples, one of the fundamental tasks in RNA-seq analysis is to identify a set of transcripts that are differentially expressed between two or more samples [RKL+13, THS+13, GHR12, RMS10]. Because of the variation from both biological and technical sources, statistical algorithms are required for detecting differential expression based on the identified transcripts and the aligned read data [RKL+13, THS+13]. Some of the popular software for detecting differential expression are cuffdiff [THS+13], edgeR [RMS10] and DEseq [AH10], which all have their own strategies for normalizing the count data, modeling gene expression statistically and testing for differential expression [RKL+13].

Instead of identifying and quantifying the transcripts present in the sample, the aligned read data can be used to for example find fusion genes [SHP+10, LCSM11, GLJ+11] or study allele-specific gene expression [RAW+11, PFFS13].

Fusion gene candidates are often identified by searching for paired-end reads where the mates, that is, the reads sequenced from the opposite ends of the same fragment, map to different genes [SHP+10, LCSM11]. While the software are designed to attempt to filter out likely mismappings [SHP+10, LCSM11], it is clear that incorrect alignments make it more likely for the software to give false positive results.

Correct alignment is also important for studying allele-specific gene expression.

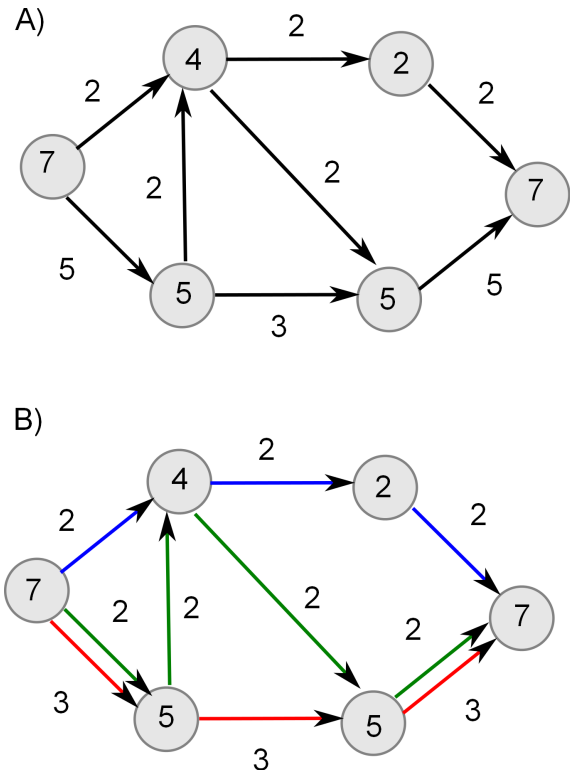


Figure 10: Transcript identification and quantification as graph problem. A) The coverages of nodes (exons) and edges between them (reads spanning two consecutive exons). B) Splitting the node and edge coverages to paths that best explain the graph.

As a common first step for the analysis of allele-specific expression is to identify the different alleles and construct a polymorphism-aware diploid reference from them [PFFS13, RAW+11], incorrect alignments can yield incorrectly constructed reference.

4 Materials and methods

4.1 Test data

As no ground truth is known for real data, the first step in any performance evaluation is to use simulated data where ground truth is known and can be compared against. As Illumina HiSeq is the currently most commonly used platform for RNA-seq [VHPV13], I decided to test the software with the type of reads this sequencer produces: 50 bp, 100 bp and 150 bp long reads in both single-end and paired-end modes. Based on Illumina MiSeq sequencer, I also added 250 bp long reads (both single-end and paired-end) to the test setup.

For simulating data for this comparison I chose the UCSC human genome version hg19 (NCBI GRCh37) chromosome 2 and the corresponding hg19/GRch37 version of the annotation of both coding and non-coding transcripts for this chromosome. Chromosome 2 contains 3,831 annotated genes, with total of 13,391 transcripts. To allow for better sampling of long paired-end reads, I discarded all the transcripts that had the length of less than one kilobase. After this filtering there remained a total of 4,839 transcripts.

I used RNASeqSimulator[6] to draw expression levels for all the chosen transcripts from a log-normal distribution with mean of -4.0 and standard deviation of 1.0. Based on these expression levels I created two single-end data sets (one million reads each) and two paired-end data sets (2x one million reads each) for each read length. Depending on the read length, this corresponds to 3.5x-17.5x average depth of coverage for single-end and 7x-35x average depth of coverage for paired-end. First data set simulated the ideal conditions with no sequencing errors, and the second data set simulated the conditions where each base had 1% chance to be miscalled.

For simulating the paired-end data sets, I set the fragment mean size parameter to 200 bp for 50 bp reads, 300 bp for 100 bp reads, 400 bp for 150 bp reads and 500 bp for 250 bp reads. This equals to 100 bp between the mates for 50 bp, 100 bp and 150 bp data sets. Illumina recommends fragment sizes no longer than 500 bp, therefore the distance between mates for the 250 bp paired-end reads is zero. I set the fragment size standard deviation to 20 bp for all the data sets.

I also tested the software with real RNA-seq data set generated from human embryonic stem cells (Caltech RNA-Seq track from the ENCODE project, NCBI SRA

[6] http://alumni.cs.ucr.edu/~liw/rnaseqreadsimulator.html

accession number SRR065504). This data set consists of 50 million 75 bp paired-end reads sequenced with Illumina Genome Analyzer II.

4.2 Existing mapping software

In this section I will describe the algorithms behind the software chosen for this comparison: Tophat [TPS09] (version 2.04), SpliceMap [AJL+10] (version 3.3.5.2), MapSplice [WSZ+10] (version 2.1.5), SOAPsplice [HZL+11] (version 1.9) and SHRiMP2 [DDL+11] (version 2.2.3).

Tophat, SpliceMap and MapSplice use a "core" genomic aligner to do the alignment of the reads or their segments. Their current implementations use Bowtie [LTPS09], but any core aligner is theoretically suitable for this purpose. Both Bowtie and SOAPsplice use FM-index for aligning the reads. SHRiMP2 uses a hash-table-based approach.

4.2.1 Tophat

Tophat's [TPS09] mapping algorithm consists of three main phases: mapping all reads to the reference, assembling putative exons and mapping the remaining unmapped reads to concatenations of the putative exons (see Figure 11). In addition to the mappings BAM file, Tophat provides a BED file describing the resulting junctions (see Appendix 1 for details on the file formats).

In the first phase, all the reads are mapped to the genome using Bowtie, a short read aligner created by Langmead et al. [LTPS09]. For each read Bowtie reports one or more alignments (default: 10 alignments) with no more than a given number of mismatches (default: 2 mismatches). Bowtie suppresses the alignments for reads with more than given number of possible alignments. The reads that did not map in this stage are referred to as *initially unmappable* (IUM) reads.

In the second phase, Tophat uses MAQ [LRD08] to assemble a consensus of all regions covered by the mapped reads. This assembly contains both called bases and their corresponding reference bases. In the case of conflict, Tophat will choose the reference base. The islands of continuous sequences are then inferred to be putative exons.

However, the putative exons are likely to be missing a small amount of sequence on both ends because most reads mapping near the ends of exons will probably

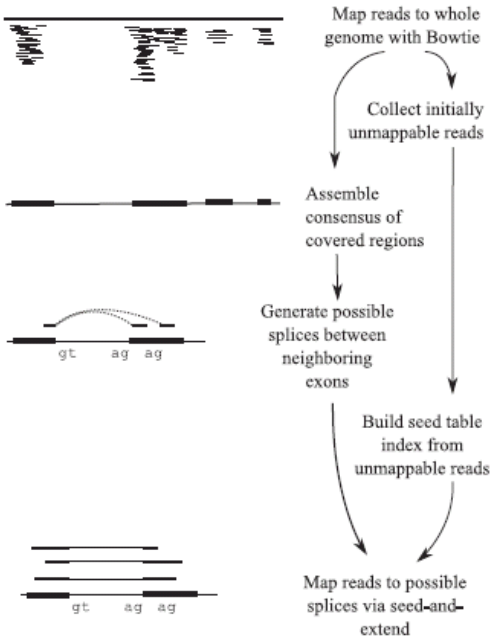


Figure 11: Schematic representation of Tophat workflow. Whole RNA-seq reads are mapped to the reference, putative exons are inferred and potential splice sites are created by enumerating over all combinations of the putative exons. Initially unmapped reads are indexed and matched against the potential splice sites. [TPS09]

also contain a splice. Tophat will include a small amount of flanking sequence from each end to capture the missing bases as well as the donor and acceptor sites of the flanking introns. Because the assembly might have gaps within exons, Tophat will also merge putative exons that are very close together (default $<$ 70 bp).

In the final step, Tophat considers all pairings of donor and acceptor sites in the putative exons that could form canonical (GT-AG, or their reverse complements) introns, with intron length within specified bounds (default longer than 6 bp and shorter than 20,000 bp). The user can decrease the maximum intron size to decrease the running time, but this might cause Tophat to miss some alignments.

To map the IUM reads to the junction candidates, Tophat creates seeds by concatenating k bases from each side of the junction (default: 6 bp). These seeds are then queried against an index table where each $2k$-mer is associated with the reads whose high-quality region at the 5'-end contain the $2k$-mer (see Figure 12). The default length s of the high-quality region is 28 bp. Increasing s will improve sensitivity and lowering it will decrease running time and possibly reduce sensitivity. Increasing k will also decrease running time but might cause Tophat to miss alignments in lowly

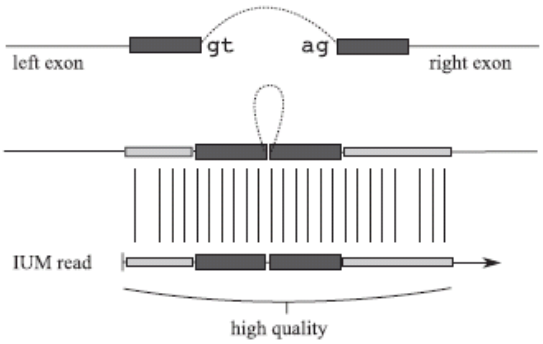


Figure 12: Tophat's seed-and-extend strategy. A small amount of sequence is combined from two exons and the resulting seed is used to query the index made from initially unmapped reads. If the seed matches, the complete alignment to the pair of exons is checked. [TPS09]

expressed genes.

By default, Tophat filters alignments based on read coverage to reduce the number of false positive junctions. It does this by checking the average read coverage depth of flanking regions and comparing this to the number of reads crossing the junction. If the number of alignments crossing the junction is less than 15% of the more deeply covered flank, the alignment is not reported. This behavior can be turned off by the user.

4.2.2 SpliceMap

Short reads with lengths of 25-36 bases in the early days of RNA-seq experiments were not suitable for *de novo* detection of exon-exon junctions, so Tophat's clustering to create exon islands was a natural way of approaching the problem of spliced reads. SpliceMap [AJL+10] takes advantage of the increase in read sequence lengths (from subsequent improvement in sequencing technologies) to directly map the exon-exon junctions without first inferring the putative exons.

At the time of SpliceMap's creation, most second-generation sequencing platforms could create reads of at least length 50-100 bp. SpliceMap's approach exploits the property that a read containing one splice junction will most likely have a match in the reference genome that is at least half of the read's length. With half the read being at least 25 bp long, it can with high probability be reliably aligned to reference genome.

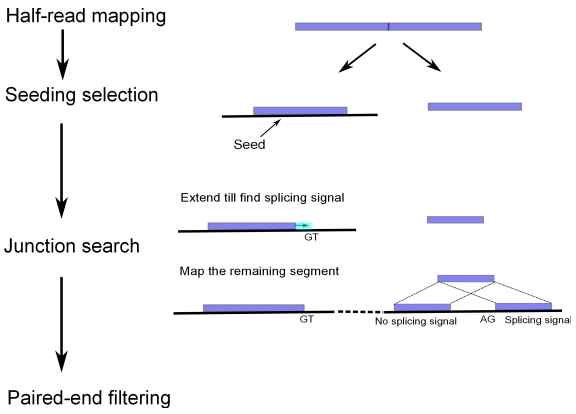


Figure 13: Splicemap's workflow consists of four phases: half-read mapping, seeding selection, junction search and paired-end filtering. Based on [AJL+10]

SpliceMap's algorithm has four parts: half-read mapping, seeding selection, junction search and paired-end filtering (see Figure 13). Junction search consists of seed extension and locating the partner splicing site. In addition to these steps, SpliceMap infers the junctions present from the alignments, and also provides the user information for assessment of their reliability in the form of counts of reads supporting the junction and the flanking exons.

In the half-read mapping step SpliceMap splits 50 bp reads into two halves and maps them using an unspliced read-mapping tool. If the reads are longer than 50 bp, they might contain multiple splice junctions. In that case SpliceMap will split the reads into several overlapping 50 bp reads (for example, split 100 bp into 1-50, 25-75 and 50-100). These partial mappings will be filtered in the post-processing step to verify that the mappings are consistent.

Next SpliceMap picks the best seed alignments. This is done on a chromosome by chromosome basis. Unique mappings are preferred, but multimappings are also accepted as long as they are at least 400,000 bp away from each other. This filtering of multilocation alignments is done to prevent false splice predictions. Selected seeds are extended base by base until a splicing junction signal (canonical dinucleotide AG-GT or its reverse complement) is found, or the number of allowed mismatches is exceeded.

After seeding selection and extension, SpliceMap tries to find the remaining segments within a user specified distance (default 400,000 bp) by querying a chromosome-wide hash table for the first 10 nucleotides of the segment and extending the alignment from there. Naturally, the segments must be at least 10 bp long for this process. In

addition, the mapped segment must reside next to the canonical dinucleotides. If the segment mapped to multiple locations, the alignment is discarded to avoid false positives.

If the reads are paired-end, junction detection specificity can be improved. Read pairs that are mapped more than 400,000 bp away from each other, or in direction or positional order conflicting with the experimental design, can be discarded. Also pairs where only one read qualifies as a good hit, i.e. either an *exonic hit* (positions of segments differ by exactly one segment's length), *extended hit* (seed was extended beyond 40 bp, but less than 50 bp) or *junction hit*, as defined above, can be discarded.

4.2.3 MapSplice

Unlike the software introduced previously, Tophat and SpliceMap, MapSplice [WSZ+10] does not require the canonical dinucleotides or limit the search by a set intron length. Therefore it is able to find non-canonical junctions as well as other novel splicing events. MapSplice works in two phases: a *tag alignment phase*, where it maps the reads to the reference genome, and a *splice inference phase*, where the junctions created during the alignment phase are analyzed to choose the junctions with highest overall quality and confidence.

In the tag alignment phase (see Figure 14), the reads, or tags as the authors call them, are first partitioned into segments $t_1, ...t_n$, that are no longer than half of the read length. Typically the length of a segment is 20-25 bp for reads of length 50 bp or more. These segments are then aligned with any unspliced short read aligner, for example Bowtie [LTPS09]. Segments failing to map might cross a splice junction or they might have exonic alignments with more errors than the allowed threshold. Generally, if the segments are of length k and the minimum exon length is at least $2k$, at least one segment out of every pair of consecutive segments should map to the reference. Therefore, the neighbors of the unmapped segment can be used to localize its alignment.

MapSplice splits the unmapped segments to two categories, *single anchored* (only one neighbor mapped) and *double anchored* (both neighbors mapped). For double-anchored spliced alignment, MapSplice checks all the possible splice positions between the two neighboring alignments. Each candidate splice alignment is given a score based on the Hamming distance between the read sequence and corresponding genomic sequence. The alignment(s) with minimum score is then reported.

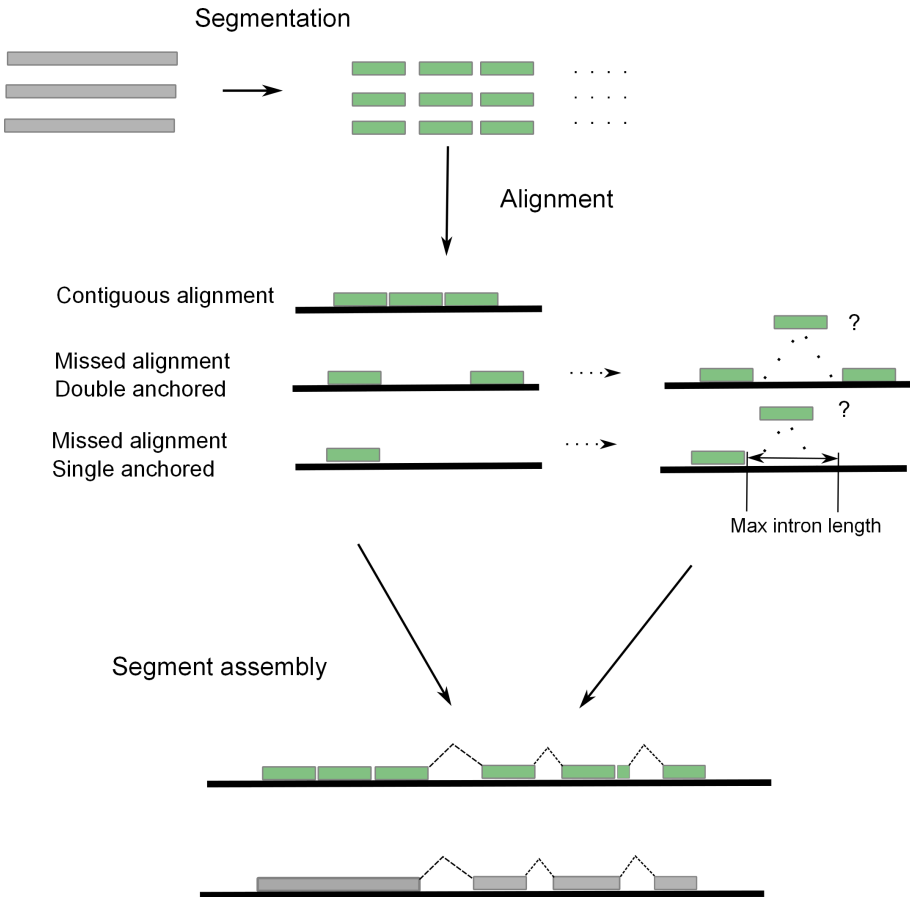


Figure 14: Tag Alignment phase consists of segmentation of the reads, mapping the segments and assembling the alignments of the segments to form continuous alignment for the whole read. Based on [WSZ+10].

Single anchored spliced alignments have a boundary only for one side, so the extent of the search is limited by the maximum intron size D (default: 50,000 bp). The alignment is found by searching for an h-mer suffix or prefix, respectively for upstream and downstream anchor, of the unmapped segment within a sliding window of length D. To speed up the search, MapSplice resolves all the single anchored spliced alignments with one pass of a sliding window over the reference.

If the transcripts contain an exon or exons shorter than $2k$ bp, where k is the length of the segments, two adjacent segments mapping to this area might both contain a splice junction. For exons shorter than k bases, a single segment might contain multiple splice junctions. MapSplice divides the sequence S, consisting of one or two missed segments between anchors t_i and t_j into a set of h-mers and indexes S with them. These h-mers can all be searched at once in the same way as with

the single-anchored spliced alignments. When a match is found, double-anchored spliced alignments between t_i and the 5'-site of the h-mer, and between 3'-site of the h-mer and t_j can be performed. If the exon is no shorter than $2h$, this method is guaranteed to find the spliced alignment. However, reducing h to shorter than 6-8 bp will increase the probability of the segment randomly aligning to the reference.

Finally the segments must be assembled into complete read alignments. When each segment aligns uniquely, the assembly is easy. But whenever any segment t_i has mapped to multiple locations, all the possible combinations must be considered for the best overall alignment. Fortunately most combinations can be ruled out based on the order and orientation of the consecutive segments. In this step MapSplice first re-checks all the non-consecutive segments' splice sites using a double-anchored spliced alignment method, and corrects possible inaccuracies caused by error tolerance in the segment mapping. Then a *mismatch score* is calculated for all valid assemblies, i.e. assemblies that conform to expected order and orientation of the consecutive segments. This mismatch score is the Hamming distance between the tag and the reference genome, modified so that the base call qualities (when available) are taken into account. If this mismatch score is lower than or equal to the allowed number of mismatches, the alignment candidate is accepted.

In the splice junction inference phase (see Figure 15), MapSplice evaluates all the proposed junctions for quality based on two statistical measures: *anchor significance* and *entropy*. Then these junction quality values are used with the alignment quality values to determine the best alignment for each tag. The anchor significance measures how long the anchors are on each side of the junction and the (Shannon) entropy measures the diversity of the splice junction positions in the set of reads crossing the junction. The final score for each junction is the weighted sum of anchor significance, entropy and average alignment quality for reads crossing the junction.

4.2.4 SOAPsplice

Like Tophat, SOAPsplice [HZL+11] first tries to map the reads fully to the reference genome (Step 1 in Figure 16) and collects the initially unmappable reads (IUM reads). In this first step, either at most five mismatches or one continuous gap that is no longer than two bp can be allowed (user can specify lower values). The gap can be either an insertion or a deletion, depending whether it is on the query or the reference sequence. SOAPsplice gives preference to ungapped hits, as single nucleotide polymorphisms are much more frequent than small indels. Given multiple

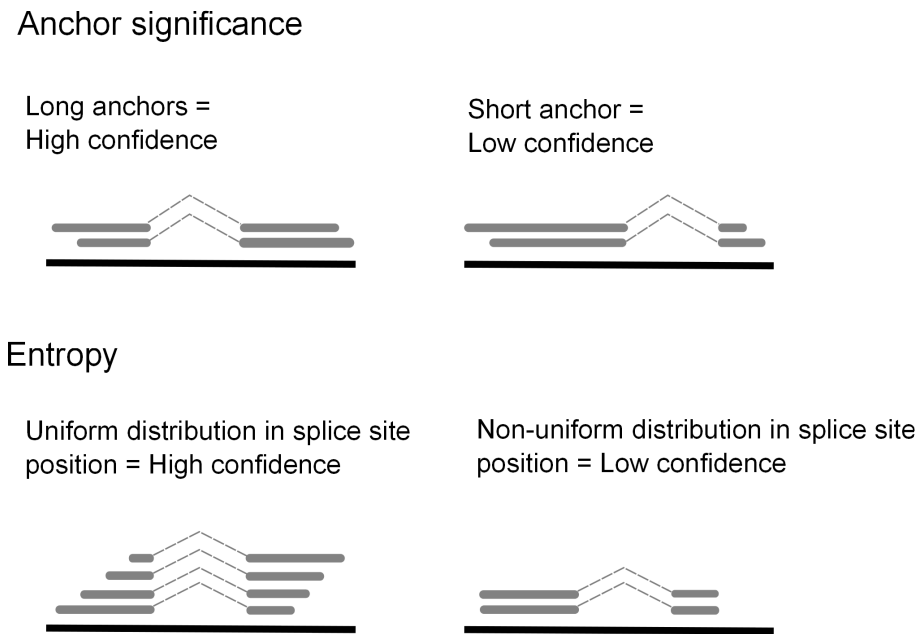


Figure 15: Two measures for quality of the junction. Longer anchors on both sides of the junction give more confidence to it being correct, as a short segment can randomly align to multiple positions. Random sampling along the length of transcript should result in the position of the splice site within the tag following a uniform distribution. The higher the entropy, the closer to uniform the distribution is, which makes it more likely to be a true splice junction. Based on [WSZ+10].

mappings with either mismatches or indels, the one with the least mismatches or smallest gap is chosen. If a read did not map in this stage, SOAPsplice trims several bp from the 3' end, as the call quality of these bases might be low, and attempts to align the read again. As with Tophat, reads that failed to map as a whole, with the trimming applied, are called initially unmappable (IUM) reads.

Next SOAPsplice attempts to find a spliced alignment for the IUM reads. In the first step of this phase, SOAPsplice searches for the longest 5' segment of a read that can be mapped to the reference. However, if the remaining segment is shorter than the threshold (default: 8 bp), the alignment is not accepted. No more than one mismatch is allowed on each segment, and gaps are not allowed at all within a segment. Distance between the segments is also limited to 50-50,000 bp, and the canonical dinucleotide pair "GT-AG", "GC-AG" or "AT-AC" is required at the exon-intron borders of the segments. Priority is given to the alignments with the dinucleotide pair in that order. Additionally, either one of the segments has to map uniquely, or each of the segments can have at most three hits to the reference. If

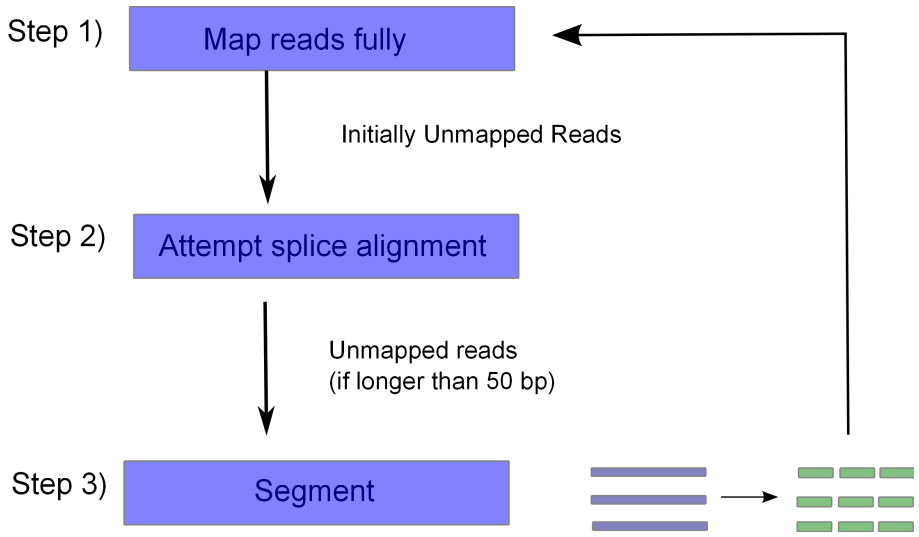


Figure 16: The workflow of SOAPsplice. In step 1, complete reads are mapped to the reference genome and in step 2 initially unmapped reads are aligned to reference by using two-segment alignment strategy. For reads longer than 50 bp, the remaining unmapped reads are split into segments that are no more than 50 bp long and steps 1 and 2 are applied on these segments. Based on [HZL+11].

there are multiple valid pairs of segments, the combination with the shortest distance is selected.

The approach of mapping a read in two segments is only able to find at most one junction, so SOAPsplice incorporates an additional alignment step for reads that are longer than 50 bp. This is important because at least some of the reads of that length are likely to contain multiple splice junctions. For reads that are shorter than 100 bp, SOAPsplice splits them into two "sub-reads" of equal size. For reads longer than 100 bp, 50 bp long segments are taken from the 5' end of the read till the remaining segment is between 50 bp and 100 bp long, then the remaining segment is divided in two. The search for intact alignment, and spliced alignment if intact alignment cannot be found, is then done for these sub-reads as above. If at least two-thirds of the sub-reads can be aligned uniquely, SOAPsplice checks whether the sub-read alignments are consistent in the position and orientation of the original segments, and if so, concatenates them to create a complete alignment.

The second part of SOAPsplice's workflow consists of filtering the junctions inferred from the alignments to remove false positives. If paired-end or mate-pair information is available, the first step in the filtering is to ensure that the aligned positions and orientations of the pair are consistent with the experimental design. For example,

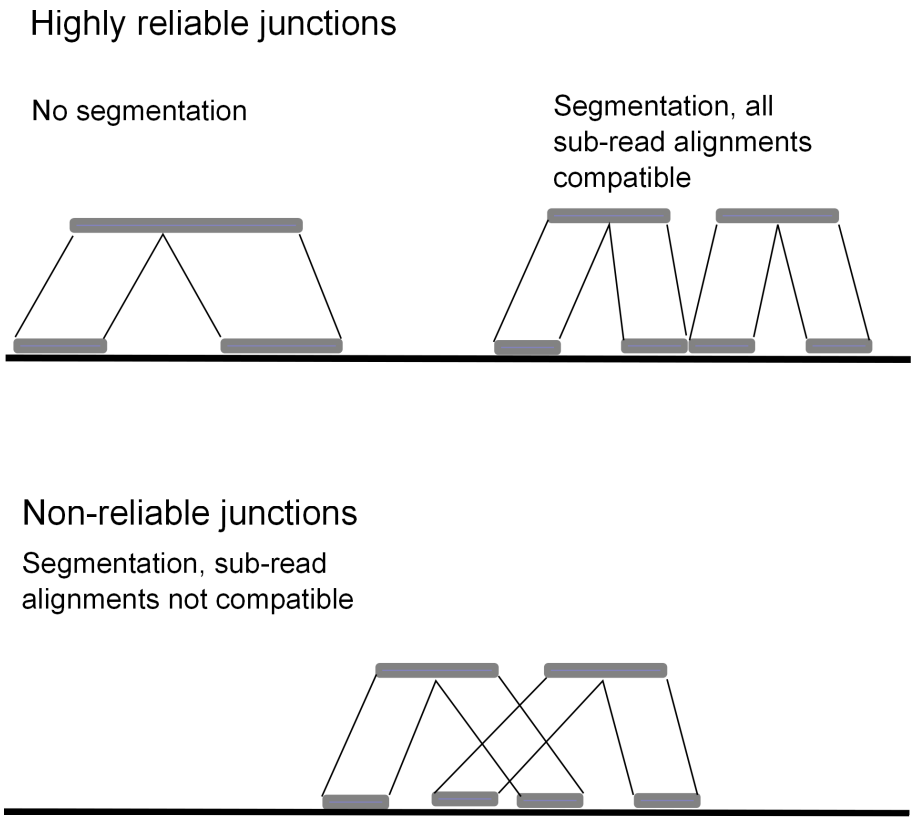


Figure 17: Junctions inferred from alignments are classified as either highly reliable or non-reliable. Based on [HZL+11].

if the experimental design is sequencing from both ends inward, and the first read contributing to the junction is in forward orientation, the second read of the pair needs to be in reverse orientation.

The second filtering step categorizes the junctions based on whether the reads were segmented to "sub-reads" or not. The junctions inferred from alignments without segmentation to sub-reads are considered to be highly reliable and are reported without filtering as are junctions inferred from segmented reads where all the sub-reads alignments are compatible with each other, that is, they mapped to the genome in correct positional order and orientation and with all the segments connecting. The remaining spliced alignments are those where the reads were segmented, and some of the sub-reads were incompatible (see Figure 17). For these alignments SOAPsplice requires that the number of reads supporting the junction should be more than 25% of the average number of reads supporting the highly reliable junctions.

4.2.5 SHRiMP2

Unlike the four suffix-tree-based software introduced previously, SHRiMP [RLD+09, DDL+11] (for SHort Read Mapping Package) uses hash-table-based approach. The original SHRiMP package [RLD+09] indexed the reads, whereas the newer version SHRiMP2 [DDL+11] indexes the genome. Indexing the genome instead of the reads decreases the running time, and allows the use of paired-end mode and the use of multi-threading.

SHRiMP2 employs the basic hashing idea of finding matches for short subsequences of the reads, called seeds, in the genome index and then attempting local alignment in the areas around the seeds, with a few modifications. Instead of requiring a completely exact match for the seeds, SHRiMP2 uses so-called *spaced seeds*. Spaced seeds have predetermined positions where mismatches are allowed. They are usually represented as strings of 0s (position may have mismatch) and 1s (position must match). The *weight* of the seed is defined as the number of 1s in the string.

As mentioned in Chapter 3.2.1, hash-table-based approaches require a lot of memory, and SHRiMP2 is no exception. SHRiMP2's genome index requires $k \times (4^w \times 12 + n \times 4)$ bytes, where n is the length of the genome, and k is the number of seeds of weight w. With the default parameters(k=4, w=12), this results to memory requirement of 48 GB for the index of the human genome (hg19). However, in practice this does not pose a problem as SHRiMP2 offers utilities to split the genome into pieces that fit into target RAM size. Reads can then be matched against each piece index sequentially or in parallel.

Unlike older hash-based alignment tools (for example BLAST [AGM+90]) that search for local alignment based on a single seed, SHRiMP2 requires multiple matches within a window of the genome before commencing a local search. This requirement allows SHRiMP2 to use shorter seeds with smaller weights (i.e. allowing more mismatches at the predetermined positions in the seeds). This follows the idea of q-gram filters introduced by Rasmussen *et al.* [RSM06].

SHRiMP2 uses Smith-Waterman algorithm [SW81] for rapid alignment of the read and the area around the seed mapping location. To speed up the computation, SHRiMP first computes only the score for each alignment, and not the alignment itself, and stores the top hits for each read. As SHRiMP supports both letter-space and color-space, the final alignment phase depends on the choice of the sequencer.

While regular Smith–Waterman algorithm works for letter-space, the nature of color-

space (see Chapter 3.1.2 for details) introduces certain complexities for direct application of the Smith–Waterman algorithm. Translating a color-space read to letter-space has the problem that any mismatches in the read would cause every base after the error to be mistranslated. Conversely translating the genome from letter-space into color-space leads to any given reference subsequence no longer being unique, as a string of colors can code for several different letter sequences, depending on the preceding base pair. To solve these issues, SHRiMP uses a dynamic programming approach to align all four possible translations of the read to the genome simultanously. Moving from one translation to another mid-read is allowed by paying the "crossover" cost, which is equal to the penalty for a sequencing error.

As the final step after aligning all the reads, SHRiMP2 calculates mapping confidence statistics for every read. These statistics measure the probability that the read aligned to certain position by chance and the probability that the read was generated by the genome at that position while taking into account observed variations and error rates. For paired-end data, SHRiMP2 also calculates the probabilities that the alignment locations of the mates were paired by chance by using the observed distribution of insert sizes in the library.

4.3 SpliceAligner

SpliceAligner is the software I implemented as part of the thesis work. In this section I will first give a general overview of the software, then describe the algorithms used in more detail in Section 4.3.1 and the implementation details in Section 4.3.2.

SpliceAligner's approach is based on splitting the reads into subsegments and attempting to map them, but it takes a slightly different approach than SpliceMap, MapSplice and SOAPsplice. SpliceAligner's algorithm has three main parts: (i) mapping full reads and finding seeds, (ii) mapping spliced reads and (iii) creating junctions and using them to map yet unmapped segments (Figure 18). SpliceAligner also offers quality controls such as trimming the reads as pre-processing, and various filtering criteria as post-processing.

In the first phase, SpliceAligner, like Tophat and SOAPsplice, attempts to map the reads fully to the reference genome. Reads that failed to map fully are then recursively split into smaller segments, till one or more segments map, or the minimum segment size threshold is reached.

In the second phase, aligned segments, or seeds, are extended base by base until the

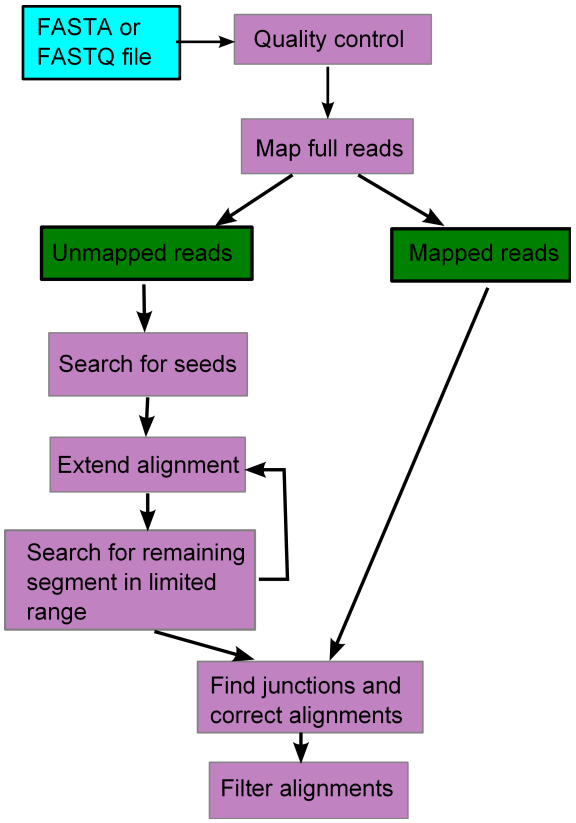


Figure 18: SpliceAligner's workflow. Reads are first mapped wholly in the reference, then unmapped reads are split into segments (seeds), seeds mapped to the reference, extended maximally and then the remaining segments are searched for in the limited range. Extension and remaining segment search are repeated as many times as necessary. After alignment phase, junctions are inferred from spliced alignments, and the junctions are then used to fine-tune alignments.

first mismatch (the current implementation does not support indels) is encountered and the extended mapping is saved for next phase. The extension is then continued until the error threshold is reached, and this alignment is saved as well. SpliceAligner also examines the sequence near the extension stop points to determine additional splice site candidates using canonical dinucleotides.

After extension, SpliceAligner attempts to map the remaining segment. If the segment does not map fully, it is split and the subsegment closest to the aligned part is used for attempting a new alignment. Remaining segment is split into smaller pieces until subsegment either maps or is too short to map reliably. If the smallest subsegment of acceptable length did not map, the alignment candidate is discarded. The mapped subsegment is then extended as above, and the process is repeated till

either the whole read is mapped or the remaining segment(s) are too short. This approach allows SpliceAligner to find an alignment consisting of arbitrarily many segments, as long as all the middle segments are sufficiently long.

SpliceAligner filters alignment candidates after every limited range query to prevent the number of candidates from growing too large with long ($>$ 150 bp) reads. Alignments that have canonical dinucleotides at the splice site are preferred. If no alignments for the given read have canonical dinucleotides at the splice site, then all the candidates are accepted.

In the third phase, SpliceAligner attempts to create a list of the supported junctions. The reads with too short segments at the ends that were saved in the previous phase are then compared to the junctions. If the border of the extended seed matches one side of the junction, the short segment is matched against the sequence on the other side of the junction. If the total Hamming distance for the alignment candidate is less than the error threshold, the alignment is adjusted to include the mapping of the segment.

4.3.1 Algorithms

In this section I will describe the algorithms for two of the main parts of SpliceAligner, limited range search and junction inference, in detail.

While SpliceAligner's strategy for searching for spliced reads might seem similar to SpliceMap and SOAPsplice, there are major differences in the search for the remaining segment after a part of the alignment is anchored via seed search. Whereas SpliceMap uses a genome-wide hash table and SOAPsplice searches for the segment in the whole index, SpliceAligner uses a novel limited range BWT-transformed index collection created by concatenating the chromosomes, splitting the result into pieces that overlap by read length or more, and creating an index for each piece (Figure 19). The overlap in the indexes makes sure that no alignment will be missed because it spans the breakpoint between indexes.

Two bit vectors that support rank and select operations in constant time are used in navigating the indexes, one with 1-bit set at the start of each index within the concatenation and one with 1-bit set at the start of each chromosome. For a bit vector B, operation rank-1(B, pos) gives the number of 1-bits occurring in $B[0, ..., pos]$, and operation select-1(B, n) gives the position of the n^{th} 1-bit. The pseudocode for navigating the indexes is shown in Algorithm 1 and Algorithm 2. An example of

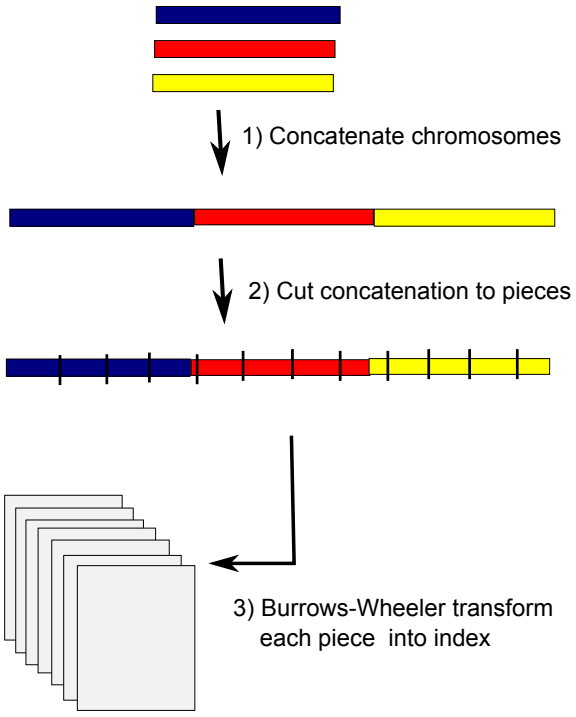


Figure 19: Creation of limited range index collection.

navigating the indexes is shown in Figure 20.

Algorithm 1 Map chromosome and genomic position to index

Input: Chromosome *chrom* and position in genomic coordinates *pos*

Output: Number of index

Require: Bit vector of chromosome starts in concatenation of chromosomes *chrom-starts*

Require: Bit vector of index start positions in concatenation of chromosomes *index-starts*

Require: List of the ordering of the chromosomes in the concatenation *chrom-order*

chrom-index = chrom-order.index(chrom)

abs-loc = chrom-starts.select-1(chrom-index) + pos

return index-starts.rank-1(abs-loc) - 1

By default SpliceAligner limits the search for the remaining segment to range of three indexes. The search is progressive, i.e. SpliceAligner looks first in the closest index, and only if there was no match found will it continue on to the next index.

In the junction inference phase, SpliceAligner first creates a chromosome-wide coverage table for each chromosome, and then matches all spliced alignments to the

Algorithm 2 Map position in index to chromosome and genomic position

Input: Index number *index-number* and position within index *index-pos*

Output: Chromosome and position in genomic coordinates

Require: Bit vector of chromosome starts in concatenation of chromosomes *chrom-starts*

Require: Bit vector of index start positions in concatenation of chromosomes *index-starts*

Require: List of the ordering of the chromosomes in the concatenation *chrom-order*

abs-loc = index-starts,select-1(index-number) + index-pos

chromosome = chrom-order[chrom-starts.rank-1(abs-loc)]

genomic-pos = abs-loc - chrom-starts.select-1(chrom-order.index(chromosome))

return chromosome, genomic-pos

coverage table to infer the junctions (Figure 21). For a junction candidate to qualify, it must have a certain number of bases covered in the flanking exons and optionally have certain number of reads spanning the junction.

4.3.2 Implementation details

The current implementation of SpliceAligner uses readaligner [Mäk10] as its core mapper.

For the seeding phase I followed the example of MapSplice's recommendation of 18-25 bp long segments and set the minimum segment size threshold default to 25 bp.

For the limited range search, following the example of the other software I set the maximum range default to be 300,000 bp. This equals to each index covering 100,000 bp by default. As there are 4^l different combinations for a sequence of length l, excluding repetitive regions, sequences at least 9 bp long are likely to map uniquely within three indexes ($4^9 = 262,144$bp), which sets the minimum segment size for the limited range search to 9 bp.

For the junction inference phase I set the default requirements for junction candidate to be accepted to 20 bp for the number of bases covered on each side and one read for the number of spanning reads.

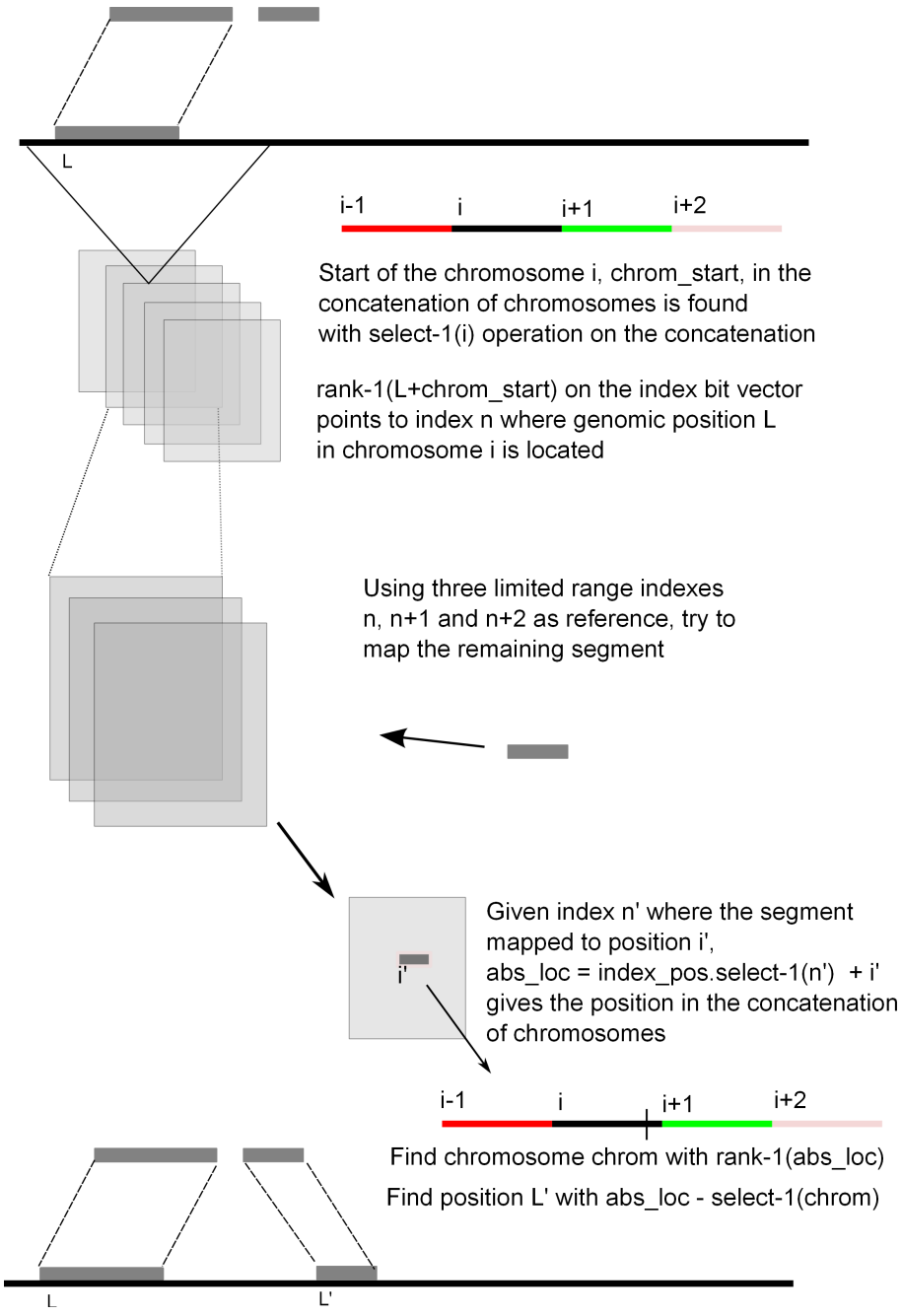


Figure 20: Navigating SpliceAligner's limited range indexes for finding the remaining segment.

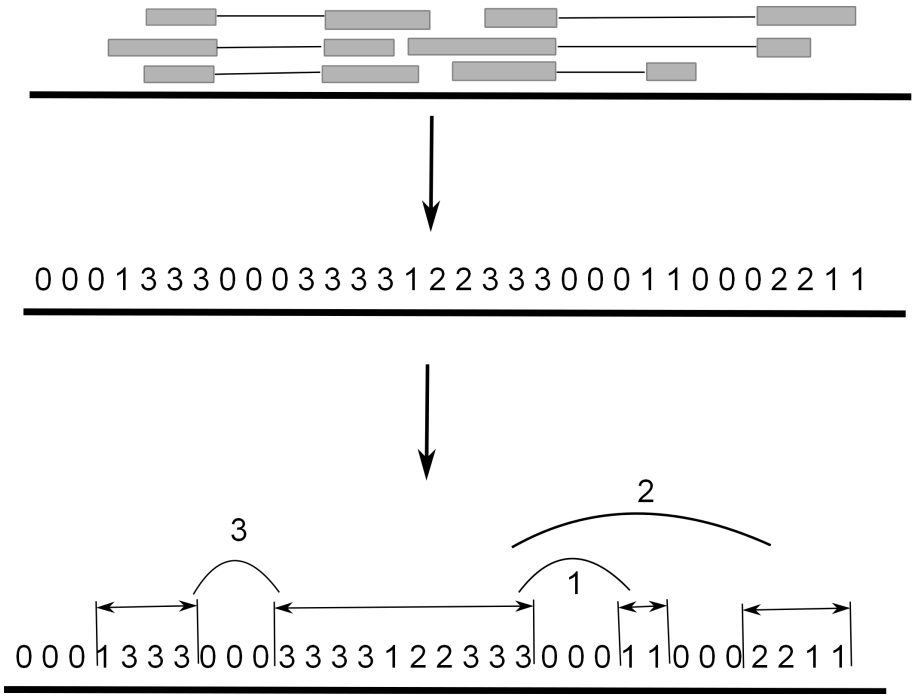


Figure 21: Inferring the junctions in SpliceAligner. First all reads are used to create chromosome wide coverage table. Each spliced read is then aligned to the coverage table to find the junction candidates. A junction candidate must have a sufficient number of spanning reads as well as a sufficient number of bases covered on each side to be accepted.

5 Results

For analyzing the performance of the chosen short read mapping software, I used sixteen simulated data sets and one data set from a real RNA-seq experiment (Caltech RNA-Seq track from the ENCODE project, NCBI SRA accession number SRR065504).

Simulated data sets were created from all the transcripts in chromosome 2 that were at least one kilobase long. Following the specs of Illumina HiSeq and MiSeq sequencers, single-end and paired-end data sets were created for read lengths of 50, 100, 150 and 250 bp. For every read length and mode, one data set was created with no sequencing errors and one data set was created using 1% error rate in calling each base. Single-end data sets consisted of one million reads and paired-end data sets of 2x one million reads.

For the simulated data, I measured the running times of all the tools as well as calculated the number of uniquely mapped reads, the number of multimapped reads (i.e. reads that aligned to multiple locations) and the number of unmapped reads (i.e. reads that failed to align anywhere). Then I scored all the alignments given by each tool based on the predicted chromosome, strand, start and end locations, and predicted splice sites. First I converted all the SAM/BAM alignment files the tools outputted to BED format (see Appendix 1 for detailed descriptions of the file formats). For uniquely mapping reads I defined *perfect match* as the alignment BED line matching exactly with the ground truth, and *fuzzy match* as matching the chromosome, strand, and start and end locations of the alignment to the ground truth. Block number, block sizes and block starts did not need to match to qualify as a fuzzy match, that is, the splice site location could be approximate. By definition, every perfect match also counts as a fuzzy match. For reads that mapped to multiple locations, I defined *perfect match for multimap* as one of the alignments matching the ground truth perfectly, and *fuzzy match for multimap* as one or more of the alignments matching chromosome, strand, and start and end locations of ground truth.

Alignment files from all tools were given to Traph [TKRM13b], a transcript prediction tool created by our group[7].

Following the example of Tomescu *et al.* [TKRM13b, TKRM13a], to compare the predicted transcripts to the annotated transcripts I extracted the corresponding

[7]Genome-scale Algorithmics, Department of Computer Science, University of Helsinki

sequences and created a bipartite graph with annotated transcripts on one side and the predicted transcripts on the other side. It should be noted that I extracted all the sequences for the comparison from the forward strand for both annotated and predicted transcripts, regardless of whether the annotated transcript was in forward or reverse strand, because not all the tools use the SAM format splice junction tags (XS:A tags) required to infer the strand in the transcript prediction.

As done in [TKRM13a], the edge weights in the bipartite graph are a combined measure of *sequence dissimilarity* and *relative expression difference*. Sequence dissimilarity between a true transcript T_i and a predicted transcript P_j is the edit distance between T_i and P_j divided by $max(|T_i|, |P_j|)$, and relative expression difference between the expression level of true transcript $e(T_i)$ and the expression level of predicted transcript $e(P_j)$ is $|e(T_i) - e(P_j)|/e(T_i)$. Computing minimal weight perfect matching for this graph finds the best pairs of annotated and predicted transcripts. Pairs with sequence dissimilarity and relative expression difference under given threshold are considered to be true positives. The remaining sequences are considered false positives or false negatives, depending on which side of the graph they were. I then calculated *f-score* for the predicted transcripts to see if the differences in the alignments between the tools affect transcript prediction accuracy.

F-score, or f-measure, is the harmonic mean of *precision* and *recall*, and has been used as the measure for the goodness of the model [LDH+12, LFJ11, TKRM13b].

$$\text{f-score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

Precision is the number of correctly predicted transcripts out of all the predicted transcripts ($\frac{TP}{TP+FP}$), and recall (also known as *sensitivity*) is the number of correctly predicted transcripts out of all the annotated transcripts ($\frac{TP}{TP+FN}$).

The real RNA-seq data set consisted of 50 million 75 bp paired-end reads from human embryonic stem cells. Without the ground truth, alignment positions cannot be verified, therefore for the real data I calculated the number of reads the tools gave the same alignment for, as well as the total number of reads each tool aligned. I also measured the running time and RAM required by each tool.

Unless noted otherwise, all the tools were ran with default parameters. Tophat allows a total of two mismatches or indels for the whole read, whereas SpliceAligner allows two mismatches but no indels. SOAPsplice accepts three mismatches or two indels for a segment, MapSplice allows for one mismatch in the first or last segment of the read and two in each of the middle segments, and SpliceMap one mismatch

per segment or two mismatches in a read without splice junction. SHRiMP2 does not set a threshold for the number of mismatches or indels but instead uses a scoring scheme featuring matches, mismatches and gaps.

SHRiMP2 index was split to three parts beforehand to allow it to fit to RAM, as I did not have a machine with 48 GB of RAM available, and the reads were mapped against the part indexes sequentially.

5.1 Simulated data

As the running time differences for the simulated data between no error and 1% error rate cases were under 10%, I calculated the average running time in CPU hours for each tool, read length and mode (single- or paired-end). The running times are shown in Table 1. SOAPsplice failed to run on all the 250 bp data sets, and Tophat on 250 bp paired-end data was terminated after running for longer than 100 CPU hours.

With 50 bp long reads, SOAPsplice was the fastest of the tools, but MapSplice, SpliceMap and Tophat were not far behind. As the read length increased, running times of SOAPsplice and Tophat increased faster than MapSplice's and SpliceMap's. Tophat and SOAPsplice especially had problems with 250 bp long reads. SOAPsplice was unable to run on both single-end and paired-end 250 bp data sets, and Tophat took approximately 43 CPU hours to process the single-end reads and over 100 CPU hours before the process was terminated for the paired-end reads. As the test data sets were only one million and two million reads for single-end and paired-end respectively, and Illumina MiSeq system produces 24-50 million paired-end reads that pass the quality filter (depending on the reagent kit), Tophat's running time for a real RNA-seq data set of 250 bp paired-end reads would be over 1200 CPU hours, or approximately 25 days running on dual-core system, at minimum.

Of the suffix-tree-based tools, my SpliceAligner was the slowest in the 50, 100 and 150 bp cases, as well as nearly ten times slower than SpliceMap and MapSplice on the 250 bp cases. This grouping leads me to suspect that the differences in the running times are at least partially caused by the underlying core aligners having different running speeds. Because readaligner [Mäk10] faithfully solves the k-mismatches problem while Bowtie [LTPS09] uses various speed-up heuristics, readaligner is much slower when using the SpliceAligner default parameter of two mismatches.

As expected of a hash table -based approach, SHRiMP2 was by far the slowest of

Table 1: Running times on simulated data sets of one million single-end and two million paired-end reads (average CPU hours between no error and 1% error rate cases)

	Tophat	SpliceMap	MapSplice	SOAPsplice	SHRiMP2	SpliceAligner
Single-end						
50 bp	0.29	0.31	0.18	0.12	1.73	0.70
100 bp	0.61	0.49	0.32	0.55	5.10	2.37
150 bp	1.05	0.69	0.39	1.42	11.04	3.95
250 bp	42.81	0.95	0.56	—	37.38	8.98
Paired-end						
50 bp	0.65	0.42	0.41	0.32	3.43	1.74
100 bp	1.50	0.78	0.64	1.23	9.95	3.07
150 bp	2.52	1.17	0.83	3.02	21.58	5.87
250 bp	> 100	1.82	1.14	—	74.75	15.47

all the tools in all categories. Having to split the index into three pieces to fit in the RAM, and therefore do the alignment process three times might add some overhead, but as the local alignment part of the query is significantly slower than matching the seeds to the hash table, this should not affect the running time significantly.

From the alignment files given by each tool, I calculated the number of uniquely mapped reads, multimapped reads and unmapped reads for all the data sets (shown in Table 2 and Table 3). It should be noted that while the results can be compared in the no sequencing error cases, they are not directly comparable in the 1% error rate cases because the tools accept varying number of mismatches per read.

In the no sequencing errors cases MapSplice and Tophat had a very low number of unmapped reads (under 2% in all the cases). The number was slightly higher in the paired-end mode, which is probably caused by simulating poly-A tails in the paired-end mode but not in the single-end mode, as a read containing mostly A's is unlikely to map anywhere.

MapSplice, Tophat and also SOAPsplice showed little variation for the number of unmapped reads as read length increased. On the contrary, the number of unmapped reads increased sharply as read length increased with SpliceMap, SHRiMP2 and SpliceAligner. With SpliceMap and SpliceAligner a possible reason could be the preference toward choosing a larger aligned segment: if a large segment aligned incorrectly, that is, the alignment could not be fully extended, a smaller segment that would be required to find the correct location will not be searched for. The description of the algorithms behind SHRiMP2 are too vague to hypothesize about the reason for missing a larger portion of alignments with read length increasing.

Table 2: Percentage of uniquely mapped reads (QM), multimapped reads (MM) and unmapped reads (UM) each tool reported for data set with no sequencing errors.

	Single-end 50 bp			Paired-end 50 bp		
	QM	MM	UM	QM	MM	UM
Tophat	95.26%	4.03%	0.71%	95.28%	3.74%	0.98%
SpliceMap	89.93%	2.32%	7.75%	43.89%	0.94%	55.17%
MapSplice	96.33%	3.67%	0.00%	97.69%	1.47%	0.85%
SOAPsplice	96.69%	0.19%	3.12%	92.35%	0.20%	7.45%
SHRiMP2	80.53%	15.68%	3.79%	80.28%	16.07%	3.65%
SpliceAligner	90.20%	5.14%	4.66%	83.82%	3.77%	12.41%
	Single-end 100 bp			Paired-end 100 bp		
	QM	MM	UM	QM	MM	UM
Tophat	97.06%	2.21%	0.73%	96.77%	1.93%	1.30%
SpliceMap	91.47%	2.78%	5.75%	44.94%	1.05%	54.01%
MapSplice	97.95%	2.05%	0.00%	99.24%	0.67%	0.09%
SOAPsplice	96.53%	0.52%	2.95%	94.46%	0.55%	4.99%
SHRiMP2	79.28%	13.20%	7.52%	78.87%	13.76%	7.37%
SpliceAligner	87.41%	5.31%	7.28%	83.82%	3.77%	12.41%
	Single-end 150 bp			Paired-end 150 bp		
	QM	MM	UM	QM	MM	UM
Tophat	97.23%	1.73%	1.05%	96.88%	1.51%	1.61%
SpliceMap	83.67%	1.67%	14.66%	41.22%	0.66%	58.11%
MapSplice	98.43%	1.58%	0.00%	99.56%	0.39%	0.05%
SOAPsplice	96.41%	0.57%	3.02%	94.73%	0.59%	4.68%
SHRiMP2	74.97%	11.57%	13.46%	74.37%	12.31%	13.32%
SpliceAligner	83.98%	6.58%	9.45%	80.51%	5.24%	14.25%
	Single-end 250 bp			Paired-end 250 bp		
	QM	MM	UM	QM	MM	UM
Tophat	96.81%	1.36%	1.82%	—	—	—
SpliceMap	37.44%	0.35%	62.21%	17.84%	0.04%	82.12%
MapSplice	98.83%	1.17%	0.00%	99.69%	0.25%	0.06%
SOAPsplice	—	—	—	—	—	—
SHRiMP2	62.33%	9.30%	28.37%	60.55%	11.24%	28.21%
SpliceAligner	77.69%	8.99%	13.31%	74.95%	7.32%	17.74%

In addition, the number of unmapped reads for SpliceMap was many times higher for the paired-end data than single-end data in 50, 100 and 150 bp cases. It is likely that the simulated pair conditions somehow conflicted with SpliceMap's expectations. The number of unmapped reads between single-end and paired-end data sets also show a larger difference with SpliceAligner than the other four tools, but not to the degree displayed by SpliceMap's alignments.

Except for SHRiMP2, all of the tools mapped the vast majority of the reads uniquely. SHRiMP2 did slightly worse with this metric, but still mapped the majority of the reads uniquely. However, it is interesting to note that the number of uniquely mapped reads compared to the number of multimapped reads stayed approximately the same for SpliceMap and SHRiMP2 across the different read lengths, increased for

Table 3: Percentage of uniquely mapped reads (QM), multimapped reads (MM) and unmapped reads (UM) each tool reported for data set with 1% base call error rate. + means the tool was ran with higher error threshold.

	Single-end 50 bp			Paired-end 50 bp		
	QM	MM	UM	QM	MM	UM
Tophat	93.68%	3.93%	2.40%	95.28%	3.74%	0.98%
SpliceMap	85.80%	2.20%	12.00%	41.68%	0.97%	57.35%
MapSplice	96.38%	3.62%	0.00%	96.60%	1.55%	1.85%
SOAPsplice	95.45%	0.17%	4.39%	91.19%	0.20%	8.61%
SHRiMP2	80.56%	15.20%	4.24%	80.35%	15.57%	4.08%
SpliceAligner	88.55%	5.17%	6.28%	84.54%	2.77%	12.69%
	Single-end 100 bp			Paired-end 100 bp		
	QM	MM	UM	QM	MM	UM
Tophat	88.86%	2.07%	9.08%	88.55%	1.79%	9.66%
SpliceMap	83.25%	2.43%	14.31%	40.77%	0.90%	58.33%
MapSplice	98.15%	1.85%	0.00%	99.20%	0.66%	0.15%
SOAPsplice	94.49%	0.38%	5.13%	92.82%	0.46%	6.72%
SHRiMP2	78.75%	12.79%	8.46%	78.48%	13.27%	8.25%
SpliceAligner	79.97%	4.93%	15.10%	68.45%	7.16%	24.39%
	Single-end 150 bp			Paired-end 150 bp		
	QM	MM	UM	QM	MM	UM
Tophat	78.29%	1.41%	20.31%	77.84%	1.28%	20.88%
Tophat+	94.52%	1.67%	3.82%	94.17%	1.50%	4.33%
SpliceMap	65.98%	1.20%	32.82%	32.46%	0.48%	67.06%
SpliceMap+	78.73%	1.60%	19.66%	38.77%	0.64%	60.60%
MapSplice	98.69%	1.31%	0.00%	99.47%	0.40%	0.13%
SOAPsplice	91.53%	0.33%	8.14%	90.52%	0.40%	9.08%
SOAPsplice+	95.06%	0.33%	4.61%	93.92%	0.40%	5.68%
SHRiMP2	74.00%	11.18%	14.82%	73.62%	11.75%	14.63%
SpliceAligner	67.57%	5.18%	27.25%	49.79%	11.36%	38.86%
SpliceAligner+	75.35%	5.47%	19.18%	62.61%	8.94%	28.45%
	Single-end 250 bp			Paired-end 250 bp		
	QM	MM	UM	QM	MM	UM
Tophat	52.56%	0.74%	46.71%	—	—	—
Tophat+	88.79%	1.21%	10.00%	—	—	—
SpliceMap	19.42%	0.14%	80.44%	9.16%	0.01%	90.83%
SpliceMap+	32.86%	0.31%	66.83%	15.62%	0.03%	84.36%
MapSplice	98.90%	1.10%	0.00%	99.62%	0.28%	0.10%
SOAPsplice	—	—	—	—	—	—
SOAPsplice+	—	—	—	—	—	—
SHRiMP2	61.21%	8.96%	29.82%	60.22%	10.21%	29.56%
SpliceAligner	41.93%	4.64%	53.42%	25.38%	11.56%	63.07%
SpliceAligner+	54.56%	5.50%	39.94%	36.79%	12.40%	50.81%

Tophat, MapSplice and SOAPsplice, but decreased for SpliceAligner. That is, contrary to the assumption that longer reads are easier to map uniquely, SpliceAligner mapped more reads to multiple locations as the read length increased. However, it is likely that these are not distinct genomic locations, but instead varying predictions for the exact splice sites within the read.

Examining the results of the 1% error rate cases, it is clear that the default parameter of two errors in the whole read is insufficient for longer reads. While the number of unmapped reads for 50 bp case (0.5 errors per read on average) are on the same level as for the data with no sequencing errors and at a tolerable level of 15% or less when the reads are 100 bp long (one error per read on average), for the 250 bp case from 45% to 80% of the reads do not map for Tophat, SpliceMap and SpliceAligner. This is unacceptable for most applications. However, Tophat performed well in the sense that it mapped nearly as many reads as theoretically possible given the read length and the default error parameters, whereas SpliceMap and SpliceAligner did not. SOAPsplice and SHRiMP2 that allow more than two errors per read perform better. MapSplice that only limits errors per segment (with each segment having static length) is not affected by the increase in read length.

To test the limits of the software, I attempted to run Tophat, SpliceMap, SOAPsplice and SpliceAligner for the 150 bp and 250 bp data sets with a higher number of errors allowed. Documentation of SHRiMP2 is unclear on how to allow a higher number of errors, so I could not attempt to run it with higher error tolerance.

SOAPsplice has hard limit of five mismatches or 3 bp gap for the error model, so I used those parameters. SpliceMap can find at most two mismatches per 25 bp segment, so I set the limit for segment mismatches to two and no limit for the total number of mismatches in the read. For Tophat I set the maximum number of allowed errors to five. This did not increase the running time significantly, so it is likely that the error tolerance parameters could be set higher. As with the default parameters, Tophat could not finish running on the 250 bp paired-end data set within 100 CPU hours. SpliceAligner could only run with three mismatches at most without going over the 100 CPU hours limit.

As expected, the number of mapped reads increased for all four tools when more errors were allowed. Tophat and SOAPsplice reached roughly the same level of mapped reads as with the 50 bp and 100 bp cases. SpliceMap did not map as many reads as could have been expected from the amount of errors allowed. It is possible that with high error tolerance SpliceMap mapped some of the seeds into too many locations and therefore discarded those reads. SpliceAligner's performance also rose slightly in this category, but three mismatches allowed seems to still be too strict for longer reads.

As the plain number of mapped reads does not necessarily equate to good performance, the next step was to investigate how correct the predicted alignments

Table 4: Correctness of predicted alignments for each tool for the data sets with no sequencing errors. PMU = perfect match for uniquely mapped read, FMU = fuzzy match for uniquely mapped read, PMM = perfect match for multimapped read, FMM = fuzzy match for multimapped read. All the values are out of the number of aligned reads. TC = total correct, percentage of total number of reads that was mapped perfectly.

	Single-end 50 bp					Paired-end 50 bp				
	PMU	FMU	PMM	FMM	TC	PMU	FMU	PMM	FMM	TC
Tophat	99.03%	99.05%	92.56%	92.59%	98.06%	98.49%	98.52%	84.10%	84.14%	96.98%
SpliceMap	96.73%	96.73%	95.99%	95.99%	89.22%	48.49%	48.49%	48.28%	48.28%	21.74%
MapSplice	97.11%	97.12%	93.22%	93.23%	96.96%	96.95%	96.97%	90.07%	90.07%	96.03%
SOAPsplice	90.30%	90.33%	88.08%	90.16%	87.48%	92.07%	92.09%	87.24%	89.40%	85.20%
SHRiMP2	83.91%	83.93%	86.93%	86.94%	81.21%	83.00%	83.02%	85.50%	85.52%	80.38%
SpliceAligner	95.05%	96.09%	90.40%	91.57%	90.38%	88.93%	93.14%	90.81%	94.45%	77.97%
	Single-end 100 bp					Paired-end 100 bp				
	PMU	FMU	PMM	FMM	TC	PMU	FMU	PMM	FMM	TC
Tophat	99.30%	99.36%	96.03%	96.10%	98.51%	98.73%	98.79%	86.12%	86.24%	97.20%
SpliceMap	85.76%	85.77%	83.90%	83.91%	80.78%	42.81%	42.82%	41.87%	41.87%	19.68%
MapSplice	97.36%	97.41%	92.94%	92.97%	97.27%	96.13%	96.18%	93.23%	93.23%	96.02%
SOAPsplice	91.10%	91.22%	90.15%	91.54%	88.41%	91.47%	91.60%	90.11%	91.50%	86.90%
SHRiMP2	69.33%	69.36%	75.61%	75.63%	64.94%	67.77%	67.80%	74.43%	74.48%	63.70%
SpliceAligner	93.66%	96.01%	92.24%	94.80%	86.77%	88.93%	93.14%	90.81%	94.45%	77.97%
	Single-end 150 bp					Paired-end 150 bp				
	PMU	FMU	PMM	FMM	TC	PMU	FMU	PMM	FMM	TC
Tophat	99.43%	99.50%	96.48%	96.67%	98.34%	98.85%	98.93%	88.42%	88.65%	97.10%
SpliceMap	86.21%	86.22%	84.85%	84.86%	73.55%	43.06%	43.07%	41.95%	41.96%	18.03%
MapSplice	97.02%	97.09%	91.14%	91.22%	96.92%	95.75%	95.83%	94.32%	94.32%	95.70%
SOAPsplice	32.32%	34.23%	91.47%	92.44%	31.68%	33.32%	35.24%	90.36%	91.37%	32.10%
SHRiMP2	61.86%	61.92%	69.90%	69.92%	54.47%	60.18%	60.23%	69.51%	69.57%	53.31%
SpliceAligner	92.60%	96.06%	89.24%	93.53%	83.63$	87.60%	93.70%	88.93%	94.00%	75.19%
	Single-end 250 bp					Paired-end 250 bp				
	PMU	FMU	PMM	FMM	TC	PMU	FMU	PMM	FMM	TC
Tophat	99.55%	99.65%	97.15%	97.29%	97.70%	—	—	—	—	–
SpliceMap	85.91%	85.92%	53.15%	53.15%	32.35%	86.34%	86.35%	75.60%	75.60%	15.43%
MapSplice	95.53%	95.64%	84.32%	84.74%	95.39%	94.11%	94.23%	93.49%	93.51%	94.05%
SOAPsplice	—	—	—	—	—	—	—	—	—	—
SHRiMP2	63.24%	63.98%	68.79%	69.31%	45.82%	61.33%	62.00%	68.47%	69.92%	44.83%
SpliceAligner	90.89%	96.23%	85.26%	93.25%	78.28%	85.52%	94.47%	86.82%	94.69%	70.45%

were (see Table 4 and Table 5). I considered uniquely mapped and multimapped reads separately for this comparison. For an uniquely mapped read, a perfect match matches the ground truth perfectly, whereas a fuzzy match matches the ground truth approximately, that is, the splice sites within the read do not need to be exact. For multimapped read to be considered a perfect match, one of the alignments needs to match the ground truth perfectly, and to be considered a fuzzy match, one or more of the alignments need to fulfill the criteria for fuzzy match.

Table 5: Correctness of predicted alignments for each tool for the data sets with 1% base call error rate ran with default parameters. PMU = perfect match for uniquely mapped read, FMU = fuzzy match for uniquely mapped read, PMM = perfect match for multimapped read, FMM = fuzzy match for multimapped read. All the values are out of the number of aligned reads. TC = total correct, percentage of total number of reads that was mapped perfectly. + means the tool was ran with higher error threshold.

	Single-end 50 bp					Paired-end 50 bp				
	PMU	FMU	PMM	FMM	TC	PMU	FMU	PMM	FMM	TC
Tophat	99.17%	99.20%	92.52%	92.54%	96.54%	98.70%	98.73%	84.58%	84.62%	95.55%
SpliceMap	97.06%	97.06%	96.07%	96.07%	85.39%	48.68%	48.68%	48.37%	48.37%	20.76%
MapSplice	95.98%	96.00%	91.61%	91.61%	95.82%	96.95%	96.96%	88.06%	88.06%	95.01%
SOAPsplice	90.46%	90.48%	88.97%	91.31%	86.49%	92.18%	92.24%	75.18%	84.56%	84.21%
SHRiMP2	84.36%	84.38%	86.92%	86.93%	81.17%	83.42%	83.43%	85.53%	85.55%	80.35%
SpliceAligner	95.45%	96.49%	90.62%	91.72%	89.21%	90.87%	92.79%	92.87%	94.54%	79.40%
	Single-end 100 bp					Paired-end 100 bp				
	PMU	FMU	PMM	FMM	TC	PMU	FMU	PMM	FMM	TC
Tophat	99.47%	99.51%	96.45%	96.53%	90.38%	99.00%	99.06%	87.66%	87.77%	89.23%
SpliceMap	82.49%	82.50%	79.94%	79.94%	70.62%	41.18%	41.18%	39.74%	39.74%	17.15%
MapSplice	96.36%	96.40%	90.00%	90.01%	96.24%	95.47%	95.51%	93.58%	93.58%	95.31%
SOAPsplice	89.99%	90.09%	91.52%	93.51%	85.38%	90.02%	90.20%	79.84%	90.73%	83.92%
SHRiMP2	70.09%	70.13%	76.03%	76.05%	64.93%	68.51%	68.54%	74.69%	74.72%	63.68%
SpliceAligner	94.37%	96.88%	93.46%	95.78%	80.08%	87.82%	92.25%	92.74%	96.11%	66.75%
	Single-end 150 bp					Paired-end 150 bp				
	PMU	FMU	PMM	FMM	TC	PMU	FMU	PMM	FMM	TC
Tophat	99.55%	99.61%	98.01%	98.12%	79.31%	99.16%	99.23%	87.65%	87.89%	78.31%
Tophat+	98.50%	98.58%	92.55%	92.79%	94.64%	97.73%	97.82%	82.94%	83.17%	93.27%
SpliceMap	83.27%	83.28%	81.03%	81.03%	55.92%	41.58%	41.58%	39.70%	39.71%	13.69%
SpliceMap+	83.56%	83.58%	80.79%	80.79%	67.09%	41.83%	41.84%	39.47%	39.48%	16.47%
MapSplice	95.48%	95.55%	85.07%	85.07%	95.34%	94.86%	94.92%	93.03%	93.03%	94.72%
SOAPsplice	69.60%	73.08%	93.07%	94.51%	64.01%	69.36%	73.01%	78.56%	92.89%	63.09%
SOAPsplice+	70.10%	74.66%	93.01%	94.47%	66.95%	69.88%	74.64%	77.66%	92.81%	65.95%
SHRiMP2	62.92%	62.98%	70.67%	70.69%	54.46%	61.25%	61.30%	69.78%	69.83%	53.29%
SpliceAligner	93.38%	97.20%	91.03%	95.28%	67.81%	86.16%	92.62%	91.88%	96.54%	53.33%
SpliceAligner+	93.68%	96.73%	91.75%	94.91%	75.61%	88.46%	94.05%	91.63%	96.09%	63.57%
	Single-end 250 bp					Paired-end 250 bp				
	PMU	FMU	PMM	FMM	TC	PMU	FMU	PMM	FMM	TC
Tophat	99.60%	99.68%	96.29%	97.22%	53.05%	—	—	—	—	—
Tophat+	99.11%	99.20%	92.12%	92.29%	89.11%	—	—	—	—	—
SpliceMap	83.64%	83.65%	47.35%	47.35%	16.31%	84.33%	84.33%	72.55%	72.55%	7.73%
SpliceMap+	84.67%	84.68%	49.61%	49.61%	27.98%	85.15%	85.16%	71.93%	71.93%	13.32%
MapSplice	69.75%	69.79%	51.88%	51.88%	69.55%	69.77%	69.83%	85.37%	85.37%	69.74%
SOAPsplice	—	—	—	—	—	—	—	—	—	—
SOAPsplice+	—	—	—	—	—	—	—	—	—	—
SHRiMP2	64.64%	65.38%	69.64%	70.16%	45.81%	62.96%	63.65%	67.55%	68.98%	44.82%
SpliceAligner	91.83%	97.63%	86.89%	95.70%	42.54%	85.41%	94.34%	90.32%	97.26%	32.11%
SpliceAligner+	93.17%	97.45%	88.30%	95.12%	55.69%	87.8%	94.77%	91.37%	97.21%	43.37%

Tophat had by far the most correct predictions, with over 98.5% of the predicted unique mappings being perfect matches and vast majority of the multimappings also qualifying as a perfect match with all the data sets. There were only a fraction of a percent more fuzzy matches than perfect matches, therefore the splice sites that Tophat predicted were very accurate. Tophat also had either the highest or second highest total number of correctly mapped reads (TC in the tables) with all the data sets, as long as the given error parameters took the read length and error rate into account.

None of the other tools had as consistent good performance as Tophat in this category. MapSplice scored over 94% perfect matches for unique mappings and over 84% perfect matches for multimappings with all the data sets except 250 bp with 1% error rate, where the values sunk to 69% and 51% for single-end and 69% and 85% for paired-end, respectively. The results for MapSplice did not have significant difference between perfect match and fuzzy match values either, pointing to splice sites being predicted accurately as well. MapSplice shared the first place with Tophat in the total number of correctly mapped reads.

For overall correctness, SpliceAligner ranked the third: over 90% of the unique mappings and over 85% of the multimappings qualified as perfect matches. In addition, over 96% of the unique mappings and over 92% of the multimappings qualified as fuzzy matches. These values point to SpliceAligner actually being more accurate in its predictions than MapSplice when the exact splice site is not as important. Considering the fuzzy matches, SpliceAligner's performance was also consistent over all the read lengths. As SpliceAligner could not be ran in a reasonable time with error parameters higher than three mismatches for the whole read, with longer reads it could not map as many reads, and therefore fell behind on the total number of correctly mapped reads.

SpliceMap started with excellent performance of approximately 96% of both uniquely mapped and multimapped reads being perfect matches in the 50 bp single-end case, but the performance dropped around 10% when read length increased to 100 bp. As mentioned above, the simulated reads most likely somehow conflict with the paired-end assumptions of SpliceMap, as under half the alignments SpliceMap predicted in the paired-end data sets qualified as even fuzzy matches.

SOAPsplice and SHRiMP2 also started with a decent to good performance with short reads, but the percentage of the alignment predictions being correct sunk with the read length increasing. SOAPsplice completely failed to run on the 250 bp data

sets. Combined with SpliceMap and SHRiMP2 mapping less of the longer reads, the total number of correctly mapped reads also decreased sharply as read length increased.

As a final test, I gave the alignment files for 50 bp (with default parameters) and 150 bp (with higher error thresholds) single-end 1% error rate data sets from each mapping software to transcript prediction tool Traph and calculated f-scores for the predicted transcripts at various sequence dissimilarity and relative expression level difference thresholds (see Figure 22 and Figure 23 for 3D plots, and Appendix 2 for 2D plots).

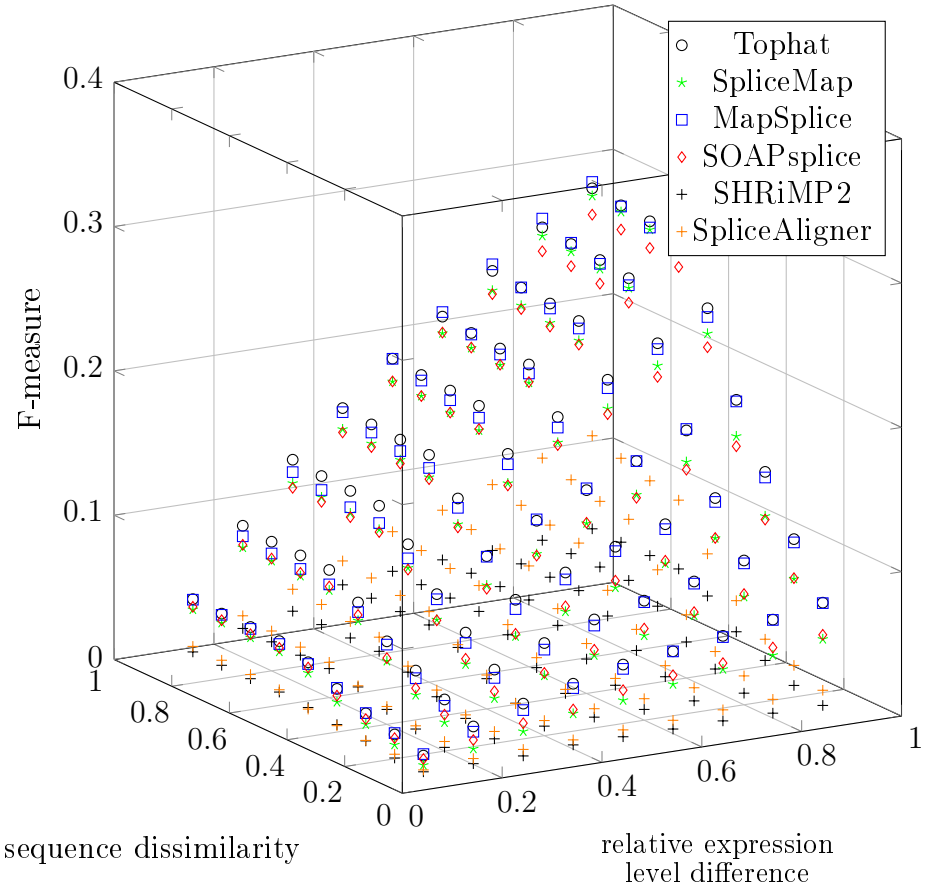


Figure 22: F-measure at different sequence dissimilarity and relative expression level difference thresholds for transcripts predicted from aligned 50 bp single-end reads data set.

As could be hypothesized from the alignment accuracy, transcripts predicted from the alignments of Tophat and MapSplice had the highest f-measure in both of the cases. With 50 bp long reads the scores were near even at all thresholds, whereas

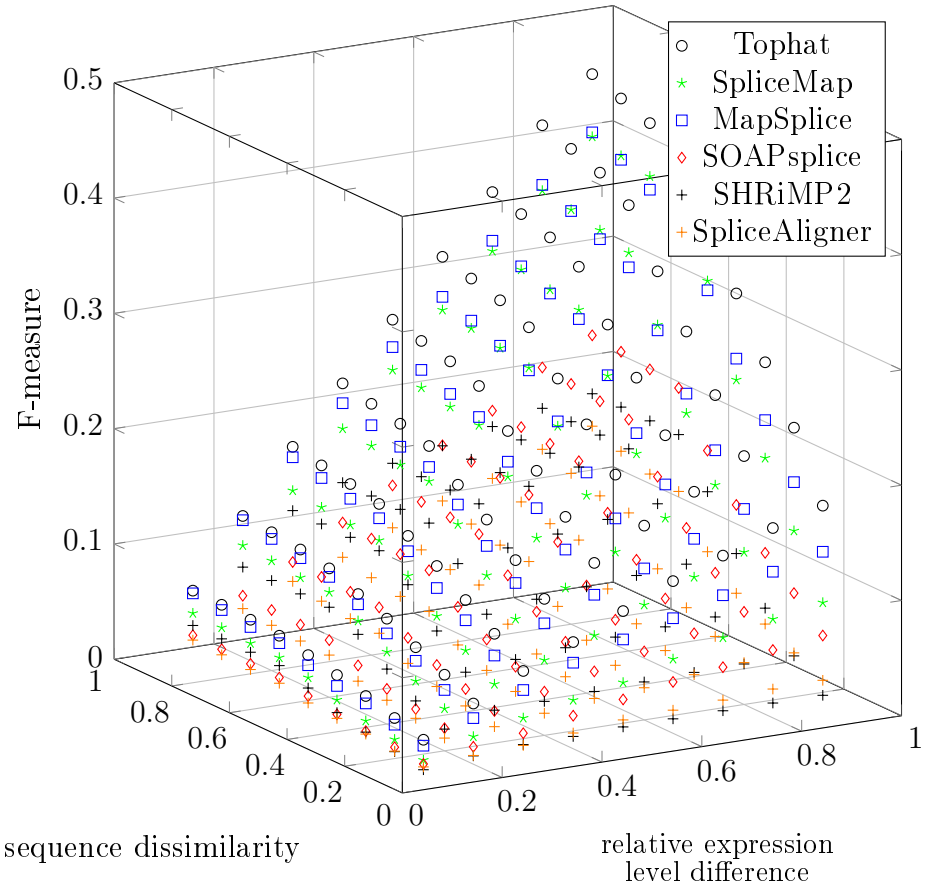


Figure 23: F-measure at different sequence dissimilarity and relative expression level difference thresholds for transcripts predicted from aligned 150 bp single-end reads data set.

with 150 bp long reads Tophat was better at high relative expression level differences. SOAPsplice and SpliceMap were slightly behind in the case of 50 bp long reads, with the gap growing for the 150 bp case. However, SOAPsplice on the case of 50 bp long reads and SpliceMap on both cases approached Tophat's and MapSplice's level at higher sequence dissimilarity thresholds.

SHRiMP2 and SpliceAligner performed badly in this category. However, it should be noted that because SHRiMP2 uses the mapping quality scores differently than Traph had been designed for, I had to disable some of the multimapping handling heuristics that were dependent on the mapping quality score to produce any transcripts for these tests. Therefore the higher number of multimappings from SpliceAligner and SHRiMP2 could have skewed the expression level estimates.

5.2 Real data

The running time and max memory usage of each tool, as well as the number of reads mapped out of the 50 million reads, for the real RNA-seq data are shown in Figure 6. Surprisingly the order of the tools from fastest to slowest on real data differs from the order on 50 bp and 100 bp long simulated data sets. Whereas on simulated 50 bp long paired-end data SOAPsplice was the fastest and on simulated 100 bp long paired-end data MapSplice was the fastest, on real data SpliceMap and Tophat were much faster than MapSplice and SOAPsplice. SHRiMP2 and SpliceAligner stayed behind, and their relative differences in the running times stayed approximately same as with the simulated data. The differences between the real and simulated data might have been caused by how the tools handle multiple chromosomes at the same time, since the simulated data is created from a single chromosome.

Table 6: Running time (in CPU hours), RAM usage (in gigabytes) and the percentage of reads mapped for real RNA-seq experiment data consisting of 50 million paired-end reads.

	Runtime	Max RAM	% reads mapped
Tophat	21.9	3.4	74.9%
SpliceMap	15.1	3.2	59.8%
MapSplice	47.3	5.1	85.9%
SOAPsplice	58.6	5.4	69.8%
SHRiMP2	337.9	13.0	87.8%
SpliceAligner	108.2	9.3	35.0%

As the ground truth is not known for a real RNA-seq experiment, the alignment positions cannot be verified. However, some information can be gained by examining the similarities and differences on the alignments between the tools. For all the tools I calculated the number of alignments they agree on. Pairwise set intersections are shown in Table 7, and some of the four-set Venn diagrams in Figure 24, Figure 25 and Figure 26.

As the criteria for the tools to agree on alignment(s) I defined that the set of alignments, whether it be a single alignment or multiple alignments, for a given read must be exactly the same. This naturally causes some artificial differences, as some of the tools use various heuristics to choose one from equally good alignment candidates, whereas some report all the good candidates. With these criteria, there were a total of 119,920,876 different alignment sets for 50 million reads.

Tophat, MapSplice and SOAPsplice shared vast majority of their alignments, approximately 27.4 million reads out of 37.4 million (Tophat), 43.0 million (Map-

Table 7: The number of alignments shared between pairs of tools. Alignments between two tools are considered shared either if for a given read they both output the same unique alignment or exactly the same set of multiple alignments (sharing some alignments within the set but not all is not sufficient). The numbers on the diagonal from top left to bottom right show how many reads total each tool mapped.

	Tophat	SpliceMap	MapSplice	SOAPsplice	SHRiMP2	SpliceAligner
Tophat	37,447,821	11,435,070	33,831,219	27,880,033	14,388,748	13,387,054
SpliceMap	11,435,070	29,911,890	11,760,019	11,028,572	6,594,841	10,305,087
MapSplice	33,831,219	11,760,019	42,965,528	29,553,257	15,658,513	13,224,874
SOAPsplice	27,880,033	11,028,572	29,553,257	34,914,840	15,018,971	12,442,911
SHRiMP2	14,388,748	6,594,841	15,658,513	15,018,971	43,878,802	6,894,268
SpliceAligner	13,387,054	10,305,087	13,224,874	12,442,911	6,894,268	17,493,880

Splice) and 34.9 million (SOAPsplice) had the same alignment for all three of the tools. SpliceAligner only aligned approximately 17.5 million reads, but also vast majority of those alignments were shared with Tophat, MapSplice and SOAPsplice. SpliceMap and SHRiMP2 gave more different alignments, as only roughly one third to one half of alignments (counting from the smallest number of mapped reads among the tools being compared) were shared with any other tool.

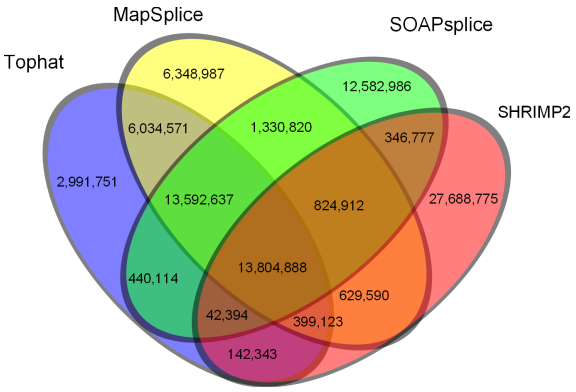


Figure 24: Shared alignments between the four tools that aligned the most reads.

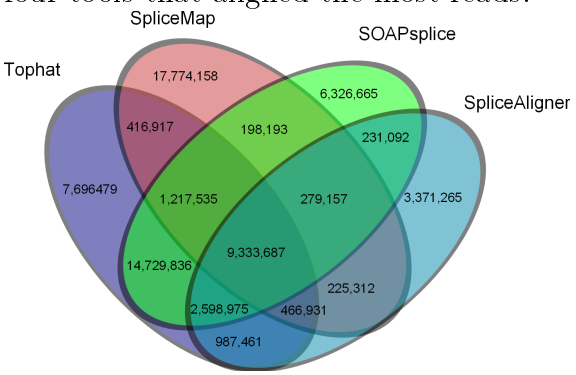


Figure 25: Shared alignments between the four tools that aligned the least reads.

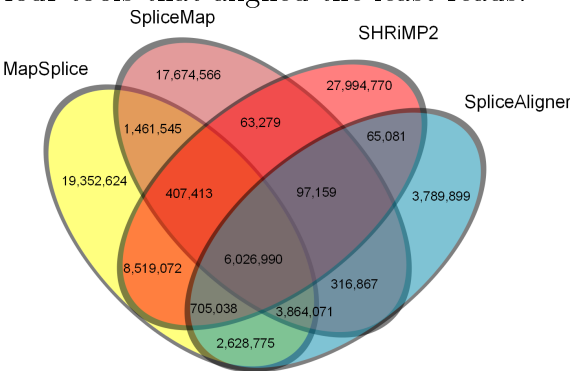


Figure 26: Shared alignments between two tools that aligned the most reads and two that aligned the least reads.

6 Conclusion

In this thesis I analyzed and compared the performance of six spliced read aligners, Tophat, SpliceMap, MapSplice, SOAPsplice, SHRiMP2 and SpliceAligner, each with their own approach to aligning split reads. Tophat creates exon "islands" from the reads that map fully to the reference genome, then uses these islands to map the split reads to the junctions. SpliceMap, SOAPsplice and SpliceAligner all split the reads into pieces, attempt to align as large a piece as possible fully and then search for the remaining segment(s). Implementation details for searching for the remaining segment(s) vary between the tools. MapSplice uses a tag-alignment approach, where each read is segmented to short segments, or "tags", and tags that mapped intact can be used to pinpoint the candidate positions for tags that are likely to contain a splice junction. Whereas these five software use a suffix-tree-based core aligner to map the segments, SHRiMP2 uses a hash table to locate k-mers from the reads in the reference, and then uses Smith–Waterman algorithm to attempt a local alignment.

For performance comparison I used 16 simulated data sets created from genes in human chromosome 2 and one real RNA-seq experiment data set. Following the specs of Illumina HiSeq and MiSeq sequencing machines, I created both single-end and paired-end data sets with read lengths of 50, 100, 150 and 250 bp. For each read length and mode, I created two data sets: one without any errors and one simulating 1% error rate for calling each base.

I evaluated the tools based on running time, number of reads mapped (uniquely and multimappings) and alignment accuracy. In addition I gave the alignments from each software to transcript prediction tool Traph, and measured correctness of each set of predicted transcripts by calculating f-score with varying sequence dissimilarity and relative expression differences allowed between the annotated and predicted transcripts.

Tophat, MapSplice and SOAPsplice were all very fast in most cases and mapped vast majority of the simulated reads uniquely in most cases as well. Tophat slowed down significantly when read length exceeded 150 bp, to the point of being near unusable for 250 bp long paired-end reads. SOAPsplice failed to process both 250 bp long reads data sets.

For the data with 1% error rate Tophat and SOAPsplice ran with default parameters naturally mapped less of the long reads, as their error tolerances were exceeded. However, they mapped almost as many reads as theoretically possible for the given

read length and error parameters. As MapSplice allows 1-2 errors per segment and the segments are a static length, it mapped 100% of the reads in all the cases with the default parameters. When the error parameters of Tophat and SOAPsplice were raised to a level more fit for the known error rate, they mapped nearly as many of the long reads as the short reads.

Only Tophat kept a near perfect record on the alignments it gave being correct over all the simulated data sets. SOAPsplice's performance dropped heavily when the read length exceeded 100 bp. MapSplice did well with reads up to 150 bp, but only 69.8% of its unique alignments were correct with 250 bp long single-end reads with 1% error rate, compared to Tophat's 99%.

SpliceMap was not significantly slower than the three top contenders, but it was lacking in both the number of reads mapped (even with higher error parameters, and especially in the paired-end cases that mapped less than half of the reads) and the correctness of the given alignments for reads longer than 50 bp. SpliceAligner suffered from the smaller number of mapped reads at longer read lengths as well, but the correctness of its alignments was high. Part of the problem with failing to align as many longer reads than shorter reads might be because if a large seed segment aligns incorrectly, a smaller seed segment will not be searched for. SpliceAligner seemed to also have a minor issue with finding the exact splice site, as it found the exact splice site less often Tophat and MapSplice in all the cases, but found the approximate correct alignment (where the splice site can be approximate) more often than MapSplice.

Being a hash-based approach, SHRiMP2 was naturally many times slower than any of the FM-index-based software. On the data with no sequencing errors SHRiMP2 generally mapped either the smallest or second smallest number of reads. On the long reads with 1% error rate SHRiMP2 mapped more than Tophat, SpliceMap and SpliceAligner when ran with default parameters, because it does not set limits to the errors per read. But when the other tools were ran with more fitting error parameters, SHRiMP2 fell behind Tophat on the number of mapped reads. SHRiMP2 also had the highest number of multimapped reads, as well as the lowest number of correct alignments for all single-end data sets and second lowest for all paired-end data sets.

For the sets of transcripts predicted from the alignments of 50 bp long reads, f-scores at all sequence dissimilarity and relative expression level difference thresholds were near identical for Tophat and MapSplice. SOAPsplice was slightly behind at

low relative expression difference levels, with the gap growing toward high relative expression difference and low sequence dissimilarity. With the sets of transcripts predicted from the alignments of 150 bp long reads, Tophat and MapSplice had similar f-scores at low relative expression difference levels, but Tophat was significantly better at high relative expression differences levels. SOAPsplice also did significantly worse on the transcripts predicted from 150 bp long reads.

Transcripts predicted from SpliceMap alignments were competitive on f-score with Tophat, MapSplice and SOAPsplice on 50 bp long reads, but like SOAPsplice, SpliceMap fell behind compared to Tophat and MapSplice on 150 bp long reads at low sequence dissimilarity threshold. However, on 150 bp long reads SpliceMap was better than SOAPsplice. Neither SpliceAligner nor SHRiMP2 did well in this category, most likely because of their higher number of multimapped reads, as I had to disable some of the multimapping heuristics of Traph to account for SHRiMP2 using SAM mapping quality score differently than the other tools.

Tophat holds well to its reputation of the most popular aligner on commonly used 50-100 bp long Illumina type reads because of its speed, high number of mapped reads and the high accuracy of the alignments, combined with the ease of use. Based on the tests on the simulated data Tophat is also the top choice for 150 bp long reads, as long as the error parameters are set accordingly.

MapSplice and SOAPsplice would be good candidates for the top choice as well, being at comparable level on speed, number of mapped reads and their accuracy. But their many required parameters, and in SOAPsplice's case lacking documentation, can be intimidating for a casual user. However, Tophat is not a good choice for reads longer than 150 bp, as based on the simulated data, it would take over three weeks to process a single 250 bp long reads data set produced by Illumina MiSeq on a dual core machine. SOAPsplice also has problems with longer reads. This leaves MapSplice, which is not affected by read length because its approach is based on segmenting the reads to pieces with static length, as the top choice for reads longer than 150 bp.

From the results of the experiments on both simulated and real data it is clear that while my software SpliceAligner can be reasonably competitive in the overall performance, it could use improvements on several areas. Two of the main concerns are the running time and memory requirement. SpliceAligner is several times slower than its FM-index -based competitors and requires over 3.5 GB more memory, for a total of 9.3 GB required.

I attempted to address the running time problem by plugging in Bowtie as SpliceAligner's core aligner, but creating the limited range index collection using Bowtie's index building tool (bowtie-build) turned out to require over 250 GB of disk space for the human genome. As for the memory requirement, in addition to the two indexes (approximately 3 GB each), SpliceAligner reads the sequence of the whole genome to memory when attempting to find exact splice sites. Therefore the memory requirement could be dropped to the same level as MapSplice and SOAPsplice using random access to the chromosome sequence files.

Some of the smaller improvements for future work include attempting to address the problem of a larger segment aligning incorrectly and fine-tuning the splice site detection.

7 Acknowledgments

I'd like to thank Antti Honkela and Veli Mäkinen for giving me the opportunity to work in a research group, without which this thesis would not have been possible, and for supervising the writing of the thesis. I'd also like to thank Niko Välimäki for coding help when developing SpliceAligner, and Alexandru Tomescu, Simon Puglisi and Esa Pitkänen for discussions and helpful comments during the writing process. And finally, thanks to Leena Salmela and Sirkka-Liisa Varvio for master's thesis seminar and all the support during the MBI studies.

References

AGM+90 Altschul, S. F., Gish, W., Miller, W., Myers, E. W. and Lipman, D. J., Basic local alignment search tool. *J. Mol. Biol.*, 215,3(1990), pages 403–410. URL `http://dx.doi.org/10.1016/S0022-2836(05)80360-2`.

AH10 Anders, S. and Huber, W., Differential expression analysis for sequence count data. *Genome Biol.*, 11,10(2010), page R106. URL `http://dx.doi.org/10.1186/gb-2010-11-10-r106`.

AJL+10 Au, K. F., Jiang, H., Lin, L., Xing, Y. and Wong, W. H., Detection of splice junctions from paired-end RNA-seq data by SpliceMap. *Nucleic Acids Res.*, 38,14(2010), pages 4570–4578. URL `http://dx.doi.org/10.1093/nar/gkq211`.

And10 Andrews, S., FASTQC. A quality control tool for high throughput sequence data. *URL http://www.bioinformatics.babraham.ac.uk/projects/fastqc*.

Ans09 Ansorge, W. J., Next-generation DNA sequencing techniques. *N Biotechnol*, 25,4(2009), pages 195–203. URL `http://dx.doi.org/10.1016/j.nbt.2008.12.009`.

BCZF12 Bonfert, T., Csaba, G., Zimmer, R. and Friedel, C. C., A context-based approach to identify the most likely mapping for RNA-seq experiments. *BMC Bioinformatics*, 13 Suppl 6, page S9. URL `http://dx.doi.org/10.1186/1471-2105-13-S6-S9`.

CADC10 Costa, V., Angelini, C., De Feis, I. and Ciccodicola, A., Uncovering the complexity of transcriptomes with RNA-Seq. *J Biomed Biotechnol*, 2010, page 853916. URL `http://dx.doi.org/10.1155/2010/853916`.

CCHD13 Casey, G., Conti, D., Haile, R. and Duggan, D., Next generation sequencing and a new era of medicine. *Gut*, 62,6(2013), pages 920–932. URL `http://dx.doi.org/10.1136/gutjnl-2011-301935`.

CFZ+13 Chung, L. M., Ferguson, J. P., Zheng, W., Qian, F., Bruno, V., Montgomery, R. R. and Zhao, H., Differential expression analysis for paired RNA-Seq data. *BMC Bioinformatics*, 14, page 110. URL `http://dx.doi.org/10.1186/1471-2105-14-110`.

DDL+11 David, M., Dzamba, M., Lister, D., Ilie, L. and Brudno, M., SHRiMP2: sensitive yet practical SHort Read Mapping. *Bioinformatics*, 27,7(2011), pages 1011–1012. URL `http://dx.doi.org/10.1093/bioinformatics/btr046`.

DDM+12 Djebali, S., Davis, C. A., Merkel, A., Dobin, A., Lassmann, T., Mortazavi, A., Tanzer, A., Lagarde, J., Lin, W., Schlesinger, F., Xue, C., Marinov, G. K., Khatun, J., Williams, B. A., Zaleski, C., Rozowsky, J., Röder, M., Kokocinski, F., Abdelhamid, R. F., Alioto, T., Antoshechkin, I., Baer, M. T., Bar, N. S., Batut, P., Bell, K., Bell, I., Chakrabortty, S., Chen, X., Chrast, J., Curado, J., Derrien, T., Drenkow, J., Dumais, E., Dumais, J., Duttagupta, R., Falconnet, E., Fastuca, M., Fejes-Toth, K., Ferreira, P., Foissac, S., Fullwood, M. J., Gao, H., Gonzalez, D., Gordon, A., Gunawardena, H., Howald, C., Jha,

S., Johnson, R., Kapranov, P., King, B., Kingswood, C., Luo, O. J., Park, E., Persaud, K., Preall, J. B., Ribeca, P., Risk, B., Robyr, D., Sammeth, M., Schaffer, L., See, L.-H., Shahab, A., Skancke, J., Suzuki, A. M., Takahashi, H., Tilgner, H., Trout, D., Walters, N., Wang, H., Wrobel, J., Yu, Y., Ruan, X., Hayashizaki, Y., Harrow, J., Gerstein, M., Hubbard, T., Reymond, A., Antonarakis, S. E., Hannon, G., Giddings, M. C., Ruan, Y., Wold, B., Carninci, P., Guigó, R. and Gingeras, T. R., Landscape of transcription in human cells. *Nature*, 489,7414(2012), pages 101–108. URL `http://dx.doi.org/10.1038/nature11233`.

EG00 Ewing, B. and Green, P., Analysis of expressed sequence tags indicates 35,000 human genes. *Nat. Genet.*, 25,2(2000), pages 232–234. URL `http://dx.doi.org/10.1038/76115`.

FC11 Fang, Z. and Cui, X., Design and validation issues in RNA-seq experiments. *Brief. Bioinform.*, 12,3(2011), pages 280–287. URL `http://dx.doi.org/10.1093/bib/bbr004`.

FM00 Ferragina, P. and Manzini, G., Opportunistic data structures with applications. *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*. IEEE, 2000, pages 390–398.

GFP+11 Grant, G. R., Farkas, M. H., Pizarro, A. D., Lahens, N. F., Schug, J., Brunk, B. P., Stoeckert, C. J., Hogenesch, J. B. and Pierce, E. A., Comparative analysis of RNA-Seq alignment algorithms and the RNA-Seq unified mapper (RUM). *Bioinformatics*, 27,18(2011), pages 2518–2528. URL `http://dx.doi.org/10.1093/bioinformatics/btr427`.

GGGT11 Garber, M., Grabherr, M. G., Guttman, M. and Trapnell, C., Computational methods for transcriptome annotation and quantification using RNA-seq. *Nat. Methods*, 8,6(2011), pages 469–477. URL `http://dx.doi.org/10.1038/nmeth.1613`.

GHR12 Glaus, P., Honkela, A. and Rattray, M., Identifying differentially expressed transcripts from RNA-seq data with biological variation. *Bioinformatics*, 28,13(2012), pages 1721–1728. URL `http://dx.doi.org/10.1093/bioinformatics/bts260`.

GHY+11 Grabherr, M. G., Haas, B. J., Yassour, M., Levin, J. Z., Thompson, D. A., Amit, I., Adiconis, X., Fan, L., Raychowdhury, R., Zeng, Q.,

Chen, Z., Mauceli, E., Hacohen, N., Gnirke, A., Rhind, N., di Palma, F., Birren, B. W., Nusbaum, C., Lindblad-Toh, K., Friedman, N. and Regev, A., Full-length transcriptome assembly from RNA-Seq data without a reference genome. *Nat Biotechnol*, 29,7(2011), pages 644–652. URL http://dx.doi.org/10.1038/nbt.1883.

Gle11 Glenn, T. C., Field guide to next-generation DNA sequencers. *Molecular Ecology Resources*, 11,5(2011), pages 759–769.

GLJ+11 Ge, H., Liu, K., Juan, T., Fang, F., Newman, M. and Hoeck, W., FusionMap: detecting fusion genes from next-generation sequencing data at base-pair resolution. *Bioinformatics*, 27,14(2011), pages 1922–1928. URL http://dx.doi.org/10.1093/bioinformatics/btr310.

Gus97 Gusfield, D., *Algorithms on strings, trees and sequences: computer science and computational biology*. Cambridge University Press, 1997.

HBD10 Hansen, K. D., Brenner, S. E. and Dudoit, S., Biases in Illumina transcriptome sequencing caused by random hexamer priming. *Nucleic Acids Res*, 38,12(2010), page e131. URL http://dx.doi.org/10.1093/nar/gkq224.

HIW12 Hansen, K. D., Irizarry, R. A. and Wu, Z., Removing technical variability in RNA-seq data using conditional quantile normalization. *Biostatistics*, 13,2(2012), pages 204–216. URL http://dx.doi.org/10.1093/biostatistics/kxr054.

HMN09 Homer, N., Merriman, B. and Nelson, S. F., BFAST: an alignment tool for large scale genome resequencing. *PLoS One*, 4,11(2009), page e7767. URL http://dx.doi.org/10.1371/journal.pone.0007767.

HZL+11 Huang, S., Zhang, J., Li, R., Zhang, W., He, Z., Lam, T.-W., Peng, Z. and Yiu, S.-M., SOAPsplice: Genome-wide ab initio detection of splice junctions from RNA-Seq data. *Front Genet*, 2, page 46. URL http://dx.doi.org/10.3389/fgene.2011.00046.

KLS12 Kvam, V. M., Liu, P. and Si, Y., A comparison of statistical methods for detecting differentially expressed genes from RNA-seq data. *Am J Bot*, 99,2(2012), pages 248–256. URL http://dx.doi.org/10.3732/ajb.1100340.

KQGW12 Kogenaru, S., Qing, Y., Guo, Y. and Wang, N., RNA-seq and microarray complement each other in transcriptome profiling. *BMC Genomics*, 13, page 629. URL http://dx.doi.org/10.1186/1471-2164-13-629.

LCSM11 Li, Y., Chien, J., Smith, D. I. and Ma, J., FusionHunter: identifying fusion transcripts in cancer using paired-end RNA-seq. *Bioinformatics*, 27,12(2011), pages 1708–1710. URL http://dx.doi.org/10.1093/bioinformatics/btr265.

LD09 Li, H. and Durbin, R., Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*, 25,14(2009), pages 1754–1760. URL http://dx.doi.org/10.1093/bioinformatics/btp324.

LD11 Li, B. and Dewey, C. N., RSEM: accurate transcript quantification from RNA-Seq data with or without a reference genome. *BMC Bioinformatics*, 12, page 323. URL http://dx.doi.org/10.1186/1471-2105-12-323.

LDH+12 Lin, Y.-Y., Dao, P., Hach, F., Bakhshi, M., Mo, F., Lapuk, A., Collins, C. and Sahinalp, S., CLIIQ: Accurate comparative detection and quantification of expressed isoforms in a population. *Proc Algorithms in Bioinformatics - 12th International Workshop, WABI 2012*, Volume 7534 of Lecture Notes in Computer Science, pages 178–189.

LFJ11 Li, W., Feng, J. and Jiang, T., IsoLasso: a LASSO regression approach to RNA-Seq based transcriptome assembly. *J Comput Biol*, 18,11(2011), pages 1693–1707. URL http://dx.doi.org/10.1089/cmb.2011.0171.

LH10 Li, H. and Homer, N., A survey of sequence alignment algorithms for next-generation sequencing. *Brief Bioinform*, 11,5(2010), pages 473–483. URL http://dx.doi.org/10.1093/bib/bbq015.

LHP+00 Liang, F., Holt, I., Pertea, G., Karamycheva, S., Salzberg, S. L. and Quackenbush, J., Gene index analysis of the human genome estimates approximately 120,000 genes. *Nat Genet*, 25,2(2000), pages 239–240. URL http://dx.doi.org/10.1038/76126.

LJB+11 Li, J. J., Jiang, C.-R., Brown, J. B., Huang, H. and Bickel, P. J., Sparse linear modeling of next-generation mRNA sequencing (RNA-Seq) data for isoform discovery and abundance estimation. *Proc. Natl.*

Acad. Sci. U. S. A., 108,50(2011), pages 19867–19872. URL http://dx.doi.org/10.1073/pnas.1113972108.

LLL+11 Labaj, P. P., Leparc, G. G., Linggi, B. E., Markillie, L. M., Wiley, H. S. and Kreil, D. P., Characterization and improvement of RNA-Seq precision in quantitative transcript expression profiling. *Bioinformatics*, 27,13(2011), pages i383–i391. URL http://dx.doi.org/10.1093/bioinformatics/btr247.

LLL+12 Liu, L., Li, Y., Li, S., Hu, N., He, Y., Pong, R., Lin, D., Lu, L. and Law, M., Comparison of next-generation sequencing systems. *J Biomed Biotechnol*, 2012, page 251364. URL http://dx.doi.org/10.1155/2012/251364.

LRD08 Li, H., Ruan, J. and Durbin, R., Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome Res*, 18,11(2008), pages 1851–1858. URL http://dx.doi.org/10.1101/gr.078212.108.

LTPS09 Langmead, B., Trapnell, C., Pop, M. and Salzberg, S. L., Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol*, 10,3(2009), page R25. URL http://dx.doi.org/10.1186/gb-2009-10-3-r25.

Mar08 Mardis, E. R., Next-generation DNA sequencing methods. *Annu Rev Genomics Hum Genet*, 9, pages 387–402. URL http://dx.doi.org/10.1146/annurev.genom.9.081307.164359.

McG13 McGettigan, P. A., Transcriptomics in the RNA-seq era. *Curr Opin Chem Biol*, 17,1(2013), pages 4–11. URL http://dx.doi.org/10.1016/j.cbpa.2012.12.008.

Met10 Metzker, M. L., Sequencing technologies - the next generation. *Nat Rev Genet*, 11,1(2010), pages 31–46. URL http://dx.doi.org/10.1038/nrg2626.

MS08 Mehlhorn, K. and Sanders, P., *Algorithms and data structures: The basic toolbox*. Springer, 2008.

Mäk10 Mäkinen, V., Välimäki, N., Laaksonen, A. and Katainen, R., Unified view of backward backtracking in short read mapping. *T. Elomaa, H.*

Mannila, and P. Orponen Eds., Algorithms and Applications (Ukkonen Festschrift), Volume 6060 of Lecture Notes in Computer Science, pages 182–195.

MW11 Martin, J. A. and Wang, Z., Next-generation transcriptome assembly. *Nat Rev Genet*, 12,10(2011), pages 671–682. URL `http://dx.doi.org/10.1038/nrg3068`.

NPP+12 Nookaew, I., Papini, M., Pornputtapong, N., Scalcinati, G., Fagerberg, L., Uhlén, M. and Nielsen, J., A comprehensive comparison of RNA-Seq-based transcriptome analysis from reads to differential gene expression and cross-comparison with microarrays: a case study in Saccharomyces cerevisiae. *Nucleic Acids Res*, 40,20(2012), pages 10084–10097. URL `http://dx.doi.org/10.1093/nar/gks804`.

OM11 Ozsolak, F. and Milos, P. M., RNA sequencing: advances, challenges and opportunities. *Nat Rev Genet*, 12,2(2011), pages 87–98. URL `http://dx.doi.org/10.1038/nrg2934`.

ORY10 Oshlack, A., Robinson, M. D. and Young, M. D., From RNA-seq reads to differential expression results. *Genome Biol*, 11,12(2010), page 220. URL `http://dx.doi.org/10.1186/gb-2010-11-12-220`.

PFFS13 Pandey, R. V., Franssen, S. U., Futschik, A. and Schlötterer, C., Allelic imbalance metre (Allim), a new tool for measuring allele-specific gene expression with RNA-seq data. *Mol Ecol Resour*, 13,4(2013), pages 740–745. URL `http://dx.doi.org/10.1111/1755-0998.12110`.

RAW+11 Rozowsky, J., Abyzov, A., Wang, J., Alves, P., Raha, D., Harmanci, A., Leng, J., Bjornson, R., Kong, Y., Kitabayashi, N., Bhardwaj, N., Rubin, M., Snyder, M. and Gerstein, M., AlleleSeq: analysis of allele-specific expression and binding in a network framework. *Mol Syst Biol*, 7, page 522. URL `http://dx.doi.org/10.1038/msb.2011.54`.

RJB+00 Roest Crollius, H., Jaillon, O., Bernot, A., Dasilva, C., Bouneau, L., Fischer, C., Fizames, C., Wincker, P., Brottier, P., Quétier, F., Saurin, W. and Weissenbach, J., Estimate of human gene number provided by genome-wide analysis using Tetraodon nigroviridis DNA sequence. *Nat. Genet.*, 25,2(2000), pages 235–238. URL `http://dx.doi.org/10.1038/76118`.

RKL+13 Rapaport, F., Khanin, R., Liang, Y., Pirun, M., Krek, A., Zumbo, P., Mason, C. E., Socci, N. D. and Betel, D., Comprehensive evaluation of differential gene expression analysis methods for RNA-seq data. *Genome Biol*, 14,9(2013), page R95. URL http://dx.doi.org/10.1186/gb-2013-14-9-r95.

RLD+09 Rumble, S. M., Lacroute, P., Dalca, A. V., Fiume, M., Sidow, A. and Brudno, M., SHRiMP: accurate mapping of short color-space reads. *PLoS Comput Biol*, 5,5(2009), page e1000386. URL http://dx.doi.org/10.1371/journal.pcbi.1000386.

RLSG12 Ribeca, P., Lacroix, V., Sammeth, M. and Guigo, R. *Analysis of RNA Transcripts by High-Throughput RNA Sequencing*, pages 544–554. Wiley-VCH Verlag GmbH & Co. KGaA, 2012. URL http://dx.doi.org/10.1002/9783527636778.ch50.

RMS10 Robinson, M. D., McCarthy, D. J. and Smyth, G. K., edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics*, 26,1(2010), pages 139–140. URL http://dx.doi.org/10.1093/bioinformatics/btp616.

RPTP11 Roberts, A., Pimentel, H., Trapnell, C. and Pachter, L., Identification of novel transcripts in annotated genomes using RNA-seq. *Bioinformatics*, 27,17(2011), pages 2325–2329. URL http://dx.doi.org/10.1093/bioinformatics/btr355.

RSC+10 Robertson, G., Schein, J., Chiu, R., Corbett, R., Field, M., Jackman, S. D., Mungall, K., Lee, S., Okada, H. M., Qian, J. Q., Griffith, M., Raymond, A., Thiessen, N., Cezard, T., Butterfield, Y. S., Newsome, R., Chan, S. K., She, R., Varhol, R., Kamoh, B., Prabhu, A.-L., Tam, A., Zhao, Y., Moore, R. A., Hirst, M., Marra, M. A., Jones, S. J. M., Hoodless, P. A. and Birol, I., De novo assembly and analysis of RNA-seq data. *Nat Methods*, 7,11(2010), pages 909–912. URL http://dx.doi.org/10.1038/nmeth.1517.

RSM06 Rasmussen, K. R., Stoye, J. and Myers, E. W., Efficient q-gram filters for finding all epsilon-matches over a given length. *J Comput Biol*, 13,2(2006), pages 296–308. URL http://dx.doi.org/10.1089/cmb.2006.13.296.

RTD+11 Roberts, A., Trapnell, C., Donaghey, J., Rinn, J. L. and Pachter, L., Improving RNA-Seq expression estimates by correcting for fragment bias. *Genome Biol*, 12,3(2011), page R22. URL http://dx.doi.org/10.1186/gb-2011-12-3-r22.

SHP+10 Sboner, A., Habegger, L., Pflueger, D., Terry, S., Chen, D. Z., Rozowsky, J. S., Tewari, A. K., Kitabayashi, N., Moss, B. J., Chee, M. S., Demichelis, F., Rubin, M. A. and Gerstein, M. B., FusionSeq: a modular framework for finding gene fusions by analyzing paired-end RNA-sequencing data. *Genome Biol*, 11,10(2010), page R104. URL http://dx.doi.org/10.1186/gb-2010-11-10-r104.

SVMC09 Scott, C. P., VanWye, J., McDonald, M. D. and Crawford, D. L., Technical analysis of cDNA microarrays. *PLoS One*, 4,2(2009), page e4486. URL http://dx.doi.org/10.1371/journal.pone.0004486.

SW81 Smith, T. F. and Waterman, M. S., Identification of common molecular subsequences. *J Mol Biol*, 147,1(1981), pages 195–197.

THS+13 Trapnell, C., Hendrickson, D. G., Sauvageau, M., Goff, L., Rinn, J. L. and Pachter, L., Differential analysis of gene regulation at transcript resolution with RNA-seq. *Nat Biotechnol*, 31,1(2013), pages 46–53. URL http://dx.doi.org/10.1038/nbt.2450.

TKRM13a Tomescu, A. I., Kuosmanen, A., Rizzi, R. and Mäkinen, V., A novel combinatorial method for estimating transcript expression with RNA-Seq: Bounding the number of paths. *Proc Algorithms in Bioinformatics - 13th International Workshop, WABI 2013*, Volume 8126 of Lecture Notes in Computer Science, pages 85–98.

TKRM13b Tomescu, A. I., Kuosmanen, A., Rizzi, R. and Mäkinen, V., A novel min-cost flow method for estimating transcript expression with RNA-Seq. *BMC Bioinformatics*, 14 Suppl 5, page S15. URL http://dx.doi.org/10.1186/1471-2105-14-S5-S15.

TPS09 Trapnell, C., Pachter, L. and Salzberg, S. L., TopHat: discovering splice junctions with RNA-Seq. *Bioinformatics*, 25,9(2009), pages 1105–1111. URL http://dx.doi.org/10.1093/bioinformatics/btp120.

TSG+11 Turro, E., Su, S.-Y., Gonçalves, n., Coin, L. J. M., Richardson, S. and Lewin, A., Haplotype and isoform specific expression estimation using multi-mapping RNA-seq reads. *Genome Biol*, 12,2(2011), page R13. URL http://dx.doi.org/10.1186/gb-2011-12-2-r13.

VDD09 Voelkerding, K. V., Dames, S. A. and Durtschi, J. D., Next-generation sequencing: from basic research to diagnostics. *Clin Chem*, 55,4(2009), pages 641–658. URL http://dx.doi.org/10.1373/clinchem.2008.112789.

VHPV13 Van Verk, M. C., Hickman, R., Pieterse, C. M. J. and Van Wees, S. C. M., RNA-Seq: revelation of the messengers. *Trends Plant Sci*, 18,4(2013), pages 175–179. URL http://dx.doi.org/10.1016/j.tplants.2013.02.001.

WGS09 Wang, Z., Gerstein, M. and Snyder, M., RNA-Seq: a revolutionary tool for transcriptomics. *Nat Rev Genet*, 10,1(2009), pages 57–63. URL http://dx.doi.org/10.1038/nrg2484.

WL09 Wilhelm, B. T. and Landry, J.-R., RNA-Seq — quantitative measurement of expression through massively parallel RNA-sequencing. *Methods*, 48,3(2009), pages 249–257. URL http://dx.doi.org/10.1016/j.ymeth.2009.03.016.

WN10 Wu, T. D. and Nacu, S., Fast and SNP-tolerant detection of complex variants and splicing in short reads. *Bioinformatics*, 26,7(2010), pages 873–881. URL http://dx.doi.org/10.1093/bioinformatics/btq057.

WSZ+10 Wang, K., Singh, D., Zeng, Z., Coleman, S. J., Huang, Y., Savich, G. L., He, X., Mieczkowski, P., Grimm, S. A., Perou, C. M., MacLeod, J. N., Chiang, D. Y., Prins, J. F. and Liu, J., MapSplice: accurate mapping of RNA-seq reads for splice junction discovery. *Nucleic Acids Res*, 38,18(2010), page e178. URL http://dx.doi.org/10.1093/nar/gkq622.

WWL12 Wang, L., Wang, S. and Li, W., RSeQC: quality control of RNA-seq experiments. *Bioinformatics*, 28,16(2012), pages 2184–2185. URL http://dx.doi.org/10.1093/bioinformatics/bts356.

WWZ10 Wang, X., Wu, Z. and Zhang, X., Isoform abundance inference provides a more accurate estimation of gene expression levels in RNA-seq. *J Bioinform Comput Biol*, 8 Suppl 1, pages 177–192.

YMW+12 Young, M. D., McCarthy, D. J., Wakefield, M. J., Smyth, G. K., Oshlack, A. and Robinson, M. D., Differential expression for RNA sequencing (RNA-seq) data: Mapping, summarization, statistical analysis, and experimental design. In *Bioinformatics for High Throughput Sequencing*, RodrÃguez-Ezpeleta, N., Hackenberg, M. and Aransay, A. M., editors, Springer New York, 2012, pages 169–190, URL `http://dx.doi.org/10.1007/978-1-4614-0782-9_10`.

ZB08 Zerbino, D. R. and Birney, E., Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome Res*, 18,5(2008), pages 821–829. URL `http://dx.doi.org/10.1101/gr.074492.107`.

Appendix 1. File formats

SAM format: SAM stands for Sequence Alignment/Map format. It is a tab-delimited text format for storing alignment data. Many alignment software output SAM format. It consists of 11 fields:

1. QNAME: The name of the query template (i.e. read ID)

2. FLAG: Bitwise flag containing for example the strand of the alignment and whether the read is paired or not.

3. RNAME: Name of the reference sequence to which the read aligned.

4. POS: Leftmost mapping position (1-based).

5. MAPQ: Mapping quality

6. CIGAR: String that represents the alignment consisting of '=' (match), 'X' (mismatch), 'M' (match or mismatch), 'I' (insertion), 'D' (deletion), 'N' (intron) and 'S' and 'H' (soft and hard clip).

7. RNEXT: Name of the reference where mate aligned (not applicable for single-end reads).

8. PNEXT: Leftmost mapping position where mate aligned (not applicable for single-end reads).

9. TLEN: Observed template length. For paired-end reads, the distance between leftmost mapping position and rightmost mapping position. 0 for single-end reads.

10. SEQ: The sequence.

11. QUAL: The basewise qualities of the sequence.

Example:

read.1 16 17 2094475 255 26M972N49M * 0 0
GGTTGATTCAAATTGCCAACTCTCTCATGATAGCTGGAAAGTCCAGGATA
*

BAM format: BAM format is the compressed, binary form of SAM. It can be indexed to allow for fast retrieval of data at any given position.

BED format: BED file is a tab-delimited text file that defines a feature track (e.g. gene, exon or junction). It consists of three required and nine optional fields. The required fields are:

1. chrom: Name of the chromosome or scaffold.

2. chromStart: Starting position of the feature in the chromosome or scaffold (0-based).

3. chromEnd: Ending position of the feature in the chromosome or scaffold. Exclusive (i.e. if chromEnd=100, last base of the feature is 99.)

And the optional fields are:

4. Name: Name of the feature.

5. Score: A score between 0 and 1000. Defines the shade of the feature in IGV genome browser.

6. Strand: The strand of the feature, + (forward), - (reverse) or . (unknown).

7. thickStart: Start position of thickly-drawn feature.

8. thickEnd: End position of thickly-drawn feature.

9. itemRGB: Display color of the feature in IGV genome browser in form of value tuple (R, G, B).

10. blockCount: The number of blocks in the feature (e.g. exons).

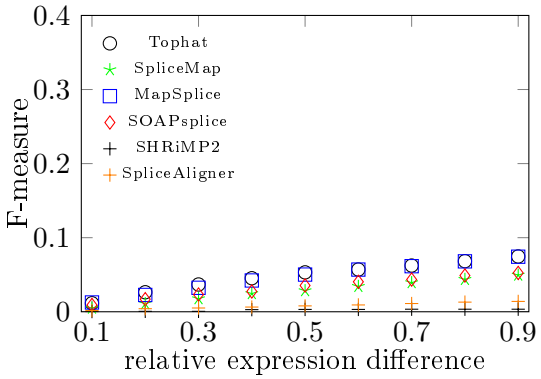
11. blockSizes: Sizes of the blocks in the feature.

12. blockStarts: Starts of the blocks in the feature compared to the starting position of the feature.

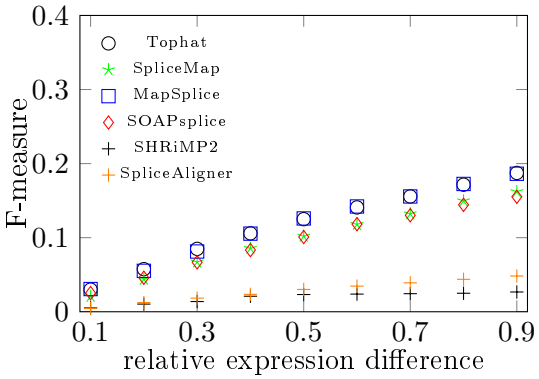
Example:

10 71923616 7193724 JUNC1 + 71923616 7193724 255,0,0 2 46,20 0,129

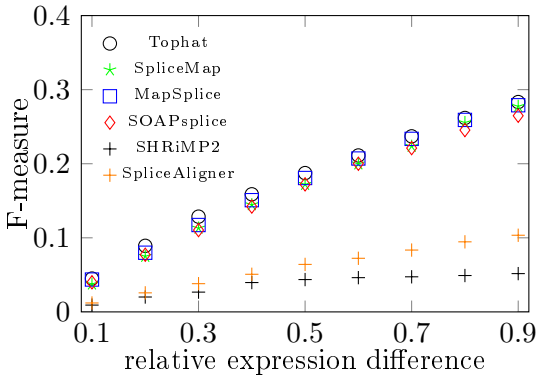
Appendix 2. Transcript prediction 2D plots



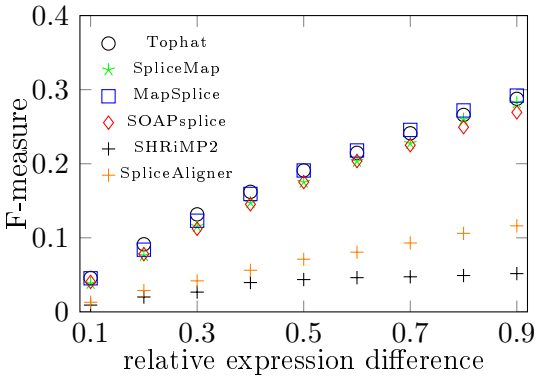
(a) sequence dissimilarity threshold 0.1



(b) sequence dissimilarity threshold 0.4

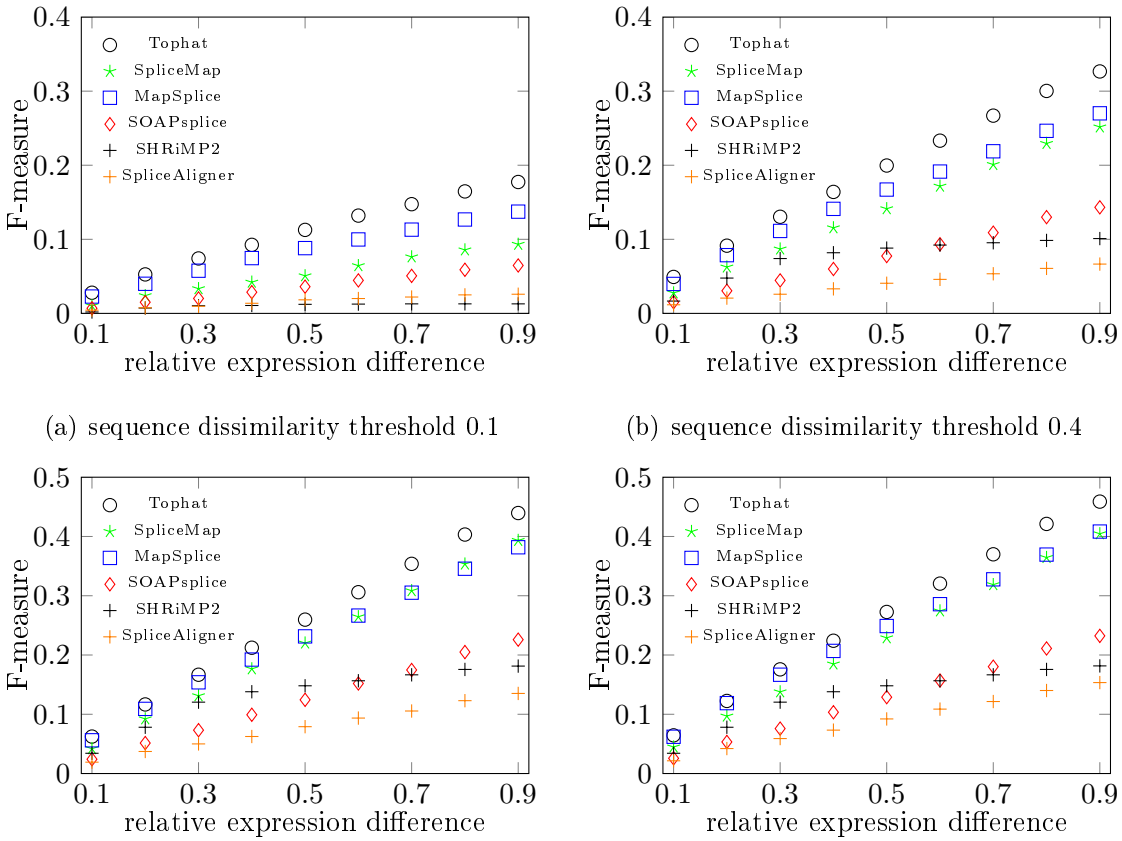


(c) sequence dissimilarity threshold 0.7



(d) sequence dissimilarity threshold 0.9

Figure 27: F-measure for the transcripts predicted from 50 bp single-end data at various sequence dissimilarity levels. Corresponds to 3D plot Figure 22.



(a) sequence dissimilarity threshold 0.1

(b) sequence dissimilarity threshold 0.4

(c) sequence dissimilarity threshold 0.7

(d) sequence dissimilarity threshold 0.9

Figure 28: F-measure for the transcripts predicted from 150 bp single-end data at various sequence dissimilarity levels. Corresponds to 3D plot Figure 23.