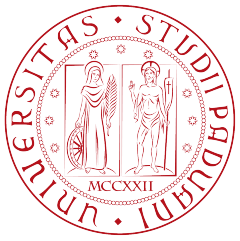

UNIVERSITY OF PADOVA
DEPARTMENT OF INFORMATION ENGINEERING

Ph.D. Course in Information Engineering
Curriculum: Bioengineering
Series: XXIX

Improving the RNA-Seq analysis pipeline: read alignment and expression level quantification

Ph.D. Candidate: Giacomo BARUZZO

Advisor: Prof. Barbara DI CAMILLO

Co-Advisor: Francesca FINOTELLO

Course director: Prof. Matteo BERTOCCO

Coordinator: Prof. Giovanni SPARACINO

This dissertation is submitted for the degree of
Philosophiae Doctor

January 2017

A Gianni e Alessandra

"The way we do RNA-Seq now is... you take the transcriptome,
you blow it up to pieces and then you try to figure out
how they all go together again... if you think about it,
it's kind of a crazy way to do things."

Michael Snyder

Nawy, Tal. "*Sequencing: End-to-end RNA sequencing.*"
Nature Methods 10.12 (2013): 1144-1145.

Abstract

DNA and RNA play an essential role in the life of each living organism. The two molecules have different characteristics and properties but their functions are strictly related. DNA encodes all the genetic instructions needed by the main cell activities in the so-called genome. DNA is related to RNA through the gene expression process, which transcribes the information encoded by DNA into RNAs. Opposite to the static information provided by DNA, the set of transcribed RNAs at a specific instant represents the current state of each cell and, at the end, it provides a dynamic characterization of its activity. For this reason, transcriptome analysis represents a powerful tool to identify the dynamic behavior of an organism, such as the response to environmental stimuli or the pathological mechanisms involved in diseases.

In recent years, transcriptomic analyses were revolutionized by the advent of *RNA sequencing* (*RNA-Seq*), a new methodology that applies current Next Generation Sequencing (NGS) techniques to RNA molecules. RNA-Seq enables to investigate at high resolution all the RNA species present in a sample, characterizing their sequences and quantifying their abundances at the same time. In practice, millions of short transcript sub-sequences, called *reads*, are sequenced from random positions of the input RNAs using the same NGS platforms employed in DNA sequencing. Unfortunately, no information is provided about which transcripts have generated the reads or from which part of the transcripts they come from. For this reason, reads represent at the same time the output of the sequencing process and the input of complex RNA-Seq data analysis pipelines. The first task in many RNA-Seq data analysis pipelines consists in identifying the relation between the sequencing output (i.e. reads) and the sequenced transcripts. The most common approach to this problem consists in aligning the reads against a reference genome. Once the reads are positioned in the genome, it is possible to infer which transcripts have generated them analyzing the read locations. The information coming from the positions and the number of reads could be employed in a wide range of downstream analyses. For example, counting the number of reads aligned to a gene could give a measure of its expression level, whereas studying which reads are located across exon junction could identify different isoforms. At first glance, these tasks may seem very

simple, but the implementation of both the single steps and the whole analysis workflow are in fact complex and still not well defined.

The aim of this Ph.D. research project was the improvement and the assessment of the computational methods involved in the RNA-Seq data analysis pipeline. Among all the analysis steps in the pipeline, this thesis is focused on the read alignment problem. Read alignment is identified as one of the most critical steps, both for its almost ubiquitous presence in the different RNA-Seq analysis workflows and for its complexity. The study of this pivotal task was carried out through several steps. First, a complete characterization of the problem was performed, analyzing the alignment challenges both from a methodological and a computational point of view. In addition, the algorithms and data structures employed in the alignment process were analyzed together with different ways of modeling the read alignment problem. Then, state of the art methods for RNA-Seq read alignment were identified performing a thorough literature search about RNA-Seq, which revealed the presence of many available methods. At the same time, the literature search highlighted that the identification of a suitable alignment method for a specific application is challenging, mainly due to the lack of accurate comparative analyses. Thus, a comprehensive benchmark analysis of fourteen splice aware alignment methods and four splice unaware tools was designed and performed. The simulation of several datasets describing real scenarios and the definition of a comprehensive set of accuracy and efficiency metrics were performed in order to assess the different alignment methods. The assessment revealed considerable differences between methods' performance, highlighting often a poor correlation between accuracy and popularity. Finally, the effect of the alignment accuracy on the reliability of an expression level quantification study was assessed for a subset of alignment methods.

Overall, this thesis considers the RNA-Seq read alignment problem and presents a thorough characterization of its characteristics and challenges. In a fast evolving research field such as RNA-Seq, the information resulting from the assessment of state of the art methods provides some valuable guidelines for the definition of robust and accurate analysis pipelines.

Ringraziamenti

Durante questi ultimi tre anni, molte persone mi hanno aiutato e accompagnato nel mio percorso di dottorato. La loro presenza e il loro contributo hanno reso possibile questo lavoro di tesi e mi sento di dover loro un ringraziamento speciale.

Vorrei in primo luogo ringraziare mio padre Gianni e mia madre Alessandra, per il loro continuo e incondizionato supporto. La loro vicinanza e il loro affetto sono stati fonte di forza e incentivo a migliorare continuamente come uomo e come ricercatore.

Un ringraziamento speciale va al mio supervisore Barbara per l'aiuto e la guida in questi tre anni di lavoro insieme. I risultati ottenuti e la mia crescita accademica e professionale sono merito del suo ottimo lavoro e della sua grande competenza.

Un grazie ai membri del mio gruppo di ricerca Alberto, Alessandra, Elif, Francesco, Gianna, Ilaria e Tiziana. Un grazie particolare a Francesca, che è stata la mia guida nel primo periodo e da cui ho imparato moltissimo.

Un ringraziamento agli amici e colleghi Alberto G., Alberto D., Alessandra, Alessandro, Alessia, Chiara, Davide, Elif, Erica, Francesca, Giacomo C., Giacomo T., Giada, Ilaria M., Ilaria P., Marco, Maria, Martina, Matteo e Tiziana per i tanti bei momenti passati insieme.

La bioinformatica è un campo di ricerca che richiede notevole potenza di calcolo e una complessa gestione delle risorse computazionali. Il lavoro svolto in questi anni è stato reso possibile dalla competenza e disponibilità di Paolo, che ringrazio infinitamente per il fondamentale aiuto nella gestione delle piattaforme di calcolo.

Un grandissimo ringraziamento e profonda gratitudine a Greg per l'opportunità che mi ha concesso ospitandomi nel suo gruppo di ricerca. Una buona parte di questa tesi è nata dal lavoro svolto nei sei fantastici mesi trascorsi a Philadelphia, periodo che mi ha fatto crescere umanamente e professionalmente.

Il lavoro sui metodi di allineamento non sarebbe stato lo stesso senza il fondamentale contributo di Katharina, che ringrazio per la disponibilità, l'eccellente lavoro e le tante ore passate con me a sviluppare questo importante progetto.

Un grazie particolare anche agli altri autori, Eun Ji e Garret, per aver reso possibile questo fantastico lavoro. Il periodo a Philadelphia è stato reso speciale anche e soprattutto per le

splendide persone che ho conosciuto lì tra cui voglio ricordare, in aggiunta a chi già citato sopra, Anand, Dimitra, Elisabetta, Emanuela, Faith e Nick.

Da ultimo, ma non per importanza, un ringraziamento lo voglio riservare ai miei amici, sia dentro che fuori l'università, che mi sono stati sempre vicini in questi anni.

Table of contents

Chapter 1

Introduction

In every living organism, DNA molecules encode all the genetic instructions required to build the RNAs and proteins that are needed to the growth, development, functioning and reproduction of the cell. However, DNA represents a static set of instructions and so is not able to explain the dynamic behavior of an organism, such as the evolution through diverse developmental stages or the response to environmental stimuli. Great part of organisms' dynamicity and complexity is indeed explained by gene expression, i.e. the process that transforms the information encoded in the genome into a functional product. In the gene expression process, each cell can activate a specific set of genes required for executing regulatory actions as well as functions or responses to stimuli; the genes are activated through their transcription into RNA copies. Since RNA transcription is the activation of several processes and control mechanisms that make up the complex gene expression machinery, substantial insights can be drawn from the study of organisms' transcriptomes.

Hybridization-based approaches such as microarrays, have been the most used solutions for gene expression profiling and differential expression (DE) analysis for many years, thanks to their high throughput and relatively low costs [1].

In 2008, the advent of a new methodology called *RNA sequencing* (RNA-Seq), has revolutionized transcriptomics research enabling the simultaneous characterization of the sequences of the transcripts present in a cell and the quantification of their expression levels [1–5]. Compared to previous approaches, RNA-Seq methodology offers several key advantages:

- It is not limited to the detection of transcripts corresponding to well-annotated genomic sequences, but can be used to perform novel transcript discovery or to sequence non-model organisms;

- It does not have an upper limit for quantification, thus ensuring a large dynamic range of expression levels over which transcripts can be detected;
- It achieves high levels of reproducibility for technical replicates [2, 6];
- Transcript sequences can be read at single-base level.

The possibility of sequencing transcriptomes at single-base resolution is borrowed from Next-Generation Sequencing (NGS) platforms, which represent the technological framework of the RNA-Seq methodology. Next-Generation Sequencing technologies produce enormous amount of data, enabling to sequence entire genomes and transcriptomes in a single instrument run at dramatically reduced time and costs. In practice, millions of short sequences, called *reads*, are sequenced from random positions of the input RNAs using the same NGS platforms employed in DNA sequencing.

Despite being already widely used, RNA-Seq is a very recent methodology that is experiencing a fast and continuous development of both experimental and computational procedures. In particular, the number of available methods for performing each step of RNA-Seq data analysis has grown at such a fast pace so to prevent the definition of a unified and standardized computational pipeline.

In this scenario, the research described in this thesis was originally motivated by the need of identifying robust computational methods for the analysis of RNA-Seq data, focusing on one of the most critical and widely performed tasks such as the alignment of RNA-Seq reads. All the contributions described in this thesis, and related to the study, improvement and assessment of the RNA-Seq analysis pipeline, are the results of the research activity carried out within this Ph.D. program.

This chapter provides a brief overview of the RNA-Seq read alignment problem, highlighting challenges and open issues. The motivations of the research are described together with the aim and the main objectives of the thesis. In addition, the structure of the thesis is presented in the last section.

1.1 Motivations and objectives

RNA-Seq provides nearly unlimited possibilities in modern transcriptome analyses, resulting in a wide variety of applications. However, the study of high-throughput RNA sequencing data requires the design of sophisticated analysis pipelines involving many computational steps.

While many downstream steps in the pipeline are specific for the particular application, few preliminary tasks are common to the great majority of studies. These common steps involve the processing of sequencing reads and have a pivotal role in the accuracy and reliability of the several downstream analyses.

The millions of reads produced by current NGS technologies are sequenced from random positions of the input RNAs. So, the output of the sequencing process are just millions of transcript sub-sequences, but no information is provided about which transcripts have generated them or from which part of the transcripts they come from. Therefore, the first step in any RNA-Seq data analysis consists in identifying the relation between the available reads and the transcripts present in the sequenced sample.

The most common approach to this problem consists in aligning the reads against a reference sequence, such as a genome or a transcriptome. Once the reads are positioned in the genome, it is possible to infer which transcripts have generated them by analyzing the read locations. The information coming from the positions and the number of reads are then employed by the different downstream analyses. For example, counting the number of reads aligned to a gene could give a measure of its expression level, whereas studying which reads are located across an exon junction could identify different isoforms. At first glance, identifying the position of each read in the reference genome may seem very simple, but the alignment of RNA-Seq reads is in fact complex and challenging, both in terms of accuracy and efficiency.

The source of such complexity are both biological and technical factors. For example, the presence of low complexity regions and polymorphisms such as insertions, deletions and single nucleotide polymorphisms (SNP) makes challenging the identification of a perfect match between the read and the reference sequence. Furthermore, the sequencing process is not error free, resulting in reads that are not perfect copies of the transcript sub-sequence which generated them and requiring some flexibility during the alignment process. In addition, the presence of splicing and alternative splicing makes it impossible to find a continuous location in the genome for reads covering an exon junction. Indeed, the removal of introns during the splicing process requires the identification of long gaps during the alignment of such reads against a genome. In particular, it is the need for RNA-Seq alignment methods to handle intronic gaps (i.e. *splice awareness*) which makes the alignment problem even harder.

From the computational point of view, the sizes of the reference genomes (from hundreds of thousands up to billions of bases) and the large amount of reads (tens or hundreds of millions) makes the read mapping a computationally intensive task. Typically, finding the correct positions of available reads in a large reference genome requires tens of GB of RAM and many CPU hours on a multicore machine.

In order to achieve feasible resource requirement and provide a rigorous modeling of the mapping process, current alignment methods exploit efficient data structures and algorithms employed in many string alignment contexts. The first generation of alignment methods employed a hash table to create an index of the genome, using short nucleotide sequences as table's keys and coding in each table's entry the locations of the associated key. The second generation of mapping methods exploited suffix arrays and suffix trees to index the genome and modeled the alignment process as the traversal of such data structures. Both class of methods have strengths and weaknesses and allow achieving different trade-off between efficiency and accuracy. Currently, the modeling of the alignment process and the research of efficient data structures, algorithms and strategies are still open problems.

Due to the pivotal importance of read alignment in the RNA-Seq analysis pipeline and the challenges in terms of accuracy and efficiency described above, in recent years there was a considerable effort of the research community in the development of splice aware alignment methods. However, the complexity of the alignment problem and the increasing number of available methods have prevented the definition of a unified and standardized computational pipeline and the possibility to assess the performance of the different methods. In this uncertain scenario, this thesis is aimed at achieving the following objectives:

- a thorough definition of the read alignment problem, identifying the most important characteristics and challenges
- the identification of the state of the art RNA-Seq read alignment methods and the assessment of their performance, both in terms of alignment accuracy and efficiency
- the assessment of the role of RNA-Seq read alignment in the accuracy of expression level quantification analyses

These objectives are designed to assess and improve the reliability of the alignment step consequently increasing the accuracy of the overall RNA-Seq analysis pipeline.

1.2 Structure of the thesis

In order to better outline the RNA-Seq scenario, in Chapter 2 some prerequisite topics such as transcription and gene expression, Next Generation Sequencing technology, RNA-Seq experimental procedures and data analysis are introduced. In Chapter 3, the read alignment problem is presented, analyzing its characteristics and challenges from both a methodological and a computational complexity point of view. In addition, the available alternatives to read alignment are briefly described and discussed. State of the art methods for RNA-Seq read alignment are then identified performing a thorough literature search that involved more than 2000 peer reviewed publications in the context of RNA-Seq. The literature search highlights that RNA-Seq alignment, a common task in the great majority of RNA-Seq analysis pipelines, remains in a state of confusion mainly due to a lack of accurate and systematic benchmarking studies. With the purpose of addressing this issue, in Chapter 4 a thorough benchmark analysis of read alignment methods is designed. In this chapter, the simulation of datasets representing several real scenarios is described together with the definition of a complete set of accuracy and efficiency metrics. In Chapter 5, these metrics and datasets are then employed to assess the performance of fourteen splice aware algorithms. The same analysis is performed also on four splice unaware tools and it is described in Chapter 6. The effects of employing some of these mapping methods on the accuracy of a common downstream analysis are then assessed in Chapter 7. Specifically, the effects on the expression level quantification are investigated, using both real and simulated data together with different read preprocessing policies and read alignment strategies. Finally, strengths, limitations and future developments of the present study are discussed in Chapter 8.

Chapter 2

RNA sequencing background

Four topics are at the basis of this thesis: RNA and gene expression, Next-Generation Sequencing technologies, RNA-Seq experiment and RNA-Seq data analysis pipelines.

RNA and gene expression represent the biological context of a RNA-Seq study. The identification and quantification of transcripts performed through the RNA-Seq methodology enable a thorough characterization of the cell's activities. Due to the fundamental role of RNA in many cell's functions and processes, the study of this molecule has a pivotal role in many research fields such as medicine, biology and pharmacology.

Next-Generation Sequencing technologies represent the technological framework of the RNA-Seq methodology and they have a major role in many strengths and weaknesses of this novel methodology. Several NGS platforms exist, each one achieving unique characteristics and features that should be related to the specific objective of the RNA-Seq study.

RNA sequencing shares many experimental steps with DNA sequencing and requires only few specific experimental procedures. Unfortunately, both the common and specific steps involved in a RNA-Seq experiment are the source of many biases that are impossible to eliminate. Therefore, it is important to know the different sources of bias and design the experiment to minimize the ones which could impact the final results.

Once the experiment is completed, RNA-Seq data are processed through a complex analysis pipeline. Several step and computational methods are involved in the analysis of RNA-Seq data, in order to both mitigate the experimental biases and fully exploit the information contained in the sequenced sample.

The RNA-Seq experiment and data analysis as well as the employed Next-Generation Sequencing technologies are strictly related to the overall accuracy and reliability of a RNA-Seq study. For this reason, an introduction to these topics is provided in the following sections.

2.1 Biological context

In all known living organisms, DNA stores the whole information needed to make functioning the cells. DNA (deoxyribonucleic acid) is a nucleic acid and its molecule is composed by monomer units called nucleotides; each nucleotide consists of a deoxyribose sugar, a phosphate group and a nitrogenous base. Nucleotides can be of four different kinds, depending on the base that they comprise: adenine (A), cytosine (C), guanine (G) or thymine (T). The nucleotides are concatenated to one other by a phosphodiester bond which joins the 5' end on one nucleotide to the 3' end of the previous one, forming a strand. In each molecule, two strands are then bound together through hydrogen bonds between the corresponding nitrogenous bases, following a precise base pairing rule: adenine only matches thymine, while cytosine only binds to guanine. Due to the fixed base pairing rule, the two strands are complementary since one DNA strand could univocally determine the sequence of its antiparallel strand. The DNA molecule stores the biological information through the specific sequence in which the four nucleotides appear, encoding in such sequences all the organism's hereditary information.

A specific DNA region which encodes a particular functional product or a regulatory function is called *gene* and the process which converts the encoded information into the final product is called *gene expression*. The conversion is performed through a first step of *transcription* in which the genetic information encoded in the DNA is transcribed into a RNA molecule. Similarly to DNA, Ribonucleic acid (RNA), is a nucleic acid and has a very similar chemical structure. However, RNA differs from DNA:

- strand structure: RNA is single stranded, DNA is double stranded
- base composition: RNA contains uracil (U) in place of thymine
- sugar composition: RNA contains ribose sugar instead of deoxyribose, which makes it less stable than DNA

Depending of the final product, the transcribed RNAs molecules could be classified as:

- coding RNA: RNAs whose final product is a protein. The results of transcription is a messenger RNA (mRNA), which would be decoded by a ribosome into an amino acid sequence through the *translation* process. In eukaryotic organisms, the result of transcription is a precursor mRNA (pre-mRNA) which would be processed to obtain the final mRNA.
- non-coding RNA: RNAs which do not code for a protein. Non-coding RNAs (ncRNAs) have control and regulatory functions and are involved in many cellular processes.

Examples of ncRNAs are ribosomal RNAs (rRNAs), transfer RNAs (tRNAs), microRNAs (miRNAs), piwi-interacting RNAs (piRNAs), small interfering RNAs (siRNAs), small nuclear RNAs (snRNAs), small nucleoar RNAs (snoRNAs) and long ncRNAs (lncRNAs).

In regards to coding RNA, mRNAs have a pivotal role in the translation of the genomic information encoded in the DNA (genotype) into the physical characteristics of an organism (phenotype). Proteins are one of the most important macromolecules in a living organism, accomplishing several functions such as responding to stimuli, transporting molecules and catalyzing metabolic reactions. Although several molecules, such as minerals, water and fats, shape the organisms' cells, proteins provide the framework for their correct functioning and organization. Different from prokaryotes transcription, which create a mature mRNA, in eukaryotic organisms the result of transcription is a pre-mRNA which has to undergo some post-transcriptional modifications. The first modification consists in the addition of a methylated guanine nucleotide at the 5' end of RNA, through a process called "capping"; the 5' methylated cap would help the cell to recognize mRNA from other molecules and protects it from degradation. The 3' end of mRNA is modified as well, adding a long tail of adenine bases, called poly-A tail. Similar to capping, this modification prevents mRNA to be quickly degraded: the longer the poly-A tail is, the longer the mRNA resists to degradation and the more it is translated into proteins. The last modification step is called *splicing* and consists in removing the non-coding regions (i.e. introns) and concatenating together the remaining coding sequences, called exons. During the splicing process, the order of exons is always preserved, while some exons can be removed along with introns, giving rise to different RNAs. This process, called *alternative splicing*, enables the production of different proteins (*isoforms*) starting from the same gene, dramatically increasing the coding potential of eukaryotic genomes. For example, in human ~95% of genes having more than one exon are alternatively spliced [7], allowing to produce more than 200000 protein coding transcripts from about 22000 genes [8].

On the other hand, non-coding RNAs are not translated into a protein but they have a fundamental role in cell development and differentiation. In addition, they are involved in several cellular processes such as RNA processing (snRNAS and snoRNAs), translation (rRNAs and tRNAs), gene expression and transcription (lncRNAs) [9–12].

2.2 Next Generation Sequencing

Since the discovery of the structure of DNA [13], huge progress have been made in understanding the complexity and diversity of living organism genomes. In this context, a major role is played by DNA sequencing, i.e. the determination of the precise order of nucleotides that constitute a DNA molecule. The first sequencing technique was developed in 1975 by Frederick Sanger [14], in which E. coli DNA polymerase was used to copy single-stranded DNA molecules. Just two years later, Sanger's group was able to sequence the first genome using this technique [15]. However, this sequencing method achieved low automation and throughput, allowing to sequence just few hundreds of nucleotides at a time.

The major breakthrough happened some time later, when the same research group introduced the "dideoxy chain-termination" method for sequencing DNA molecules, also known as the "Sanger method" or "Chain termination method" [16]. The new method allowed a faster and more accurate DNA sequencing, even if the low throughput still remained the main issues.

In early 90', the advent of capillary electrophoresis defined the first milestone in the so-called "high-throughput" sequencing, allowing to sequence up to 96 DNA sequences in parallel. The method records the light emission signal produced by labelled deoxynucleotides (dNTPs) during the synthesis of the DNA template complementary strand. Finally, an algorithm translates the recorded fluorescent emissions into DNA sequences, called "reads".

Between 2007-2008, the continuous trend in increasing the sequencing throughput leaded to the development of the so called "Next-Generation Sequencing" (NGS) technologies, which at the same time greatly reduced sequencing costs and increased the throughput by a factor of 100-1000 [17, 18]. The main advantage of this new sequencing techniques is a massive parallelization which allows sequencing of millions of fragments at the same time. The high parallelization reduces the costs due to the reagents needed and drastically increases the throughput per run.

Nowadays, these advances have brought the cost of sequencing a human genome down to around US$1,000 and they have enabled the use of sequencing as a clinical tool [19]. On the other hand, these advancements are not without limitations since some existing problems are exacerbated and new problems arise. For example, data produced by NGS platforms show a higher error rate (0.1–15%) and read lengths generally shorter (35–700 bp for short-read approaches) than those of traditional Sanger sequencing platforms [20]. Although some long-read sequencing technologies mitigate the length limitation, they remain considerably more expensive and at the same time show lower throughput than other platforms. A complete characterization of modern NGS technologies is presented in the next section.

2.2.1 Next Generation Sequencing platforms review

Many NGS platforms share some common steps, such as DNA template fragmentation, adapter ligation and clonal amplification. However, each technology exploits different techniques and strategies [21–23].

Short read next generation sequencing technologies fall under two broad classes: sequencing by synthesis (SBS) and sequencing by ligation (SBL). In the first class of methods, a polymerase is used and a signal (e.g. a fluorophore or a change in ionic concentration) reveals the strand elongation through the incorporation of a nucleotide. In SBL approaches, the DNA fragment is hybridized with a probe sequence bound to a fluorophore which is then ligated to an adjacent oligonucleotide for imaging. The base complementary to specific positions within the probe is then identified through the emission spectrum of the fluorophore. In order to better distinguish signal from background noise, both classes of methods usually create many identical copies of the DNA fragment exploiting bead-based or solid-state generation strategies (Figure 2.1).

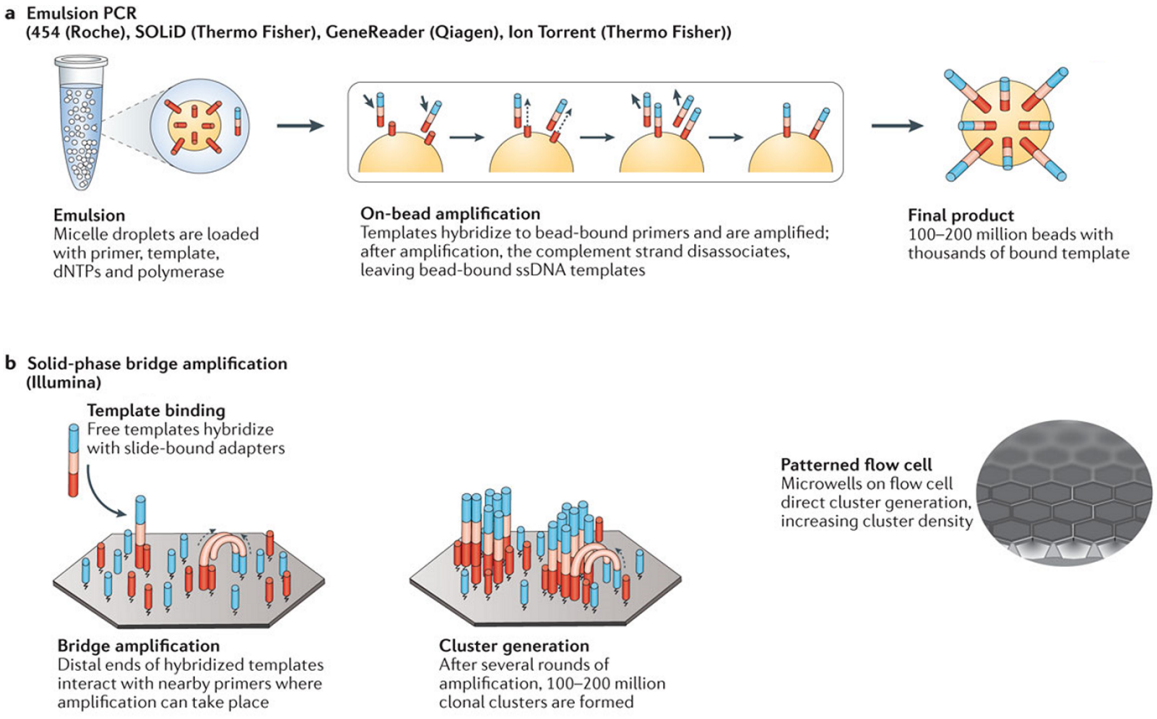

Fig. 2.1 *Amplification strategies* **a)** Bead-based amplification methods, **b)** Solid state amplification methods. Image taken from [23].

Bead-based preparation methods use beads covered with deoxynucleotides (dNTPs), primers, DNA polymerase and oligonucleotide fragments complementary to adapters. DNA templates are then ligated to adapters sequences and are captured in micelles along with

the beads. Finally, emulsion PCR (emPCR) [24] is performed within the micelle, resulting in many copies of the same DNA template on each bead. In solid-state methods [25] the amplification is performed directly on a solid support containing forward and reverse primers complementary to DNA templates. PCR is then used to create the second strand from the immobilized primers and the unbound DNA templates are then removed. Template concentration and location should be properly handled to avoid overlapping clonal cluster. In order to increase the cluster density, and consequently the sequencing throughput, some NGS platforms have recently introduced *patterned flow cells*. Patterned flow cells are prearranged microwells on the solid support that optimize the cluster spacing. The area between the microwells is devoid of DNA probes, ensuring that DNA clusters only form within the wells. The microwells fixed locations provide even and consistent spacing between adjacent clusters, allowing accurate resolution of clusters during imaging.

In the context of long-read technologies, the leading approach is single-molecule real-time (SMRT) sequencing, in which the sequence of nucleotides is detected in real-time. SMRT methods differ from short-read approaches in that they do not require chemical cycling for each dNTP added and they do not perform a clonal amplification of DNA fragments to generate a stronger signal.

In the next sections are described four short-read platforms (SOLiD, Illumina, 454 and Ion Torrent) and two long-read platforms (PacBio and Oxford Nanopore).

SOLiD

The SOLiD platform exploits a sequencing by ligation approach employing a two-base-encoded probe [26], i.e. each fluorometric signal represents a dinucleotide. After cluster generation or bead deposition onto a slide, the two-base-encoded probe is added to the DNA library (Figure 2.2), followed by degenerate or universal bases. The probe is ligated onto an adapter complementary anchor and then imaged. Next, some degenerate bases and the fluorophore are cleaved from the probe, leaving a 5' phosphate. The process is repeated several times until the entire strand is elongated. At this point, all anchors and probes are removed and many other rounds are performed to ensure every base in the template is sequenced. Since each signal represents a dinucleotide, the output is not directly associated with a single nucleotide. To identify which of the 16 possible dinucleotide combinations is the correct one, four fluorescent signals are employed, each one representing a subset of four dinucleotide combinations. This technique allows decoding the information during the data analysis and leads to the term color-space data, in opposition with classic base-space data, to identify the SOLiD output.

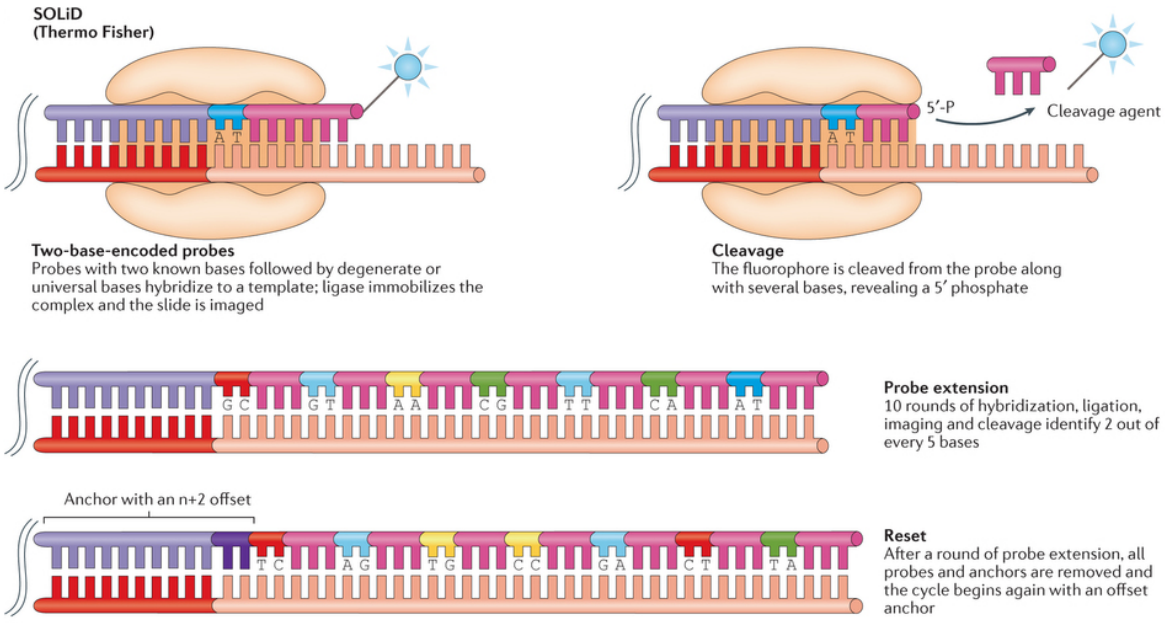

Fig. 2.2 *SOLiD sequencing.* Main steps involved in the SOLiD sequencing process. Colored elements: probe (dark blue), degenerate bases (pink), anchor (light purple), adapter (red). Image taken from [23].

Illumina

The Illumina system is a sequencing by synthesis platform using a cyclic reversible termination (CRT) approach. Similar to Sanger sequencing, a CRT approach prevents the elongation using terminator molecules in which the ribose 3'-OH group is blocked [27, 28]. After solid-state template amplification, a mixture of DNA polymerase, primers and modified nucleotides are added to the flow cell. The modified nucleotides are 3' blocked and labelled with a base-specific cleavable fluorophore. At each cycle, fragments in each cluster incorporate just one nucleotide while the unbound dNTPs are washed away (Figure 2.3). The slide is then imaged and the dNTPs are identified through total internal reflection fluorescence (TIRF) microscopy using either two or four laser channels. The fluorophores are then cleaved and removed and the 3'-OH group is regenerated with the reducing agent tris(2-carboxyethyl)phosphine (TCEP). Several cycles of nucleotide addition, strand elongation and cleavage are performed until the end of the sequencing process.

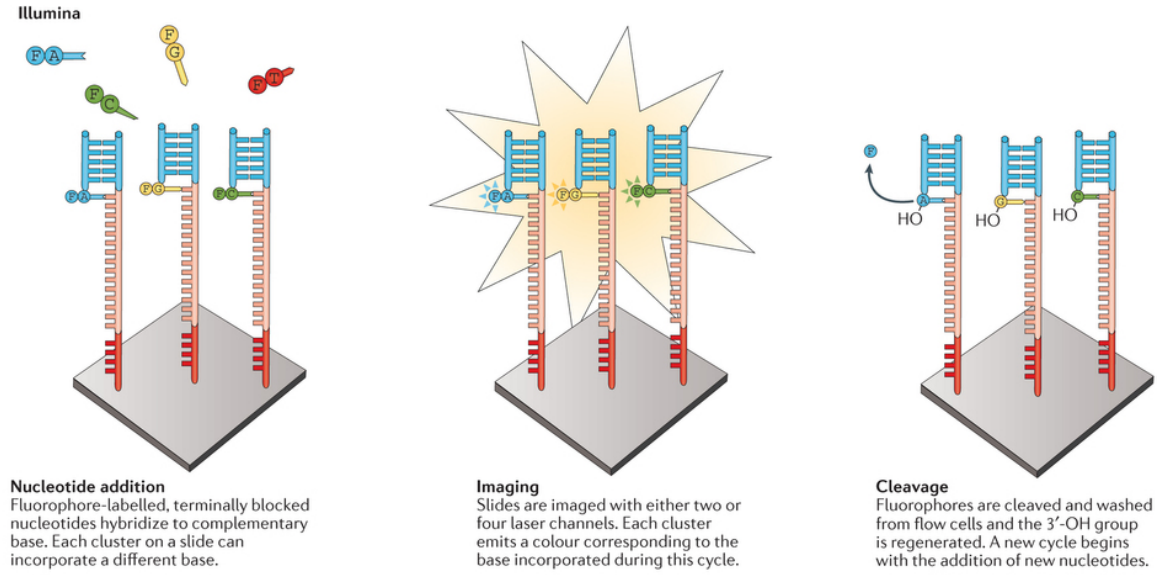

Fig. 2.3 *Illumina sequencing.* Main steps involved in the Illumina sequencing process. Image taken from [23].

Ion Torrent

Ion Torrent is a sequencing by synthesis platform employing a bead-based template enrichment and a single-nucleotide addition (SNA) approach. Unlike CRT, the SNA approach requires that each of the four dNTPs is added iteratively during the sequencing reaction. As a consequence, the SNA approach does not require the nucleotides to be 3' blocked, since the elongation is prevented by the absence of the next dNTP in the sequencing reaction. In the particular case of homopolymer regions, the nucleotides identification relies on the detection of a proportional increase in the signal due to the incorporation of multiple dNTPs. Ion Torrent is the first NGS system without optical sensing [29], exploiting the detection of H^+ ions that are released during each dNTP incorporation. As each dNTP is incorporated, the H^+ ion release results in a variation of 0.02 unit in pH, which is detected by an integrated complementary metal-oxide semiconductor (CMOS) and an ion-sensitive field-effect transistor (ISFET) device (Figure 2.4). However, the pH variation detected by the sensors is imperfectly proportional to the number of dNTPs detected, resulting in limited accuracy in measuring homopolymer regions.

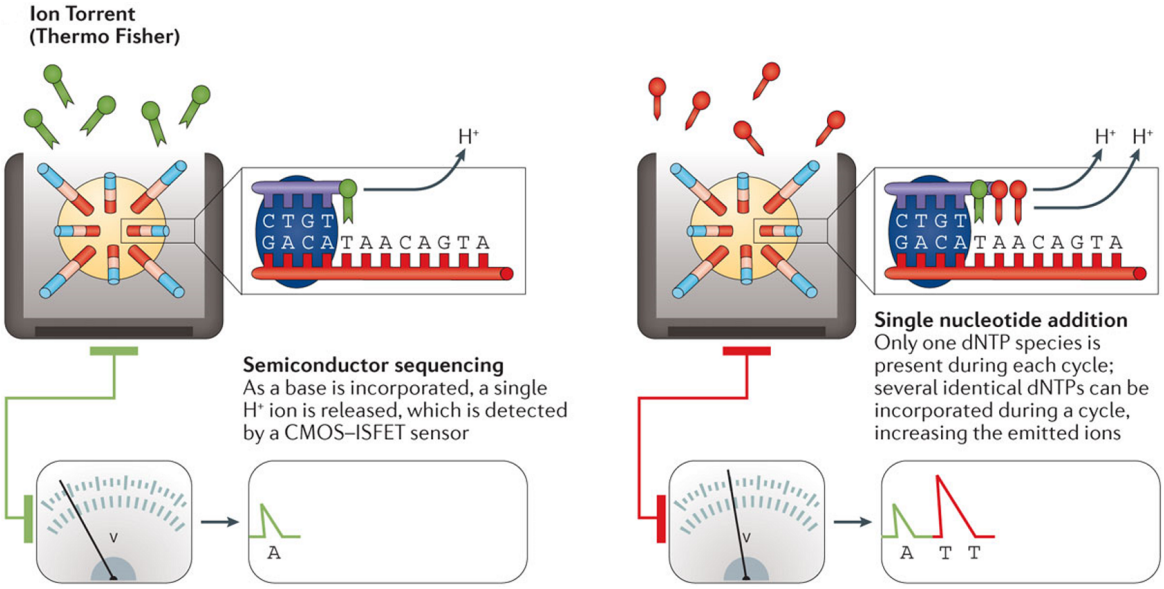

Fig. 2.4 *Ion Torrent sequencing.* Main steps involved in the Ion Torrent sequencing process. Image taken from [23].

454

454 pyrosequencing [30] is the first NGS instrument ever developed. It belongs to the sequencing by synthesis class and exploits a single-nucleotide addition approach. After bead-based template amplification, beads are distributed on a microplate along with primers and beads containing an enzyme cocktail. A single nucleotide species is then added to the plate and DNA polymerase synthesizes the complementary strand incorporating the available dNTPs. A pyrophosphate molecule is the by-product of this reaction and along with ATP sulfurylase transforms adenosine 5' phosphosulfate (APS) into ATP. ATP is involved in the conversion of luciferin to oxyluciferin by luciferase, resulting in a bioluminescence signal. Finally, incorporated bases are degraded using apyrase and the next dNTP is added to the wells. The light emissions due to the incorporation of one or more identical dNTPs is then detected by a charge-coupled device (CCD) camera (Figure 2.5).

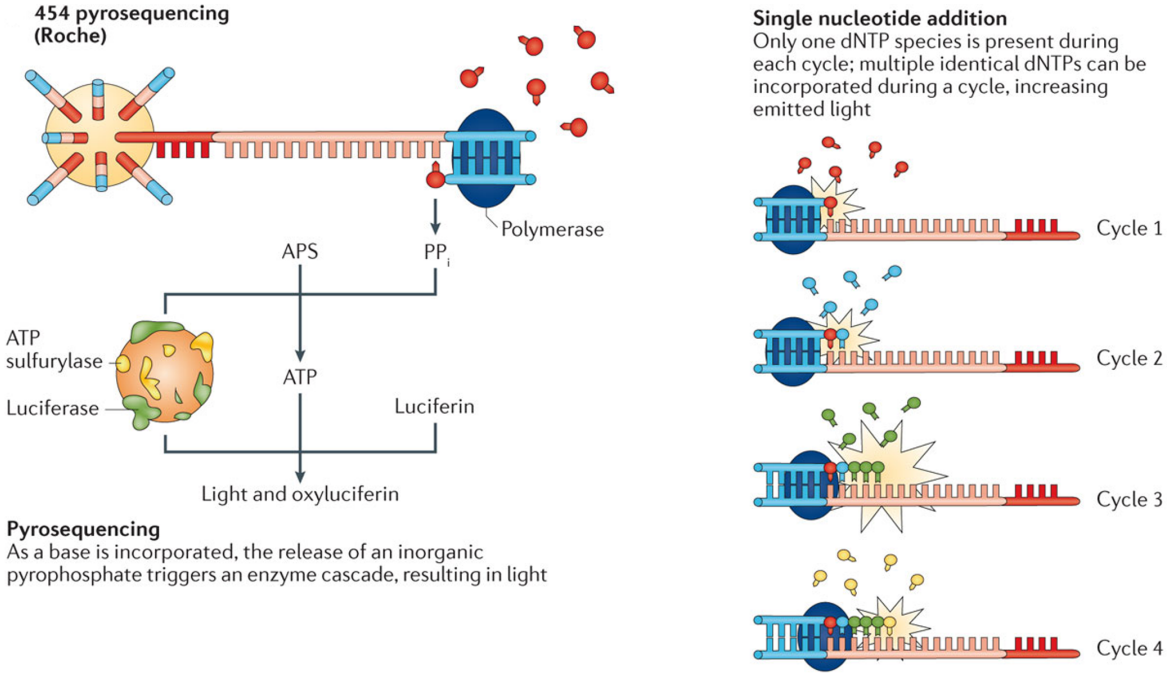

Fig. 2.5 *454 sequencing.* Main steps involved in the 454 pyrosequencing process. Image taken from [23].

PacBio

PacBio is currently the most widely used long-read platform employing a SMRT approach [31]. The process starts with the ligation of the DNA templates to hairpin adapters at each end, resulting in the 'SMRTbell', a circular DNA molecule containing the double-stranded DNA template in the middle and constant single-stranded DNA (ssDNA) regions at each end. A size selection protocol is then performed, followed by the attachment of primers and DNA polymerase to the ssDNA regions of the SMRTbell. The library is added to a particular flow cell with several thousands of individual picolitre wells with transparent bottoms, called zero-mode waveguides (ZMW) [32]. Polymerase is fixed to the bottom on the well and allows the DNA strand to progress through the ZMW (Figure 2.6). A mixture of labelled dNTPs is added and the nucleotide incorporation per well is monitored with a laser and a camera. The sensors allow detecting the color and the duration of the emitted light as the nucleotide momentarily pauses during the activity of the polymerase at the bottom of the ZMW. Before the next labelled nucleotide is incorporated, the polymerase cleaves the fluorophore during strand elongation, allowing it to diffuse away from the sensor area. In addition, the circular shape of the SMRTbell allows each template to be sequenced many times since the polymerase would repeat the traverse through the circular molecule. These multiple passes are exploited to create a consensus read of insert, called circular consensus sequence (CCS).

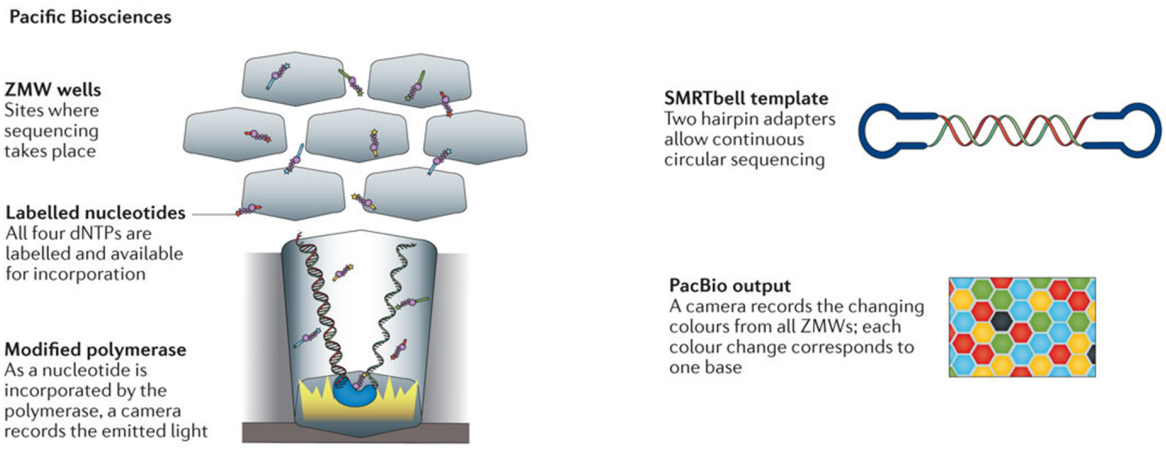

Fig. 2.6 *PacBio sequencing.* Main steps involved in the PacBio process. Image adapted from [23].

Oxford Nanopore

Oxford Nanopore platform is different from any other sequencing system since it does not monitor incorporations or hybridizations of nucleotides guided by a template DNA strand. A nanopore sequencer directly identifies the DNA sequence of a native ssDNA molecule passing it through a protein pore as current is passed through the pore [33]. In details, while a motor protein moves the DNA into the pore, a voltage change occurs modulating the current passing through the pore (Figure 2.7). Various parameters, including the duration and magnitude of the current shift, define the so-called squiggle space. Through the analysis of the squiggle space data it is possible to identify the particular DNA sequence passing through the pore. In the library preparation process, DNA is fragmented to 8–10 kb and two different adapters, a leader and a hairpin, are ligated to either end of the fragmented dsDNA. The leader adapter consists of a double-stranded fragment containing a tether sequence to help direct the DNA to the membrane surface and a sequence to direct the DNA into the pore. The harping links the two DNA strands, allowing for both the full forward and full reverse strand of a double-stranded DNA molecule to be sequenced and associated.

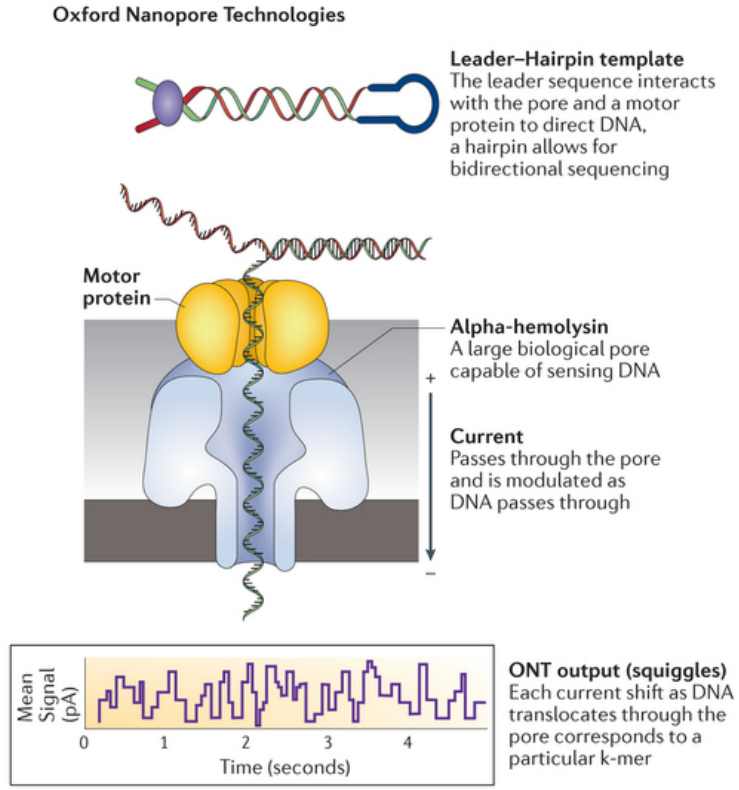

Fig. 2.7 *Oxford Nanopore sequencing.* Main steps involved in the Oxford Nanopore process. Image taken from [23].

Platforms comparison

As summarized in Table 2.1, each sequencing platform achieves different throughput, cost and error profile. Even if the SOLiD platform shows one of the highest accuracies, the short read length and some known substitution errors and AT/GC-rich under representation biases [35, 36] limit its popularity. Similarly to SOLiD, the Illumina platform shows some under-representation in AT/GC-rich regions [36–38] and a tendency towards substitution errors [39]. However, Illumina dominates the short-read sequencing industry, probably due to the maturity of its technology and the wide range of platform options. Both the Ion Torrent and the 454 platforms achieve higher read lengths compared to the other short read systems, but the employed SNA approach results in a higher indel error rate and a difficult handling of homopolymer regions [40].

In the context of long read platform, PacBio is the most widely used platform. One of the main limitation of this system in the high indel error rate (up to 15%) for a single-pass sequencing (i.e. the molecule is not sequenced multiple times) [41]. However, if the molecule is sequenced more times the accuracy quickly increases, reaching 99.999% for a molecule sequenced 10 times [42]. Compared to PacBio, Oxford Nanopore technologies are not very widespread, mainly due to several limitations on fragment size and accuracy on homopolymer regions [43]. On the other hand, the small dimension of the USB-based ONT MinION platform (MK1 dimension are 3 cm $\times$ 10 cm) could be a benefit in several scenarios, such as rapid clinical responses and hard-to-reach field locations.

Table 2.1 Summary of NGS platforms. Data taken from [23] and [34]; 2016 update.

Platform	Read length (bp)	Reads	Throughput (Gbp/run)	Runtime	Error profile	Instrument cost (US$)	Cost per Gb (US$)
SOLiD 5500xl	50(SE)	~1400M	160Gb	10d	$\leqslant$ 0.1%, AT bias	$251000	$70
SOLiD 5500xl	75(SE)	~1400M	240Gb	10d	$\leqslant$ 0.1%, AT bias	$251000	$70
Illumina MiSeq v2	150(PE)	24-30M	4.5-5.1Gb	24h	0.1%, substit.	$99000	$212
Illumina MiSeq v2	250(PE)	24-30M	7.5-8.5Gb	39h	0.1%, substit.	$99000	$142
Illumina MiSeq v2	75(PE)	44-50M	3.3-3.8Gb	21-56h	0.1%, substit.	$99000	$250
Illumina MiSeq v3	300(PE)	44-50M	13.2-15Gb	21-56h	0.1%, substit.	$99000	$110
Illumina NextSeq 500/550 High output	75(PE)	800M	50-60Gb	18h	<0.1%, substit.	$250	$41
Illumina NextSeq 500/550 High output	150(PE)	800M	100-120Gb	29h	<0.1%, substit.	$250	$33
Illumina HiSeq 2500 v2 Rapid run	100(PE)	600M	50-60Gb	27h	0.1%, substit.	$690	$52
Illumina HiSeq 2500 v2 Rapid run	150(PE)	600M	75-90Gb	40h	0.1%, substit.	$690	$45
Illumina HiSeq 2500 v2 Rapid run	250(PE)	600M	125-150Gb	60h	0.1%, substit.	$690	$40
Illumina HiSeq 2500 v3	50(PE)	3000M	135-150Gb	5.5d	0.1%, substit.	$690	$78
Illumina HiSeq 2500 v3	100(PE)	3000M	270-300Gb	11d	0.1%, substit.	$690	$45
Illumina HiSeq 2500 v4	50(PE)	4000M	180-200Gb	2.5d	0.1%, substit.	$690	$58
Illumina HiSeq 2500 v4	100(PE)	4000M	360-400Gb	5d	0.1%, substit.	$690	$45
Illumina HiSeq 2500 v4	125(PE)	4000M	450-500Gb	6d	0.1%, substit.	$690	$30
Illumina HiSeq 3000/4000	150(PE)	2500B	650-750Gb	1-3.5d	0.1%, substit.	$900	$22
Illumina HiSeq X 2 flow cells	150(PE)	6000B	1800Gb	3d	0.1%, substit.	$1000	$7
454 GS Junior+	400 avg. (SE, PE)	0.1M	70Mb	18h	1%, indel	$108000	$19500
454 GS FLX Titanium XL+	700 avg. (SE, PE)	1M	700Mb	23h	1%, indel	$45000	$9500
Ion PGM 314	200(SE)	0.4-0.5M	30-50Mb	23h	1%, indel	$49	$25-3500
Ion PGM 314	400(SE)	0.4-0.5M	60-100Mb	3.7h	1%, indel	$49	$25-3500
Ion PGM 316	200(SE)	2-3M	300-500Mb	3h	1%, indel	$49	$700-1000
Ion PGM 316	400(SE)	2-3M	600-1Gb	4.9h	1%, indel	$49	$700-1000
Ion PGM 318	200(SE)	4-5.5M	0.6-1Gb	4h	1%, indel	$49	$450-800
Ion PGM 318	400(SE)	4-5.5M	1-2Gb	7.3h	1%, indel	$49	$450-800
Ion Proton	200(SE)	80M	Up to 10Gb	4h	1%, indel	$224	$80
Ion S5 520	200(SE)	3-5M	0.6-1Gb	2.5h	1%, indel	$65	$2400
Ion S5 520	400(SE)	3-5M	1.2-2Gb	4h	1%, indel	$65	$1200
Ion S5 530	200(SE)	15-20M	3-4Gb	2.5h	1%, indel	$65	$950
Ion S5 530	400(SE)	15-20M	6-8Gb	4h	1%, indel	$65	$475
Ion S5 540	200(SE)	60-80M	10-15Gb	2.5h	1%, indel	$65	$300
Pacific BioSciences RS II	~20Kb	~550000	0.5-1Gb	2.5h	13% single pass, $\leqslant$1% circular consensus read, indel	$695	$1000
Oxford Nanopore MK 1 MinION	Up to 200Kb	>100000	Up to 1.5Gb	Up to 48h	~12%, indel	$1000	$750

bp, base pairs; d, days; Gb, gigabase pairs; h, hours; AT, adenine thymine; indel, insertions and deletions; Kb, kilobase pairs; M, million; Mb, megabase pairs; PE, paired-end; SE, single-end; substit, substitutions.

2.2.2 Typical RNA sequencing experiment

RNA-Seq data can be employed in several downstream analyses and the design of the RNA experiment should reflect the specific goal of the study. Indeed, the choice of the right RNA preparation techniques, library preparation procedures and sequencing technologies can considerably affect the reliability and accuracy of the final analyses. The choices of sequencing depth, read coverage, read length, coverage uniformity, number of replicates (technical / biological), library type, etc. allow to shape the experiment to better fit specific requirements. For example, long reads are very important for de novo transcript assembly, while they have a minor effect on gene quantification. Conversely, the presence of replicates is extremely important for differential expression analysis, whereas they are less useful in de novo transcript assembly. The best tradeoff between these options highly depends on the available budget and the kind of downstream analysis. By the way, the different procedures for RNA processing, library preparation and sequencing introduce several experimental biases that are impossible to eliminate. For this reason, it is important to know the different sources of bias and design the experiment to minimize the ones which could impact the final analysis.

2.2.3 RNA processing

The first step in a RNA-Seq experiment is isolating and purifying RNAs. First, the cells are disrupted and the RNAs are extracted from the total cell lysate. The cell disruption is achieved using chaotropic agents and detergents and, depending on the sample / experimental protocol, a mechanical disruption could also be performed. The extraction of RNA from the cell lysate is usually performed by organic solvents or solid-phase extraction onto silica. In order to assess how the isolation step achieves both a preservation of RNA integrity and a separation of RNA from cellular materials, the RNA quality and quantity is usually assessed at the end of these steps.

The RNA extracted using the above procedure consists of several types of RNAs: more than 80-90% of total RNA is ribosomal RNA (rRNA) while messenger RNA (mRNA) represents less than 5% of the total RNA. If the study is not focused on rRNA, then a rRNA removing step is mandatory otherwise the great majority of reads would come from ribosomal RNAs. Moreover, the increased sequencing depth allows the identification of low expressed transcripts and rare variants. There are two methods that are commonly used for rRNA depletion: selection of target RNAs via hybridization to oligo-dT and removal of not-target RNAs via hybridization [44]. They work following opposite approaches, since the first one allows directly selecting mRNA while the second one selectively removes ribosomal RNA.

The first method uses oligo-dT to recover poly-adenylated RNAs by duplexing with their poly-A tails. For instance, mature mRNAs are selected by this process, while immature mRNAs and non-polyadenylated ncRNAs are lost. Poly-A selection method requires a high proportion of mRNA with minimal degradation and usually produces a higher overall fraction of reads falling onto known exons. The second method uses oligos that are complementary to highly conserved rRNA sequences. Different from the first method, this technique allows preserving non-polyadenylated RNAs and it is widely used for prokaryotic organism.

The last step before library preparation is RNA fragmentation. The RNA sequences are usually longer than current read lengths, therefore a fragmentation step is useful to both improve the transcriptome coverage and reach the appropriate size for sequencing. The original protocols performed the fragmentation step after the cDNA conversion, which is still mandatory when the first strand synthesis is performed using oligo-dT or when the goal is to sequence full length RNA transcripts. However, in recent years fragmentation of RNA before cDNA conversion is becoming more popular. The most commonly used RNA fragmentation techniques are enzymatic, heat, metal ion and sonication. The fragment sizes is determined by the NGS platform and the specific sequencing application. For example, gene expression analysis does not require long fragments while for analysis of transcription start and stop sites or alternative splicing a large insert size would be advisable.

2.2.4 Library preparation

The RNAs obtained by the previous steps are still not suitable for the sequencing process. Since RNA is much more labile than DNA and RNases are harder to inactivate compared to DNases, RNA extraction is a critical procedure.

Current sequencing technologies work with DNA as input, so the RNA must be converted to double stranded complementary DNA (cDNA). The conversion of RNA into cDNA ensures the stability of the sample's information content. The current protocols use a particular type of polymerase known as reverse transcriptase (RT) to synthesize DNA from a RNA template. Reverse transcriptase requires a primer annealed to DNA or RNA to initiate the polymerization. There are several techniques for first-strand priming, the most used are *oligo-dT* and *random primers*. The first technique is one of the oldest first strand priming methods and uses oligo-dT to prime synthesis of the poly-A tail of mature RNA. Since the priming sequence is the same for all the mRNAs, they should be equally primed independently from their coding sequence. On the other hands, this technique works only for polyadenylated RNAs; for instance, not mature mRNAs or bacterial mRNAs are lost by this method. Obviously, this method requires to not fragment the RNA before the priming process. An important problem that affects this priming technique is the not uniformity

in sequence coverage [1]. Indeed, the RT is not a highly processive polymerase and so it could prematurely terminate resulting in a higher coverage in the 3'-end compared with the 5'-end. The second technique employs primers with random sequences and it is probably the most used first strand synthesis method. Compared with oligo-dT, this technique shows a reduced 3' coverage bias since the random primers could anneal throughout the length of the RNA. However, the priming process is not completely random and some nonuniformity in transcript coverage is still present [45]. Other advantages of this technique are the possibility to use RNA fragmentation and the recovering of non-polyadenylated RNAs.

After the first strand synthesis is completed, the second cDNA strand must be synthesized. As for the first strand synthesis, several options exist. One of the oldest and most used techniques employs E. coli DNA polymerase I for the synthesis, while E. coli RNase H is used to nick the RNA template and creates the RNA fragments that act as primers. The last step consists in the repairing of nicks and it is performed by T4 DNA ligase. This second strand synthesis procedure is highly efficient and well optimized, but has the major drawback of losing the RNA 5' end. This disadvantage was tackled by some new techniques, which pre-ligates an adapter to the 5' end of the RNA template and uses oligos that are complementary to this adapter to prime the second strand synthesis.

Once the cDNAs are ready, the next step in library preparation consists in adding adapter sequences at the ends of the fragments. Adapters are required by the different sequencing platforms both for clonal amplification and for priming the sequencing reaction. Even if they have the same function, the adapter composition is specific for the particular sequencing platform. Moreover, adapters could contain several optional elements employed by particular techniques (e.g. multiplexing, paired-end sequencing). For instance, adding to each library a specific tag (called index or barcode) in the adapter sequence allows the identification of which sequence comes from which library. This procedure allows to pool different libraries in a single sequencing reaction, saving both time and money. The process of pooling libraries from different experiments is known as multiplexing. Another useful procedure involves adapters containing sequencing priming sites for the opposite sides of the fragment. These adapter elements allow sequencing of both ends of the fragment (paired-end sequencing), resulting in a higher coverage. The additional information coming from having both ends of the fragments could be used to increase the accuracy in isoform detection [46] and in the mapping/assembly process [47].

Additional information about RNA strandedness could be collected by using so called *strand-specific protocols* [48], i.e. protocols which allow creating libraries that retain the strand orientation of the original RNA. Several strand specific protocols exist [49], falling into two main classes. The first class works marking one strand, either through a bisulfite

treatment on the RNA or during the synthesis of the second strand, and performing then a degradation of the unmarked strand. The second class of methods marks the 5' end and the 3' end of the original mRNA with different adapters, employing a known orientation pattern in the attaching process. One of the most used protocols belongs to the first class and is based on the idea of performing the cDNA conversion and removing one of the two strands selectively, by using dUTP for the synthesis of the second strand. The knowledge of which strand a read comes from may be exploited by many applications such as the identification of antisense transcripts, the determination of expression levels of coding/noncoding overlapping transcripts and the determination of the transcribed strand of noncoding RNAs.

The final step in library preparation entails a number of PCR cycles to enrich for product that has adapters ligated to both ends. However, amplification is the source of several biases, as the well documented relation between GC content and PCR amplification efficiency [37, 50]. For this reason, it is advisable to minimize the amplification steps, even if it is challenging for samples with low input.

2.2.5 Sequencing

Sequencing is the last step in the data production process. There are several sequencing platforms available, each one using different proprietary technologies and chemistries. The current leading platform is Illumina, followed by IonTorrent and PacBio. Legacy platforms such as SOLiD and 454 are still used, even if they have been employed less frequently in recent years. Each sequencing platform has unique strengths and weaknesses, so the choice of platform depends on the goal of the study. For instance, a transcriptome assembly study could benefit from PacBio long reads, while for a differential expression study the Illumina high sequencing depth could be the best choice.

2.3 RNA sequencing analysis pipeline

The output of a RNA-Seq experiment, as any other NGS experiment, consists of one or more read files. In order to achieve useful information from the reads, the analysis of RNA sequencing data requires the design of a complex analysis pipeline (Figure 2.8). The first part

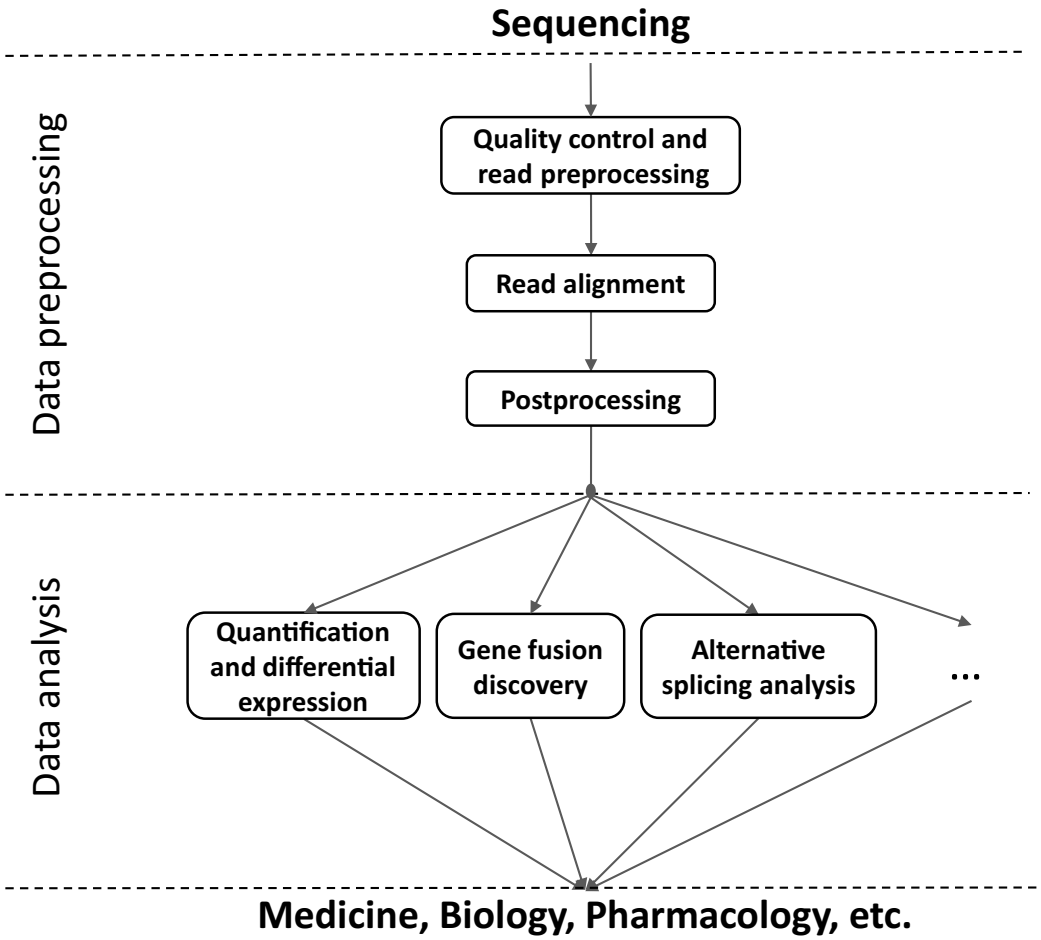

Fig. 2.8 *RNA-Seq analysis pipeline.* Flowchart of a typical RNA-Seq analysis pipeline. The first part of the pipeline (*Data preprocessing*) handles sequencing reads. The second part (*Data analysis*), exploits the processed RNA-Seq data to perform a variety of downstream analyses.

of the pipeline (*Data preprocessing*) analyzes the available reads, with the goal of accurate identification of the relation between the read sequences and the sequenced transcripts. Once the raw RNA-Seq data are processed, the second part of the pipeline (*Data analysis*) exploits the collected information to perform a wide range of downstream analyses, depending on the particular application. A more detailed description about data preprocessing and data analysis stages is provided in the next sections.

2.3.1 Data preprocessing

Once the reads are produced by the sequencing platform, many processing steps are performed in order to exploit the information contained in the reads sequences. Except for some conversion format procedures, the first task in the read analysis is a quality control step to assess the overall outcome of the sequencing process. This step allows the identification of possible issues and the application of a specific set of corrections. Once the quality of the reads is assessed, the next step involves the identification of which transcripts are present in the sequenced sample. This goal is achieved by aligning the available reads against a reference sequence such as a genome or a transcriptome. A brief description of the steps involved in the data preprocessing stage is presented below.

Quality control and reads preprocessing

Reads coming from sequencing output are usually analyzed in order to assess the quality of the dataset and identify potential sequencing errors, contamination and PCR artifacts. The quality control step involves the analysis of GC content, sequence quality (bases quality values), overrepresented k-mers, duplicated reads and adapters presence. Unfortunately, the evaluation of the previous elements is not trivial due to the dependency of many of these metrics on the particular experiment and organism. For example, the threshold for GC content level should be set depending on the organism, while the rate of duplicated reads would be different in experiment involving low or high input quantities. Once the correct scenario for the particular dataset is defined, there are many tools to perform reads quality control, as the popular software FASTQC [51].

When one or more of the previous analyses highlight some issues, several procedures could be adopted depending on the detected problem. For example, once an adapter sequence is detected at the end of a read or poor quality bases are identified, a common action consists in trimming this part of the read. On the other hand, in the case of contamination or a low quality read the usual procedure consists in removing the entire read from the dataset. Software such FASTX-Toolkit [52], NGS QC Toolkit [53], CutAdapt [54] and Trimmomatic [55] can be used to perform trimming and filtering. However, the definition of a robust procedure for read trimming and deleting is challenging and it is often specific for the particular dataset and experiment. In addition, the chosen tradeoff between overall quality and number of preserved reads could have a major role in the accuracy of many downstream analyses.

Read alignment

After quality control, the next step of read processing is the *read alignment*. The term *read alignment* or *read mapping* refers to the process of finding the read locations in a reference sequence such as a genome.

Read alignment has a pivotal role in the RNA-Seq analysis because it represents the step which extracts the information produced by sequencing and finally makes the reads usable and informative. Therefore, alignment accuracy would heavily affect the reliability of many downstream analyses, making this process the most critical step in the data preprocessing part of the RNA-Seq analysis pipeline.

The read alignment process is particularly challenging both in terms of efficiency and accuracy. The sizes of the reference genomes (from hundreds of thousands up to billions of bases) and the large amount of reads (tens or hundreds of millions) make read mapping a computationally intensive task. Typically, finding the correct positions of available reads in the reference genome requires tens of GB of RAM and many CPU hours on a multicore machine. What makes alignment even harder is the presence of sequencing errors, low complexity sequences, polymorphisms, insertions and deletions which require inexact matches between the read and the reference sequence and smart alignment strategies to avoid repetitive regions. In addition, the main challenge is identifying splice junctions correctly, since the reads come from transcriptome but the alignment reference sequence is the genome. For this reason, the alignment process should properly handle intron sized gaps, intron signals, incomplete annotation, alternative splicing and pathological splicing events. The alignment methods able to perform *spliced alignment* are called *splice aware* tools, as opposed to classic DNA aligners identified as *splice unaware* tools. Software such as STAR [56] and TopHat2 [57] are examples of widely used splice aware aligners.

Even though a genome is employed as reference sequence is the great majority of RNA-Seq studies, few alternatives exist. First, reads can be mapped to the transcriptome using a splice unaware tool such as BWA [58] and Bowtie2 [59], with the main advantage of requiring a low computational effort. However, aligning reads against a transcriptome has the main limitation to preclude the discovery of new and unannotated transcripts and its accuracy is heavily affected by the reliability of the available annotation. The second option consists in de novo assembly of the reads into a transcriptome using tools such as SOAPdenovo-Trans [60], Trans-ABySS [61] and Trinity [62] and then performing an unspliced alignment against the inferred transcriptome. Although assembly methods allow de-novo transcripts discovery, they are more computationally intensive than read alignment, resulting in efficient applications only on small genomes (e.g. archaea genomes).

The pivotal importance of the alignment step and the consequent big effort of the bioinformatics community resulted in the availability of a large number of splice aware methods. Since 2009, more than 20 splice aware software were developed by the research community, making the choice of the right aligner a non-trivial step.

Postprocessing

Depending on the particular downstream analysis, a read alignment postprocessing step may be necessary. Typical postprocessing actions involve file format conversion, alignment sorting, alignment filtering and generation of alignment statistics. Alignment statistics usually report the number of aligned/unaligned reads, reads mapped to multiple locations, potential PCR duplicates, etc. These statistics are useful to identify potential issues (e.g. very low percentage of mapped reads) rather that to assess the accuracy of the alignment process (e.g. a high percentage of mapped reads does not necessarily mean that the reads are mapped correctly). Software such as SAMTOOLS [63] and Picard [64] are commonly used for read alignment postprocessing.

2.3.2 Data analysis

Once the reads are aligned to a reference sequence, and an optional preprocessing step is performed, the second part of the RNA-Seq analysis pipeline exploits this information through a wide range of analyses. Preprocessed RNA-Seq data could be employed in several applications to both identifying transcript characteristics and quantifying transcript abundances.

The most common application involves the quantification of gene/transcript expression levels among samples followed by differential expression analyses. A brief description of this analysis is presented below. However, RNA-Seq data are suitable for many other analyses such as alternative splicing analysis, gene fusion discovery, allele specific expression, non-coding RNA discovery, single nucleotide variants identification, etc. making RNA-Seq one of the most powerful and flexible techniques for transcriptomic studies.

Quantification and differential expression analyses

Estimation of gene/transcript expression level is the most common application of RNA-Seq studies. The idea behind expression level quantification is that the number of reads that map to each transcribed sequence is a good proxy of its expression level. Quantification could be performed at the transcript, gene or exon level, depending on which coding units is employed to summarize the raw counts of mapped reads. The number of reads overlapping the chosen

feature is used as a digital measure of its expression level and is called *counts*. Provided with the read alignment file and an annotation file, usually in GTF, GFF or BED format, tools such as featureCounts [65], HTSeq-count [66] and BedTools [67] are able to compute the counts of each feature included in the input annotation. As any other downstream analysis, the reliability of expression level quantification heavily relies on the accuracy of the read alignment step. In addition, it was demonstrated that even the choice of annotation has a strong impact on the quantification results [68].

Unfortunately, raw read counts are not suitable for comparing expression levels within and between samples since these values are affected by several biases. In order to remove or at least mitigate these effects, several normalization techniques were developed to both handle within-sample and between-sample biases. Within-sample normalization allows an unbiased comparison of expression levels within the sample, handling effects such as the different feature lengths. Instead, between-sample normalizations like TPM (transcripts per million) [69], TMM [70], DESeq[71] and UpperQuantile [6] allow a fair comparison of expression levels between the samples, correcting effects such as differences in the library sizes. In the context of DE analysis, between-sample differences have to be corrected while some within-sample biases may cancel out when comparing samples. Finally, the widely used RPKM (reads per kilobase of exon model per million reads) [2] and FPKM (fragments per kilobase of exon model per million mapped reads) [72] are examples of approaches that perform both types of normalizations. Sophisticated tools such as Cufflinks [72], RSEM [73] and eXpress [74] are able to quantify at the gene/transcript level and at the same time perform a normalization procedure.

Finally, differential expression analysis is performed employing a test statistics which selects the genes/transcripts for which expression levels (i.e. normalized counts) are significantly different between the compared conditions. Several tools have been developed for this task, both using parametric and non-parametric approaches. Methods such as edgeR [75] and DESeq2 [76] employ a negative binomial model to describe the data, while EBSeq [77] and baySeq [78] exploit a Bayesian approach, also based on the negative binomial model. Other approaches exploit the sample variance to create discrete expression level distributions and analyze them using a regular linear model [79]. On the other hand, non-parametric methods such as NOISeq [80] or SAMseq [81] estimate the null distribution from the counts alone with minimal assumptions about the data. Several independent studies have revealed that no single method clearly outperforms the others and performance is highly dependent of the particular dataset [82–84].

Chapter 3

Read alignment

The output of the sequencing process consists of millions of reads, short sequences representing fragments of the transcripts in the original sample. As seen in the previous chapter, the alignment of millions of reads against a reference genome is a common step in the great majority of RNA-Seq studies, whereas only few studies rely on read assembly. The alignment process allows the identification of which fragments belong to each transcript, mapping each read to the region of the genome from which it originated. Unfortunately, performing an accurate and efficient alignment of several millions of NGS reads against a reference genome is challenging.

In this chapter, different aspects of the read alignment problem are explained and analyzed. Since the mapping of RNA reads shares many challenges with the alignment of DNA reads, the most important aspects and issues in mapping NGS short sequences are described together with the peculiarities of RNA-Seq data. Moreover, a brief analysis of de-novo transcriptome assembly is discussed, since this method represents an alternative option to read alignment in some contexts. The state of the art techniques adopted in the sequence alignment problem and implemented in the available RNA-Seq aligner tools are reviewed, based on a thorough literature search that involves more than 2000 peer reviewed publications in the context of RNA-Seq. For each tool, its adopted mapping strategies, specific characteristics and implementation choices are briefly described.

3.1 The read alignment problem

In the context of NGS reads, the term *read alignment* or *read mapping* refers to the process of finding the position of a read in a reference sequence. In this thesis, and more generally in the context of RNA-Seq, the term *read alignment* is a synonym for *read mapping*. However, the two terms have slightly different meanings: a *mapping* is the region where a read sequence is placed and it is regarded to be correct if it overlaps the true region, whereas an *alignment* is the detailed placement of each base in a read and it is regarded to be correct only if each base is placed correctly. In other words, the alignment process consists of more than searching from where the read likely came in the reference sequence, it requires the identification of a base to base correspondence. The idea behind the read alignment is that the read originated from the same sequence (or a similar one) used as reference sequence in the alignment process. Depending on the experiment and the kind of analysis, the reference sequence could be a transcriptome, a genome, a set of contigs, a chromosome, etc. The read alignment process represents a solution to the problem of identifying what is the composition of the sequenced sample. Indeed, the output of the sequencing process are simply nucleotide sequences with no additional information about which element in the sample or which part of it have generated them. The mapping of the reads against the right reference sequence allows the identification of this missing information and makes the NGS data usable.

3.1.1 From DNA to RNA read alignment

RNA-Seq was first introduced in 2008; consequently, the problem of aligning RNA-Seq reads was addressed only in recent years. However, despite the pivotal role in the analysis of RNA-Seq data, the alignment of sequencing reads is a very important task in several sequencing contexts. The alignment of genomic sequences had a pivotal role even in pre-NGS era, where sequence alignment tools such as BLAT [85] an SSAHA [86] made some of the most important genomic studies possible, including the Human Genome Project. In the context of NGS read alignment, the first milestone was probably the development of the ELAND aligner in 2006 [87]. ELAND was provided to costumers buying the Solexa Genome Analyzer, allowing end users to perform alignment procedures since then performed only by large genome sequencing centers. After 2006, a large number of read aligners were developed resulting in more than 20 new tools in the next 3 years, including popular tools like MAQ [88], SOAP [89], RazerS [90], mrFAST [91], SHRiMP [92] and BFAST [93]. The second milestone could be identified by the development of two famous aligner tools in 2009: BWA [58] and Bowtie [94]. Both tools exploited the Burrows-Wheeler transform, introducing some innovations and smart algorithms to obtain fast and accurate read alignment.

In the same year, the standard sequence alignment output file format (SAM file format) was introduced together with a tool to handle it (SAMtools)[63]. At the end of 2009, the progress achieved in the field of NGS read alignment allowed the mapping of several millions of short DNA reads containing few mismatches and small gaps in few hours using a desktop computer. The main challenges at the time were handling longer reads (> 50-75bp) and allowing more than 2-3 mismatches and indels in the read alignment.

Between 2008 and 2009, the advent of RNA-Seq data introduced a new big challenge in the context of NGS read alignment: *spliced alignments*. Spliced alignment refers to the need of identify a long gap in the read alignment against the genome due to the presence of introns in the reference sequence. The transcriptome of eukaryotic organisms is affected by splicing, which removes intron sequences and joins exons together. Since RNA-Seq reads come from transcriptome, some reads could be sequenced from an exon-exon junction region of the transcript, corresponding to no contiguous sequence in the genome. The alignment of such reads against the genome requires handling of intron size gap properly, identifying the exon-exon junction and splitting the read across on the two sides of the intron. Read mapping tools able to properly manage spliced alignment are called *splice aware* aligners, in opposition with classic DNA-Seq aligner identified as *splice unaware* tools. The first read aligner developed to properly handle spliced alignments was QPalma [95] in 2008. QPalma employed a machine learning approach, in which data from known splice junctions are used to train the software on identifying exon-exon junctions. However, the tool had the main limitation to require an accurate annotation as input in order to perform spliced alignment, making it impossible to find novel junctions. In 2009, the first method to perform spliced alignments with no input annotation was developed and implemented by the TopHat [96] aligner. TopHat exploited the Bowtie alignment engine to perform unspliced alignment and identify candidate exon regions. Reads flanking potential exon-exon sites within neighboring regions are the clustered together to form potential splice junctions' sequence. Finally, Bowtie is employed to align the remaining reads against the inferred splice junction sequences. After TopHat, a lot of effort was put by the bioinformatic research community on the spliced alignment problem, resulting in more than 20 tools published from 2008 to 2016.

3.1.2 RNA sequencing read alignment challenges

The alignment of reads against a genome is a special case of the string matching problem, which requires modeling and incorporating the characteristic of DNA/RNA sequences and sequencing technologies in order to achieve an accurate result. In this section, the alignment problem is analyzed describing the most important aspects and the related challenges.

Mapping criteria

Since the true position of a read in the reference sequence is unknown, an important aspect of the problem is the definition of mapping criteria. Ideally, the alignment process should identify the most probable mapping position for a read given the available information. The definition of 'most probable mapping position' is what defines the alignment criteria of each method. Since the setting of a probabilistic framework for read mapping is complex, in practice there are two main approaches: i) *alignment scoring* and ii) *errors threshold*.

The alignment scoring method uses a scoring function to assign a value to each alignment. There are several scoring functions, usually inspired by the Smith-Waterman scoring system [97], that assign different penalties to matches, mismatches, gap open, gap extension and gap end. Depending on the alignment strategies, the alignment of a read is chosen among the ones that passes a threshold and has the highest score. These methods allow modeling of different aspects of the read alignment process, combining and weighting them in a deterministic way. On the other hand, defining a smart scoring function and assigning reasonable penalties values is not trivial.

Error threshold methods are instead based on the definition of a maximum number of allowed errors in the alignment process. Depending on the implementation, the term 'errors' could refer to mismatches, insertions and deletions between the read and the reference genome. More complex approaches are able to apply different thresholds depending on the position in the read. This method is conceptually simpler compared to alignment scoring, since defining a maximum number of allowed errors is easier than defining a penalty score system. However, this simplicity is paid for in terms of lack of flexibility and difficulties in handling complex situations.

Multireads

The term *multiread* (or *multimapper*) refers to read than cannot be mapped unambiguously, since the adopted mapping criteria identifies more than one possible location originating the fragment in the genome. In practice, this is due to repetitive sequences in the genome or a high number of mismatches and gaps. In order to provide one candidate position for each read, some alignment methods try to prioritize multimappers using some criteria while other methods simply discard or report all the multireads. Due to the ambiguity associated with these read mappings, the great majority of downstream analyses simply remove them, while only few post-alignment programs are able to handle them.

Base quality scores

Base quality scores are measures of correctness of each base in the read, provided as output by some sequencing technologies. The base quality score employs the phred scheme, where the score Q is equal to $-10log_{10}(e)$ and e is the probability that the base is wrong. Quality scores provide a useful information about the reliability of each base in the read and for this reason they are often employed in the alignment process. Some tools exploit the quality score to perform a pre-processing step, removing or trimming low quality reads while other aligners use the quality score during the mapping process. For example, the quality value could be employed in the computation of an alignment score or could be used to identify a possible mismatch.

Polymorphisms and sequencing errors

Alignment methods should allow inexact matching between the read and the reference sequence to account for the presence of polymorphisms and sequencing errors. Mismatches make the alignment problem harder, since inexact matching algorithms have a higher computational complexity compared to exact alignment procedure. In order to make easier the alignment process, some methods allow providing a set of known single nucleotide polymorphisms (SNPs) as input. Other tools try to identify SNPs by analyzing overlapping reads, exploiting the idea that the same polymorphism would be shared by all the reads mapping in the same region. At the same time, this procedure allows the identification of sequencing errors by looking for genome mismatches that are not shared by overlapping reads. A different approach consists in employing Smith-Waterman-like algorithms, usually applying a heuristically bounded version or reducing the size of the problem to a portion of the read. Irrespective of the chosen solution, many state of the art methods try to limit the number of available mismatches (usually up to 2-3 mismatches per read) in order to achieve a reasonable execution time.

Insertion and deletions

The presence of insertion and deletions (indels) necessitates inserting or deleting nucleotides during the mapping process. Insertions and deletions are commonly identified by the term *gaps*, especially in the computer science field. Compared to mismatches, gaps represent a bigger challenge in the alignment process. Similar to mismatches, gaps are usually handled by Smith-Waterman-like algorithms and the number, size and location of allowed gaps are limited.

Paired-end reads

Paired-end reads result from sequencing both ends of a DNA fragment. The mapping of paired-end reads could improve the accuracy of the alignment, exploiting the estimated distance between the two ends to better identify the correct location in the genome. The extra information provided by paired-end reads could be exploited in different mapping strategies. Some methods align independently the two mates, performing an extra step at the end of the alignment to recover the distance information and exploiting it to rearrange some alignment locations. Other methods try to align the two reads at the same time, limiting the alignment positions to a defined window. Exploiting the distance information results particularly useful for multimapper or complex reads (i.e. high rate of mismatches and gaps), employing the unambiguously aligned mate as an anchor and trying to identify the correct location in the nearest genomic regions.

End-to-end and local alignment

The alignment of a read could be performed in two different ways, called *end-to-end alignment* and *local alignment*. End-to-end alignment refers to the mapping of the entire read sequence against the reference genome, while local alignment refers to aligning only a portion of the read, usually avoiding to align few bases at the ends of the read. Since first generation sequencing reads were very short (< 50bp), the first mapping methods performed only end-to-end alignment. Indeed, a full length alignment reduced repetitive region issues and did not result in too high of a computational requirement. However, the introduction of longer reads increased the required computational effort and the number of mismatches and gaps to handle. In particular, the longer reads produced by many sequencing technologies resulted in low quality ends which required a significant amount of time to be properly handled by the aligners. In this scenario, many tools changed their mapping policy from end-to-end to local alignment. Local alignment helps managing adapter sequences at the end of the reads, reducing the amount of effort in read preprocessing. Obviously, the unaligned portion of the read could result in a less informative data, depending on the downstream analysis. Nowadays, many state of the art aligners allow choosing both kinds of alignment or automatically choose the most appropriate one for each read.

Spliced alignment

Spliced alignment is the major challenge in RNA-Seq read alignment and the main difference between DNA and RNA mapping tools. RNA-Seq reads require to correctly align on the genome reads that come from exon-exon junctions in the transcriptome, introducing long

gaps in the alignment due to the presence of introns. In eukaryotic model organisms the intron length ranges from tens to tens of thousands nucleotides, while the usual read length is just 100 bp. The complexity of spliced alignment is further increased when the read spans more than one exon-exon junction or the overlap between the read and the exon is just few bases long. The huge challenge represented by spliced alignment is addressed by current state of the art methods using many different strategies, each one having peculiar strengths and weaknesses.

Annotation and *de-novo* splicing detection

Detection of spliced alignment could be performed in three ways: using a known set of splicing sites, detecting de-novo splicing sites and using and hybrid approach employing both techniques. The first class of methods use an input annotation, usually a set of exon coordinates, to assist the alignment process. A common approach consists of creating a set of candidate junctions, exploiting the information contained in the annotation, and applying a splice unaware aligner to these junction sequences. These methods are very fast but have the main limitation of heavily relying on the accuracy of the annotation and the impossibility to find novel junctions. These class of methods are often referred to as *transcriptome alignment methods*. The second class of methods try to infer the exon-exon junctions from the available reads, using different strategies to recover this information from the data. Compared to the previous class of methods, performing de-novo splicing detection results in a more complex task and longer execution time. However, these methods can identify un-annotated exon-exon junctions. The last class of methods tries to combine the strengths of the two previous approaches, being able to perform de-novo splicing detection and, if an annotation is available, integrate this extra information to assist the alignment process.

3.1.3 An alternative solution: de-novo transcriptome assembly

The main alternative option to aligning RNA-Seq reads against a reference sequence consists in performing a de-novo transcriptome assembly. Similar to DNA assembly, transcriptome assembly creates contiguous segments of sequence data (called *contigs*) exploiting the overlapping sequences between reads. State of the art methods for de-novo transcriptome assembly include well known tools such as SOAPdenovo-Trans, Trans-ABySS and Trinity. Assembly methods have the main advantage of not relying on the availability and accuracy of a reference genome. In addition, assembly methods are not affected by the multiread issue and allow de-novo transcript discovery. On the other hand, transcriptome assembly is more computationally intensive than read alignment, resulting in efficient applications only on

small genomes such as bacteria and archaea. Compared to alignment methods, assembly requires a higher sequencing depth in order to achieve an accurate assembly result, especially for the detection of low abundance transcripts. In addition, the use of assembly based analysis was demonstrated to underperform compared to alignment guided analysis [98]. Since the majority of studies involve human or model organisms for which a high quality genome is available, assembly methods are limited to organisms having small genomes or no reference genome.

3.2 Data structures and algorithms

The alignment of a read against a reference genome is a particular case of the more generic computer science problem of finding the position of a string (query) in a text (reference). For this reason, the read alignment methods exploit many concepts developed in computer science to obtain efficient and accurate alignment procedures.

Even in the simpler case of finding an exact match between the read and the reference genome, an efficient algorithm is required. A naive approach which consists of sliding the read along the reference genome would results in an unfeasible complexity of $O(NMm)$ when N is the size of the genome, M is the number of reads and m is the read length. Regarding the accuracy, the methods should be able to handle repetitions, mismatches, insertions, deletions and intronic gaps, which significantly increase the complexity of the alignment problem.

Therefore, all the methods rely on some sort of preprocessing of the input data, both on reads or reference genome. The first NGS read aligner methods performed a preprocessing on the reads [87, 88], while the most common approach in recent years consists of preprocessing the reference genome, building a so-called *genome index*. An index is a data structure which allows fast exact matching between the query and the reference sequence. Even though the creation of an index requires extra computation, it could be stored and reused for any other string matching problem on the same reference text. All the current and many past read alignment methods employ a genome index, though different methods use different data structures and implementations.

The next section describes the main index implementations and the algorithms employed for exact and inexact matching in the context of RNA-Seq read alignment and, more generally, NGS read alignment.

3.2.1 Hashing methods

Hashing methods build a hash table representation of the genome, where ideally each read sequence is a key and the associated table element is the list of its positions in the genome. Using this approach, the alignment process of error free reads would result in a table look-up for each read in the read set (Figure 3.1). However, this does not work in practice due to the current read lengths: for a read of length 100, the hash table would contain $4^{100} \approx 10^{60}$ elements. The adopted workaround consists in using k-mers as keys, employing key lengths significantly smaller than read lengths. In this way, the alignment process consists of selecting a k-mer for each read, called *seed*, and mapping it to the genome using the hash table.

Then, for each possible location, the procedure tries to extend the seed alignment to a full read alignment using a Smith-Waterman-like approach. The strategy of performing an exact

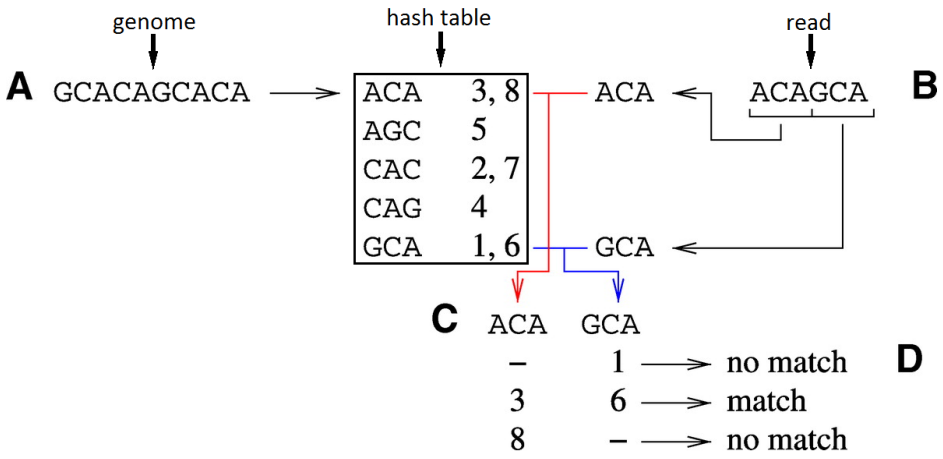

Fig. 3.1 *The hashing algorithm.* **(A)** The genome sequence is cut into overlapping 3-mers and the position of each 3-mer in the genome is stored in a hash table. **(B)** The read is cut into 3-mers and each read 3-mer is compared to genome 3-mers using a hashing procedure. **(C)** Positions for each read 3-mer are sorted and compared to the other 3-mers. **(D)** Compatible positions are kept. Figure adapted from [99].

.

alignment of a read portion (seed) and then extend it to a full alignment allowing mismatches and gaps is called *seed&entend*.

Using this approach, the size and the position of the seed in the reads are critical. A small seed requires a smaller hash table but would result in more genomic locations. On the other hand, using a larger seed would increase the probability of containing an error or a gap which does not allow mapping to the correct location. In addition, the location of the seed in the read could affect the alignment accuracy, since the error probability in not uniform along the read for some sequencing technologies (e.g. Illumina). In order to overcome some of the above limitations, many methods employ a multi seed approach. Instead of using a single seed, these methods extract several substrings from the read sequence and map them independently. The different seed alignments are then exploited to identify a final read alignment, employing the usual full alignment extension procedure. In addition, the use of multiple seeds permits finding reads with errors in the seed sequences, if there is at least one seed with no errors.

3.2.2 Suffix array and Burrows–Wheeler transform methods

This class of methods exploit some fast string searching characteristics from suffix trees [100] and suffix arrays [101]. A suffix tree for a string is a tree in which there is a one-to-one relation between the suffixes of the string and the paths from the root to the leaves. In the

context of read alignment, a suffix tree of the genome sequence could be used to identify an exact match of a read in $O(m)$, through a simple path from the tree's root to a leaf (Figure 3.2). Despite the great performance in exact alignment, a suffix tree for a large genome requires a significant amount of memory to be stored.

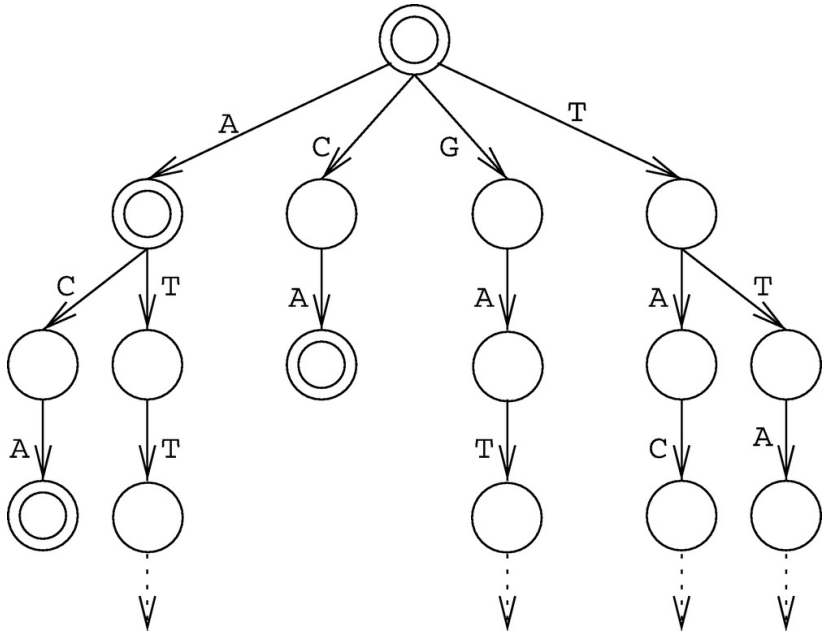

Fig. 3.2 *The suffix tree of the genome sequence "GATTACA".* Dotted arrows indicate that the tree continues there. Double circle indicate that a suffix ends there. Figure taken from [99].

Suffix arrays (SA) represent a solution to the memory issue, even though the alignment procedure is a little bit more difficult. A suffix array is defined as the set of suffixes of a string sorted lexicographically (Figure 3.3). In order to reduce the memory requirements, it is possible to store only the position of the beginning of the suffixes instead of storing all the suffixes, using then the genome sequence to recover the entire suffix. The suffix positions and the suffix sequences allow to traverse the suffix array in a way similar to the traversal method for suffix trees, enabling fast string matching.

Among the common applications of SA due to its fast matching properties, the most important is the one which involves the Burrows–Wheeler transform [102]. The Burrows–Wheeler transform (BWT) is a text transformation which is strongly related with the suffix array. The strong relation between the two elements are exploited by the FM-Index [103], which describes an efficient algorithm, called *backward search*, to perform an exact match on the BWT using some information derived from the SA. Even though the backward search algorithm is very efficient in retrieving exact matches, the extension to inexact matches is still an open problem. Methods employing BWT for entire read alignment usually adopt heuristics

genome	suffixes	sorted suffixes	positions	cylinder suffix array	B–W
GATTACA$	GATTACA$	$	7	$GATTACA	A
	ATTACA$	A$	6	A$GATTAC	C
	TTACA$	ACA$	4	ACA$GATT	T
	TACA$	ATTACA$	1	ATTACA$G	G
	ACA$	CA$	5	CA$GATTA	A
	CA$	GATTACA$	0	GATTACA$	$
	A$	TACA$	3	TACA$GAT	T
	$	TTACA$	2	TTACA$GA	A

Fig. 3.3 *Suffix array of the genome sequence "GATTACA".* Last column is the Burrows-Wheeler transform. The "$" character indicates the end of the sequence. Image adapted from [99].

or define ad hoc mapping strategies to overcome the inexact matching limitation. On the other hand, BWT based methods are useful to align a small part of a read in a seed&extend approach. Similar to a hash table, the SA or BWT can be used to align one or more read's seeds allowing no mismatches or gaps, performing then a full read alignment extension using a Smith-Waterman approach.

3.2.3 Implementation choices

Data structures and algorithms play a pivotal role in the success of a mapping tool but an equally important role is played by the implementation choices. For example, the read alignment process could be easily parallelized since each read is mapped independently. The great majority of tools exploit multi-threading, loading into a shared memory space the genome index and splitting the read set between the available threads. In the processing of each read, the use of Single Instruction Multiple Data (SIMD) instructions and bit-wise string comparison operations allow the tool to further increase the speed of the alignment process. Another important aspect in the software development is a smart use of the memory and its hierarchical organization. Avoiding frequent disk operations to retrieve reads or genome index as well as designing algorithms and data structure to exploit cache locality and fit entirely in RAM have a huge impact on the speed of alignment tools.

3.3 Available methods and state of the art

In order to identify the available methods for RNA-Seq read alignment, a literature search was performed on about 2000 peer reviewed publications involving RNA-Seq studies. The 2000 publications were randomly sampled by the available articles in the PubMed archive in the period from 2011 to 2016, looking for the keyword "RNA-Seq" in the article title or abstract (~7000 publications). In addition, the research was initially not limited to RNA-Seq read alignment methods but included also transcriptome assembly methods and splice unaware tools. The goal of including the two main alternatives to the RNA-Seq reads mapping was to have a clear and unbiased perspective of view of the currently employed methods.

The bibliographic research highlights many interesting trends about the choice of RNA-Seq alignment and assembly methods. The great majority of methods are used by only a few publication per year, while a small subset of methods shows a consistent high usage among the studied period. Based on the method usages per year, a subset of 14 tools was identified as representative of the most popular algorithms in the context of RNA-Seq reads (Figure 3.4).

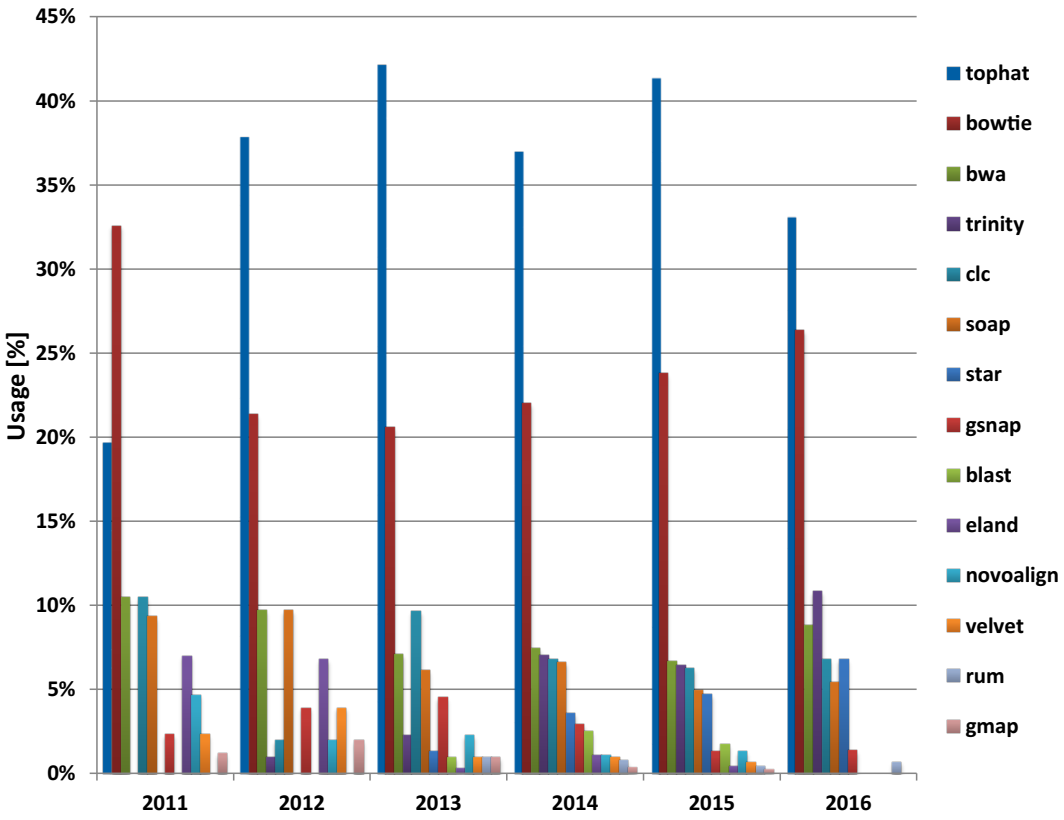

Fig. 3.4 *Comparative usage of the methods identified through the literature search, stratified by year.* The figure shows the 14 most employed tools among all the identified methods.

Focusing on alignment methods, even though TopHat2, STAR, GSNAP, CLC and Novoalign result the most employed RNA-Seq aligners, the literature search reveals a long list of RNA-Seq mapping tools. In order to identify the state of the art methods, the list was filtered by removing tools that are no longer maintained, requiring a last update after 2012/2013. The final list resulted in fourteen splice aware methods: CLC Genomic Workbench [104], ContextMap2 [105], CRAC [106], GSNAP [107], HISAT [108], HISAT2 [108], MapSplice2 [109], Novoalign [110], Olego [111], RUM [112], SOAPsplice [113], STAR [56], SUBREAD [114] and TopHat2 [57].

Not surprisingly, TopHat and TopHat2 are the most employed tools in this subset, being used by ~35% of the publications on average. Even though they are far from TopHat in terms of usage, the literature search identifies STAR, GSNAP, CLC and Novoalign as the most adopted alternatives for RNA-Seq read alignment. Interestingly, the second most adopted tool is Bowtie/Bowtie2 (~30%) followed by BWA (~8%). Both Bowtie and BWA are splice unaware tools, so they cannot be employed in every kind of RNA-Seq study. Other splice unaware aligners are on the list, highlighting the common trend of employing splice unaware tools in the context of RNA-Seq. This is a common error in the RNA-Seq research community, since splice unaware tools are not generally suitable for any kind of RNA-Seq study and their application should be limited to simple organisms or transcriptome alignment. In addition, the literature search revealed that a very small number of publications used de-novo reads assembly methods instead of read alignment methods.

In order to identify the alignment techniques and strategies adopted by state of the art read alignment methods, the fourteen splice aware tools resulting from the literature search were analyzed. In addition, 4 very popular splice unaware methods were considered since they play as alignment core engine in many splice aware tools and can be used as stand alone in studies involving organisms with no or negligible splicing. Both lists of methods are described in what follows.

3.3.1 Splice aware methods

The 14 splice aware methods result in a variety of alignment strategies and policies, which make it difficult to identify a common trend in the way the alignment problem is addressed. Indeed, each tool adopts specific solutions to handle mismatches, SNPs, insertions, deletions, spliced alignments and paired-end reads. The only common elements between the tools are the adoption of the SAM file format, the possibility of exploiting multi-threading parallelism and the use of genome indexing strategies to achieve faster alignment procedures. In addition, all the tools are designed to work on a command line interface, which makes them feasible for execution on High Performance Computing (HPC) clusters. Among the listed tools, only CLC seems to be primary designed to work with a graphic interface. Regarding the use of an input annotation in the identification of exon junctions, two algorithms entirely rely on it (CLC and Novoalign), three methods do not utilize any kind annotation (CRAC, SOAPsplice, SUBREAD) while the remaining work both with and without this additional input. The most important characteristics of each of these 14 state of the art splice aware methods are described in the following section.

CLC Genomic Workbench

CLC Genomic Workbench is a commercial software to analyze and visualize NGS data. For RNA-Seq data, it allows performing many analyses and processing steps including read alignment on a reference genome. The alignment algorithm is based on an uncompressed suffix array and uses a seed and extend approach [115]. Each base position of the read is considered as start position of a seed candidate, which is elongated as long as there are fully matching rows in the SA. The resulting list of seeds is prioritized by decreasing length and read offset and then a banded Smith-Waterman algorithm [116] is applied to the first 100 seed candidates.

ContextMap2

ContextMap2 is an RNA-Seq aligner developed in Java that uses a context-based approach to identify the best alignment for each read. The read context is defined as a set of reads all originating from the same stretch of the genome and likely corresponding to transcripts of the same or overlapping genes. The first step of the algorithm consists in an un-spliced alignment which is performed, depending on the user's choice, using Bowtie, Bowtie2 or BWA. The second step identifies read alignments close to each other and assigns these reads to the same context. The next step defines additional alignments for each read in the same context trying to identify spliced alignments. Specifically, the tool exploits the reads aligned locally (not

end-to-end) or which exceeded the max mismatch threshold to identify exon junctions. Then, it tries to create spliced alignments for previously fully aligned reads using the candidate junctions. At the end of this step, several different alignments have been created for each read, resulting in multiple alignments both within and between contexts. In the last step, the algorithm performs a multiple alignment resolution using a scoring function to determine the best alignment for each read, both within and between contexts.

CRAC

CRAC is a C++ multi-purpose tool for analyzing RNA-Seq data which includes an alignment algorithm. The alignment algorithm employs a double-indexing strategy, using a FM-Index for the genome and a Gk array [117] for the reads. Moreover, CRAC uses local coverage information in the alignment process. The tool bases the alignment algorithm on the assumption that a for a given genome length, a sequence of length k will match on average a unique genomic position with high probability, if k is computed in the right way [118]. For each read, the tool considers the k-mers starting at any position in the read and it computes two distinct k-mer profiles called the *location profile* and the *support profile*. The location profile records the k-mer exact matching locations on the genome and its number while the support profile registers for each k-mer the number of reads sharing it. The genomic locations of a k-mer are computed using the genome FM-index, while the support of a k-mer is obtained by interrogating the reads Gk arrays. The k-mers having no exact matching locations are analyzed with their support profiles in order to identify SNP, sequencing errors, indels, splice alignments or chimeric alignments.

GSNAP

GSNAP (Genomic Short-read Nucleotide Alignment Program) is SNP-tolerant read aligner available as a part of the GMAP package [119]. The tool is developed in C and it uses a suffix array and a hash table for genome indexing. The genome index consists of a list of genomic positions for each k-mer in the genome. The alignment process is performed by computing all possible seeds of length k in a read and then using the index to obtain a set of genomic positions for each seed. The set of genomic positions and the number of overlapping intervals is used by the algorithm to identify a set of possible read alignments. The set is then filtered using a scoring function based on mismatch and gap penalties. Splice alignment identification is assisted by input annotation and/or by a maximum entropy model which uses frequencies of nucleotides neighboring a splice site to identify true and false splice sites.

HISAT

HISAT (Hierarchical Indexing for Spliced Alignment of Transcripts) is a spliced aligner for RNA-Seq data and will be the core alignment engine for TopHat next version, TopHat3. The tool is developed in C++ and relies on Bowtie2 implementation to handle many indexing operations. HISAT uses an indexing scheme based on the Burrows-Wheeler transform and the FM-index, employing a whole-genome FM-index and several local FM-indexes. The hierarchical indexing strategy is designed to perform fast alignments requiring low memory usage and to handle short anchored reads. In the alignment process, for each read HISAT first tries to find candidate locations across the genome using the global FM-index. In the second step, the tool selects one of the local indexes for each candidate and uses it to align the rest of the read. Spliced alignments are identified using input annotation and/or a two-step method. First, HISAT collects candidate splice sites supported by reads with long anchors, then it uses the splice sites collected thus far to align short-anchored reads.

HISAT2

HISAT2 is an enhanced version of HISAT and it is a successor to both HISAT and TopHat2. It extends the hierarchical indexing strategy introduced in HISAT using a new implementation of the FM-index. The new indexing scheme is called a Hierarchical Graph FM-index (HGFM) and relies on a graph FM-index (GFM), which is an extension of BWT to graphs [120].

MapSplice2

MapSplice2 is a software for mapping RNA-Seq reads developed in Python and C++. It is an updated version of MapSplice and it introduces multi-threading and an improved mapping sensitivity. MapSplice2 splits each read into segments and maps them to the genome by using Bowtie. The unmapped segments are used to infer splice junctions through several steps. First, splice junctions that appear in one or more segments are analyzed to determine a splice significance score based on the quality and diversity of alignments that include the splice. In addition, a remapping step is performed to identify spliced alignments involving small exons. Last, the best candidate alignment is chosen by selecting the alignment showing the highest quality match and highest confidence splice junctions.

Novoalign

Novoalign is a C++ commercial software for read alignment that can handle both DNA and RNA reads. Since the tool is not specifically developed for RNA-Seq reads and can

not perform de-novo spliced alignments, some preprocessing and post-processing steps are required. As preprocessing step, the tool requires the creation of transcripts and splice junction sequences from the reference genome using the available annotation. The transcript sequences, splice junction sequences and genome are used by the tool to build a reference index using a k-mer hash table. Novoalign uses the index to perform an iterative search to find the best alignment and any other alignments with similar score. The score assigned to each alignment is computed using the Needleman-Wunsch algorithm with affine gap penalties [121]. In addition, read base qualities are employed in a position specific scoring schema. At the end of the alignment process, a postprocessing step is required to convert the spliced alignment coordinates back to genome coordinates.

Olego

OLego is a tool specifically designed for de novo mapping of spliced RNA-Seq reads. Olego is implemented in C++ and adopts a multiple-seed-and-extend scheme using very small seeds (12-14 nt). Using the same approach as in BWA, each read is continuously mapped to the genome through a BWT and FM-index. Reads that could not be mapped by the previous step are segmented into multiple small seeds and each seed is mapped independently. The seeds are then clustered in order to identify potential spliced alignments with the constraint that the distances between clusters are less than twice the specified maximum intron size. Candidate alignments are scored and ranked according to an E-value calculated from the uniqueness and the number of aligned seeds. The holes between seeds within each candidate exon are filled by realigning the seeds employing banded dynamic programming which allows indels and substitutions. Multiple alignments are solved prioritizing the most reliable alignments according to a regression model that combines splice-site motif score and intron size to infer the strength of exon junctions.

RUM

RUM (RNA-Seq Unified Mapper) is a RNA-Seq aligner tool implemented in Perl. The tool is implemented as a three-stage pipeline built on top of Bowtie and BLAT. In the first stage, the reads are mapped against the genome using Bowtie. Bowtie is also run against the transcriptome, since Bowtie could erroneously align a read having a small anchor to the start of the intron instead of to the adjacent exon. The genome and transcriptome alignments are filtered and merged, using a set of rules based on annotation and paired-end information to solve ambiguous cases. In the second stage, the unaligned reads by Bowtie are mapped using BLAT. A complex post-processing is then performed on BLAT output in order to reduce the

number of false alignments and to utilize paired-end information. The final stage consists of merging the Bowtie and BLAT alignments using similar rules to the first Bowtie merging step.

SOAPsplice

SOAPsplice is a de novo mapping tool for spliced RNA-Seq reads which uses the BWT to index the reference genome in the main memory. SOAPsplice performs an end-to-end alignment of each read allowing a limited number of mismatches and gaps. Then, the tool trims several bases at the 3' end of unaligned reads and repeats the end-to-end alignment for the remaining part of the read. The reads that are not mapped by the previous procedure are candidates for spliced alignments. SOAPsplice splits the unmapped reads in two segments and tries to map the longest 5' end segment first. The remaining segments are aligned following several criteria for segment length, number of mismatches, splicing signals and intron length. For reads longer than 50bp, SOAPsplice splits the reads in multiple sub-reads performing the alignment procedure described above on each sub-read. In the final step, the tool uses two main strategies to filter out false positives spliced alignments. The first strategy exploits the pair-end relationship of paired-end reads to ensure the accuracy of inferred junctions. The second strategy involves detected junction having sub-reads mapping locations not connected with each other at the same segmentation point. In order to improve the accuracy of this type of junctions, the tool requires that the number of junction reads is greater than the 25% of the average number of spliced reads supporting the non-segmented read junctions.

STAR

STAR (Spliced Transcripts Alignment to a Reference) is an alignment algorithm designed for RNA-Seq data mapping. The tool is implemented in C++ and uses an uncompressed suffix array to index the genome. In the first part of the alignment process, STAR performs a sequential search for a Maximal Mappable Prefix (MMP) in both forward and reverse directions of the read sequence. Given a read and a read location, the MMP is defined as the longest continuous substring that matches exactly one or more substrings of the genome, starting from the given read location. When one or more mismatches does not allow the MMP search to reach the end of a read, the MMP is extended allowing for alignments with mismatches. If the extension procedure does not achieve a good genomic alignment, the tool performs a soft clipping of the remaining part of the read identifying a possible low quality tail, poly-A tail or library adapter sequence. In the second part of the alignment

process, STAR identifies read alignments by stitching together all the seeds that were aligned to the genome in the first phase. Starting from uniquely aligned seeds, the aligned seeds are clustered together by proximity allowing for any number of mismatches and for one indel. A local alignment scoring scheme, with penalties for matches, mismatches, insertions, deletions and splice junctions is employed in the stitching process to score the accuracy of different solutions.

Subread(Subjunc)

The Subjunc aligner is an RNA-Seq read aligner implemented in C language and included in the Subread package. The tool employs a hash table to index the reference genome, using 16bp sequences extracted from the genome every three bases as keys. The alignment process employs a multi seed strategy, called *seed-and-vote*. The seed-and-vote paradigm uses a large number of short overlapping and equi-spaced seeds for each read, called *subreads*, mapping each seed independently without mismatches. The paradigm allows all the seeds to vote on the candidate location for the read, using a voting procedure similar to the q-gram counting. The alignment between the subreads that make up the winning voting block are performed using conventional dynamic programming algorithms allowing mismatches and indels. In order to identify spliced alignments, the tool selects the two most voted mapping locations for each read. A candidate exon-exon junction is then identified when a splicing site (a donor site '*GT*' and a receptor site '*AG*') exists and the distance between the two sets of subreads is consistent. The discovered putative exon–exon junctions are employed in a second scan step to identify the final positions of the reads, considering all mapping possibilities between exonic and spliced alignments.

TopHat2

TopHat2 is a spliced aligner for RNA-Seq reads built using Bowtie2 as core read-alignment engine. The alignment process is performed in three main steps: a transcriptome mapping step, which is used only when annotation is provided, a genome mapping step and finally a spliced mapping. In the transcriptome mapping step, the annotation is employed to create the transcript sequences. Bowtie2 is then used to index the transcriptome and align the reads. Reads that were potentially misaligned or unmapped due to unknown transcripts in the previous mapping step are aligned to the genome using Bowtie2. The unmapped reads from the last mapping step are exploited to find novel splice sites based on known splicing signals. In order to do this, the unmapped reads are split into small non-overlapping segments which are then aligned independently against the genome. The tool identifies segments that are

mapped within a maximum intron size and re-aligns the entire read to that genomic region in order to identify the most probable location of the splice site. The genomic regions flanking the candidate splice sites are concatenated in order to create a novel transcriptome. In the last alignment step, TopHat2 re-aligns the unmapped reads against the novel transcriptome using Bowtie2.

3.3.2 Splice unaware methods

The 4 splice unaware tools in this section represent widely used NGS read mappers, achieving together more than 16000 citations on peer reviewed publications. Beside the field of DNA-Seq data, these tools are important in the context of RNA-Seq data for two main reasons. First, many splice aware tools employ one of these mappers as core alignment engine in their implementation. Second, these tools find a direct application in the context of RNA-Seq data on organism having no or negligible splicing. Similar to splice aware methods, these tools are designed to work on a command line interface, allow multithreading execution and exploit genome indexing strategies to achieve faster alignment procedures.

Bowtie

Bowtie is a DNA read aligner and is not specifically developed to work with RNA-Seq data. However, Bowtie is used as core alignment engine by many splice aware tools and is also applied to align RNA-Seq reads ignoring spliced reads. Bowtie is one of the first tools employing BWT and FM-Index in the context of short read alignment and introduces also some extensions to the FM-Index. In order to extend the exact-matching algorithm employed in the FM-Index, Bowtie introduces a quality-aware backtracking algorithm that allows mismatches. The tool performs a randomized depth-first search through the space of possible alignments using a greedy approach base on the base qualities. In addition, the tool presents a novel strategy to avoid excessive backtracking called *double indexing*. This strategy relies on an auxiliary genome index containing the BWT of the genome with its character sequence reversed. The use of both indexes allows an alignment policy that limits the backtracking algorithm dividing the reads in multiple segments. First, the tool defines the first bases (about 30bp) on the high quality end of the read as a seed, limiting to two the number of allowed mismatches in this part of the read. The seed consists of two halves: a high quality part (usually the 5' end) and a low quality part. The algorithm uses both indexes to find a potential alignment starting from the seed, trying to minimize first the number of mismatches in the high quality part of the seed. The alignment process continues alternating the use of the 2 indexes between the 2 halves of the seed and the nonseed part of the read.

Bowtie2

Bowtie2 is the successor of Bowtie and introduces many new features like local alignments, gapped alignments and unlimited read lengths. Bowtie2 is implemented in C++ and employs a FM-Index to represents the reference genome. In addition, the tool uses an extension of the BWT, called bi-directional BWT, to efficiently switch between right-to-left and left-to-right alignments. The alignment process starts by extracting substrings (seeds) at regular intervals from each read and its reverse complement. Each seed is aligned allowing up to one mismatch in order to identify candidate genomic locations. If a read results in a high number of candidate genomic locations per seed, Bowtie2 performs a re-seeding procedure to reduce the ambiguity in the candidate location and reduce the computational effort. In addition, the seeds are prioritized for lower number of candidate positions, since each seed can describe many genomic locations. Finally, the tool proceeds by repeatedly selecting a genomic location and trying to identify a full length alignment extending the seed alignment through a SIMD-accelerated dynamic programming algorithm using a Smith-Waterman-like scoring scheme.

BWA

BWA (Burrows-Wheeler Alignment) is a DNA-Seq reads aligner and is one of the first tools using BWT and FM-Index. Similar to Bowtie, BWA is used as core alignment engine in several splice aware tools. The tool allows performing both exact alignment using the FM-Index backward search and inexact alignment using a bounded backtracking algorithm developed ad hoc. Essentially, the inexact alignment algorithm uses backward search to identify distinct substrings from the reference genome and the search process is bounded by the number of allowed differences (mismatches or gaps). Several strategies are then applied to reduce the complexity and increase the accuracy, including setting a limit on the maximum allowed differences in the first part of the read (seed) and assign different penalty scores for mismatches, gap opens and gap extensions.

BWAMEM

BWAMEM is a read aligner implemented as a component of the BWA package that exploits the canonical seed-and-extend paradigm. The tool employs a variant of the FM-index, called FMD-index, which builds both forward and reverse strand DNA sequences in one index. The use of a FMD-index instead of a bidirectional BWT improves the speed of exact alignment since only one index is used in the search process. In the first part of the alignment process, the tool splits the read in many segments (seeds) searching for supermaximal exact matches

(SMEM) for each seed. Formally, a maximal exact match (MEM) is an exact match that cannot be extended in either direction of the match, while a SMEM is a MEM that is not considered in other MEMs on the seed sequence. In order to reduce unsuccessful seed extensions, the seeds mapping close to each other are then clustered together and small clusters are filtered out. The algorithm extends the seed with a banded affine-gap-penalty dynamic programming technique using a heuristic similar to the X-dropoff heuristic in BLAST [122]. In addition, BWAMEM defines a minimum difference between the best end-to-end alignment score and the best local alignment score to automatically choose between the two alignment options.

Chapter 4

Design of a benchmark analysis for RNA sequencing read alignment methods

Interestingly, in the current scenario the main issue is not the lack of available solutions, since a lot of splice aware algorithms have been developed in recent years. The main problem is instead the state of confusion deriving by so many methods with conflicting claims of superiority, mainly due to a lack of thorough and systematic benchmarking studies. A solution to this problem is presented in the next chapters, where an in-depth comparative analysis of the current state of the art methods is performed.

In this chapter, two important methodological steps of the benchmark analysis are introduced and described. The first step consists in simulating a set of realistic dataset describing several real scenarios and different organisms. For the analysis of splice aware tools, reads are created from *H.sapiens* and *P. falciparum* genomes, simulating different level of complexity. In the context of splice unaware tools, the *S. cerevisiae* genome is employed to create several libraries adding an Illumina error model. The second methodological step involves the definition of a complete set of metrics to assess each method. Metrics at base, read and junction level are designed in order to provide a complete view and describe different aspects of the alignment process. In addition, several computational metrics for execution time and memory usage are defined.

4.1 Data simulation

In order to perform an accurate comparative analysis, the simulation of several datasets is a pivotal step. The use of simulated data allows knowing the true position of each reads in the genome, making possible to precisely assess the performance of each tool. On the contrary, the use of real data would lack this information and only allow collecting the number of aligned and unaligned reads. The latter is not completely informative, since a high percentage of mapped reads is not necessarily associated to the fact that those reads are mapped to the right positions. For this reason, the use of simulated data seems the best choice in the context of comparing RNA-Seq alignment methods.

4.1.1 RNA sequencing data simulators

The generation of simulated data is a very sensitive step that must be carefully considered. There are many RNA-Seq simulators available and each one is specifically designed for benchmarking alignment, quantification, normalization or differential expression. Simulators employed in the assessment of downstream RNA-Seq analysis, as differential expression, must be able to capture complex properties of RNA-Seq data as biological variability, positional bias, gene-gene correlation, realistic null distributions, etc. On the other end, for benchmarking alignments, it is not mandatory to capture those properties, since alignment algorithms do not use these complex information. In the context of alignment, the important factors to capture are indels, substitutions, sequencing error, realistic splice junctions, etc. For this reason, the BEERS simulator [112] and the Flux simulator [123] have been employed in the generation of simulated data.

BEERS simulator

BEERS was specifically designed to benchmark alignment algorithms. In order not to be biased against a particular set of gene models, BEERS is capable of using many different sets of annotations and merges them. For example, human data are simulated combining ten annotation tracks: RefSeq, GeneID, Aceview, Augustus, ENSEMBL, UCSC, Vega, GenCode, GenScan and lincRNA. The simulator first chooses a set of N gene models at random (default N=30000) and then creates alternate splice forms; both the number of alternate forms and the percentage of signal coming from splice forms are simulation parameters. Then, BEERS introduces polymorphisms (insertions and deletions) according to a specified rate. The empirical distribution of gene expression levels is generated from an input gene quantification file, allowing to mimic several real data scenarios; the same

file is used also to determinate the distribution of intronic signal. Finally, BEERS simulates reads by choosing a gene at random, choosing a fragment of normally distributed length and adding random base and tail errors. One parameter allows setting random base error rate whereas tail errors are set according to three parameters: *i)* the percent of low quality tails, *ii)* the quality of the low quality tail and *iii)* the length of the low quality tail. Setting a read length of M bases, the simulator reports M bases of the fragment from either end and puts them into a FASTA file. The tool is able to generate paired-end reads of different lengths and can also simulate strand specific reads. Several output log files contain the true coordinates of each read, the true junctions spanned, the set of gene models used and the alternate splice forms generated.

Flux simulator

Flux is a widely adopted RNA-Seq simulator, developed to model and study many aspects of RNA-Seq data. Flux models gene expression, reverse transcription, fragmentation, library preparation and sequencing, producing simulated data that are suitable for many kinds of analyses. Even though it was not specifically developed to benchmark alignment algorithms like BEERS, it presents a set of features that make it feasible for simple comparative analysis of alignment methods. Compared to BEERS, it has the main limitation of not simulating indels and providing less options for error and substitution simulations. On the other hand, Flux data could be used for the assessment of post alignment tasks such as expression level quantification and normalization, whereas BEERS data are limited to the mapping step. Flux data simulation starts by generating an expression level profile for the transcripts specified in the provided input annotation. Starting from a global number of expressed RNA molecules, the simulator assigns expression levels to each expressed transcript using a modified Zipf's Law [124]. In addition, Flux simulates two biological modifications of annotated transcripts such as variable transcription start sites and poly-A tail lengths. The next simulation step consists in simulating the fragmentation process, allowing to choose among *in silico* versions of enzymatic digestion, nebulization and hydrolysis. In addition, it is possible to select whether fragmentation is carried out before or after reverse transcription. The reverse transcription process is modeled separately for first and second strand synthesis. Both poly-dT primers and random primers techniques can be simulated and optionally a position weight matrix can be employed to describe sequence bias. Finally, Flux can perform an optional size selection step before the *in silico* PCR amplification, which captures both GC preferential biases and sequence biases. The last step in data simulation consists in the *in silico* sequencing, where the fragments in the library are sub-sampled and the sequences of an arbitrary end (single-end reads) or of both ends (paired-end reads) are obtained; Illumina

style sequencing errors could be introduced during this step. The desired library size, read length and strandedness could be set through simulation parameters. Detailed log files about the several simulation steps are produced together with a FASTA file containing the simulated reads.

4.1.2 Simulated data for splice aware methods comparison

BEERS was employed in the creation of many simulated libraries to benchmark splice aware alignment methods. Data were simulated from two genomes, H.sapiens (human) and P. falciparum (the most deadly malarial parasite) in each of three levels of complexity for a total of six conditions, each generated in triplicate for a total of 18 datasets. Each dataset consists of 10 millions 100 base paired-end strand-specific reads, giving a total of $2 \cdot 10^9$ bases per data set. The human genome was employed due to its pivotal importance, while the P. falciparum genome was chosen because it is a commonly studied organism very different from human [125]. Moreover, the P. falciparum genome is 80% AT rich making it a good candidate for testing the performance of the alignment methods on handling low complexity regions. Human data was limited to chromosomes 1-22, X and Y. For human, 30,000 transcript models were chosen at random from a conglomeration of 858,063 gene models obtained by combining ten annotation tracks: RefSeq, GeneID, Augustus, ENSEMBL, Aceview, Vega, GenCode, GenScan, UCSC and lincRNA. For each gene an alternate splice form was generated by randomly including/excluding exons. Thus, a total of 60,000 transcript models were used. Expression levels were taken from an exponential distribution with λ=0.01 applied to a random 2/3 of the transcripts, the rest were left unexpressed. Intron signal was introduced at levels representative of real data (approximately 40% intronic reads). Intron signal is introduced by inserting one intron back into the edited transcript before fragmentation. The used fragment length distribution has minimum length equal to 100 bases, mean equal to 200 bases and maximum length equal to 500 bases. The three levels of complexity, identified by the labels T1, T2 and T3, were achieved by combining different levels of substitution rate, indel rate and error rate. Table 4.1 summarizes the characteristics of each complexity level. T1 simulates data with low polymorphism and error rates, similar to aligning most regions of

Table 4.1 Simulated levels of complexity

Level ID	Substitution rate	Indel rate	Error rate
T1	0.001	0.0005	0.005
T2	0.005	0.002	0.01
T3	0.03	0.005	0.02

human RNA-Seq reads to the reference human genome. In particular, the substitution rate is 0.001, the indel rate is 0.0005 and the error rate is 0.005, which is a typical low Illumina error rate [34]. The T2 complexity level data has moderate polymorphism and error rates, similar to model organisms. Finally, the reads in T3 are designed to represent highly polymorphic regions in the reference genome which is common in organisms different from human. T3 also represents polymorphism rates which can be observed when aligning across different (but similar) species. The simulated levels of complexity make it possible to test the different alignment method performance on a wide set of real case scenarios.

4.1.3 Simulated data for splice unaware methods comparison

Data employed in the comparison of splice unaware methods were generated using Flux. The choice of Flux instead of BEERS was motivated by the need to use the data both in the mapping (Chapter 6) and post mapping (Chapter 7) assessments.

The S. cerevisiae genome was used to simulate 6 libraries of 10 million 100 base paired-end strand-specific reads. The S. cerevisiae is a species of yeast and its genome was employed due to the low number of introns, resulting in only ~0.24% of reads coming from exon junctions. This low number of spliced reads makes it safe to employ splice unaware tools. Gene expression levels were taken from an exponential distribution, starting from an initial pool of 5000000 expressed RNA molecules. In the read simulation, RNA was fragmented before reverse transcription using *in silico* hydrolysis. Then, first strand synthesis was performed using random hexamer primers and the fragments were size selected by gel segregation filtering the fragment lengths through a normal distribution with mean equal to 300 and standard deviation equal to 50. Finally, errors in the reads were added using the built-in Flux error model, which employs a Markov model to generate an error profile similar to the one observed in Illumina data.

4.2 Assessment metric definitions

An extensive set of metrics was defined in order to assess the most important aspects of the mapping process. First, a literature search was performed to identify metrics already employed in similar studies [112, 126, 127]. Then, additional metrics were defined with the goal of finding the smallest set of indices able to describe the most important characteristics of the RNA-Seq data alignment. The resulting set of metrics was organized into three levels, one for each basic concept of RNA-Seq data alignment. As such, the metrics are based on events defined as follows: a single base of a single read aligning to the right location (base level), a single read having at least one base aligning to the right location (read level) and a single read crossing a single intron (junction level). In practice, the metrics where computed comparing the ground truth available as output of the read simulation process to the SAM files resulting from the read alignment using the different tools. Specifically, the information contained in the *FLAG*, *POS* and *CIGAR* fields of the SAM file were exploited to identify the read location; details about the SAM file format are provided in Appendix A. Metrics are then based on standard measures of accuracy for each type of event. In particular, the standard accuracy metrics *precision* and *recall* are computed for each level. Alternatively, the results can be presented as *FNR (False Negative Rate)* and *FDR (False Discovery Rate)* using the relations *FNR = 1 - Recall* and *FDR = 1 - Precision*. In addition to accuracy metrics, some computational metrics were introduced in order to analyze the performance of the tools in term of execution time and memory usage.

4.2.1 Base level metrics

The base level metrics focus on the behavior of the aligner with single base resolution, resulting in the strictest metrics. The base-wise accuracy involves individual bases of the reads and whether they align uniquely and to the correct location. A single base resolution is very important in studies involving the identification of SNPs, insertion and deletions, so these metrics provide useful indications about the accuracy of the mapping tool in these scenarios. Since some cases are ambiguous, some flexibility was introduced. For example, if a TT sequence in the reference is replaced by T in the read, then the aligner will typically choose one of the two T's to be the aligned base and the other to be the deleted base. In this case, it is not reasonable to indicate which one of the two T's was retained and which was lost. Therefore, the tools are credited for specifying either of the two possibilities. The design of the metrics results from the identification of some basic concepts involved in the alignment of a single base. These basic concepts represent the events that could happen in the mapping process of a single read's nucleotide, including:

- **Aligned base**: a base is defined as "aligned" if its read is aligned and its CIGAR character in the SAM file is different from "S" and "H" (clipping).
- **Unaligned base**: a base is defined as "unaligned" if its read is unaligned or its read is aligned and its CIGAR character in the SAM file is "S" or "H" (clipping).
- **Ambiguously aligned base**: a base is defined as "ambiguously aligned" if its read is ambiguously aligned.
- **Correctly aligned base**: a base is defined as "correctly aligned" if it is aligned (uniquely, not ambiguously) and the CIGAR character in the SAM file is the same as the corresponding one in the simulator file.
- **Incorrectly aligned base**: a base is defined as "incorrectly aligned" if it is aligned (uniquely, not ambiguously) and the CIGAR character in the SAM file is different from the corresponding one in the one in the simulator file.
- **Insertion**: a base is called "insertion" if its CIGAR character in the SAM file is an "I"
- **Deletion**: a base is called "deletion" if its CIGAR character in the SAM file is a "D"

Starting from the basic concepts described above, the base level metrics are defined as follow:

- **Base level precision**: (# correctly aligned bases) / (# uniquely aligned bases)
- **Base level recall**: (# correctly aligned bases) / (total # bases)
- **Insertion precision**: (# insertions called correctly by the tool) / (# insertions called by the tool)
- **Insertion recall**: (# insertions called correctly by the tool) / (total # of real insertions)
- **Deletion precision**: (# deletions called correctly by the tool) / (# deletions called by the tool)
- **Deletion recall**: (# deletions called correctly by the tool) / (total # of real deletions)

4.2.2 Read level metrics

The read level metrics focus on the read as a unit and are appropriate for studies that do not require a single base accuracy, as for example gene level quantification. Indeed, in gene level quantification it is generally sufficient to get the correct location of the read without the constraint of having every single base correctly aligned. For example, popular tools such as featureCounts, HTSeq-count and BedTools count a read if at least one read base is found to overlap a gene.

Thus, accuracy is measured at the read level in terms of percentage of reads for which at least one base is in the right location. The basic concept involved in the read level analysis are:

- **Aligned read**: a read is defined as "aligned" if the SAM bit flag 0x4 is unset.
- **Unaligned read**: a read is defined as "unaligned" if the SAM bit flag 0x4 is set.
- **Ambiguously aligned read**: a read is defined as "aligned ambiguously" if either read in the read pair (fragment) was aligned, but has multiple entries in the SAM file.
- **Correctly aligned read**: a read is defined as "aligned correctly" if it is aligned (uniquely, not ambiguously) and at least one base of the read is mapped to the right position.
- **Incorrectly aligned read**: a read is defined as "aligned incorrectly" if it is aligned (uniquely, not ambiguously) and no base of the read is mapped to the right position.

As consequence, the read level metrics are defined as follow:

- **Read level precision**: (# correctly aligned reads) / (# uniquely aligned reads)
- **Read level recall**: (# correctly aligned reads) / (total # reads)

4.2.3 Junction level metrics

Aligning over a junction is one of the most important features for RNA-Seq aligners. This feature is so important that it defines one of the most relevant ways to classify an NGS aligner: "splice aware" versus "splice unaware". Even if a tool is able to perform spliced alignment and identify the need to split a read, the correct identification of both junction sides is particularly hard. The challenges mainly arise from two scenarios: *i)* a short anchor in one or both the junction sides or *ii)* a high sequence similarity between the read and the near exon. In the first case, the short anchor could be mapped to the wrong position due to the

ambiguity of aligning just a few bases. In the second case, the alignment of the exonic part of the read could be extended by few bases in the near intron, due to the similarity between the two sequences. Finally, both cases result in the wrong identification of at least one side of the junction. The accurate identification of exon bounds is an important task in many studies, thus this metrics allow to describe the performance of the aligner in this kind of analyses. The metric definition is achieved by the identification of some basic concepts:

- **Correctly called junction**: A junction is defined as being called correctly if both the junction start and the junction end sites were identified correctly.
- **Incorrectly called junction**: If at least one junction site was called incorrectly, the whole junction is classified as an incorrectly called junction.

The junction level metrics are defined in the most strict way, requiring both sides of the junction to be correctly identified. The resulting accuracy metrics are:

- **Junction level precision**: (# junctions called correctly by the tool) / (# junctions called by the tool)
- **Junction level recall**: (# junctions called correctly by the tool) / (total # of real junctions)

4.2.4 Computational burden metrics

The alignment process is a computational expensive task, especially when applied to large genomes and many read set. Depending on the adopted algorithm and implementation, each tool would result in different computational requirement. In order to assess the performance of the different methods, a few metrics were designed to measure the execution time and the memory usage.

- **Total run time**: the tool execution time from the start to completion of the alignment process
- **Total CPU time**: the total time the CPUs spent performing the alignment process. If only one thread/CPU is used, this time is the same of *Total run time*; if more than one thread/CPU is employed, this time is the sum of the times spent by each thread/CPU.
- **Maximum memory usage**: the maximum amount of RAM memory used during the alignment process

Chapter 5

Comparative analysis of splice aware alignment methods

Despite the pivotal importance of RNA-Seq read alignment, there have been only two extensive RNA-Seq alignment benchmarking studies which consider both the accuracy and the performance of several splice aware aligners. In 2011, *Grant et al.* [112] performed the first comprehensive comparison study, benchmarking seven splice aware tools and four splice unaware methods. In the same year, the RNA-Seq Genome Annotation Assessment Project (RGASP) consortium completed the *"Sequence Mapping and Assembly Assessment Project (SMAAP) RGASP3/dnGASP"*, a collaborative effort among researchers to compare and evaluate six spliced aligners and four alignment pipelines [127]. Both studies demonstrated that choice of alignment methods is critical for the accuracy of a RNA-Seq study and highlighted many aspects of the alignment process that need further attention.

Since then alignment methods have undergone considerable development, resulting in both new methods and updated versions of existing tools.

Ideally, a good RNA-Seq aligner should be able to accurately align several kinds of data (different organisms, error rates, etc.) in many different scenarios. The reads simulation described in section 4.1.2 was performed to obtain several kinds of data, so the last step consists in defining a set of different scenarios to analyze. In this chapter, the 14 splice aware alignment methods described in section 3.3.1 are assessed in terms of accuracy and efficiency, first introducing a complete set of analyses and then reporting the most relevant results on these scenarios.

5.1 Tested scenarios

- **Alignment using the default parameters**: the common approach to read mapping consists in studying the aligner's manual and running the tool providing as input the read files and the suggested set of parameters. Since this approach is employed in the great majority of RNA-Seq studies, the accuracy of the alignment methods in this scenario was assessed.

- **Role of input annotation**: for many organisms a high quality annotation is available and providing it as input for the aligner could potentially increase the accuracy of read mapping. This potential improvement was assessed by comparing the performance of each tool providing and omitting the annotation as input. In addition, the role of the annotation was studied in the context of short anchored reads, where the use of known exon boundaries should improve the splice junction identification. Finally, the different role of annotation in detecting canonical (i.e. the removed intron contain GT at the 5' splice site and AG at the 3' splice site) and non-canonical junctions was analyzed.

- **Role of parameters tweaking**: each alignment tool has several parameters and many of them could influence the accuracy of the alignment. Typically, these parameters control how the tool handles mismatches, insertions, deletions, spliced alignments and more generally the most critical aspects in the alignment process. In order to identify the most important parameters and how far the accuracy obtained using the defaults is from the best achievable results, a search in the parameter space was performed for each tool. This analysis allows the assessment of how far the default is from a reasonable value.

- **Role of read preprocessing**: performing a read preprocessing, usually removing low quality reads or low quality read's tails, could reduce the number of unmapped or incorrectly mapped reads. However, defining a minimum quality threshold is not trivial and removing too many reads or parts of them could bias some downstream analyses. For this reason, it is very important to understand if an aligner is able to handle such kind of data. The accuracy of the different tools in this scenario was assessed by aligning both preprocessed and unpreprocessed data.

- **Multimappers**: a common question in the context of read alignment is how to handle multireads. Many downstream analyses simply discard them, whereas more sophisticated approaches try to exploit them. To identify the recall and precision in the case of multimappers, the multiread with the most correct bases aligned was chosen and

any further calculations were based on this best alignment. This analysis allows the identification of which aligner is closer to solving the ambiguity about the location of these reads and would be more suitable for subsequently exploiting this information in downstream analyses.

- **Insertions and deletions**: the correct detection of indels has a major role in many studies involving human, since indels can alter human traits and cause diseases. Unfortunately, the identification of insertions and deletions represents one of the most challenging tasks in the alignment process. To assess the performance of the alignment methods on this complex task, a specific analysis was designed to collect accuracy metrics about indel detection.

- **Alignment speed and memory requirement**: reads alignment is a computationally expensive task, which often requires many hours and tens of GB of RAM. Understanding which aligner has the best performance in terms of execution time and memory requirements, achieving at the same time a high level of accuracy, is an important information. To measure the amount of resources employed by each tool, the global run time, the single CPU time and the maximum amount of RAM memory were collected.

In the following sections a more complete description of each analysis is provided, together with the most interesting and important results. Generally, the aligners show comparable results on P. falciparum and human dataset, resulting in a consistent trend between the two organisms. For this reason, while the results on both human and P. falciparum datasets are described in the following sections, the figures in this chapter show only the results on human datasets. The most relevant figures about the assessment on the P. falciparum datasets are presented in Appendix B. For both organisms, the details about the alignment parameters are described in *Baruzzo et. al* [128].

5.2 Results using the aligner tools default parameters

For each tool, the alignment parameter configurations were designed starting from the default parameters. When the tool provided specific preset parameters or precise suggestions to increase the quality of the alignment, these suggestions were followed. In addition, the parameters related to the read (read length, fragment length, inner mate distance, etc.), or related to the kind of genome (suggested seed size, k-mer size. etc.) were set if the default values were not feasible for the current data. Since in the real scenario the users have only a standard knowledge of the dataset, the information about the error rate, indel rate and polymorphism rate coming from the read simulation were not exploited. For the alignment of human libraries, RefSeq [129] was used as provided annotation, whereas for the mapping of P. falciparum libraries the standard annotation provided by plasmoDB [130] was employed. Since the data were generated by a random selection of gene models from different annotation tracks, no biases were introduced. In addition, several tools present a set of parameters to optimize the execution time or the memory usage, defining different trade-offs between these two elements. Since the amount of memory is usually a minor concern compared to the execution time, the parameters which achieved the best execution time were chosen. Importantly, only the performance parameters that allowed a lower execution time without any loss of accuracy were employed.

5.2.1 Base level

The base level metrics are the strictest and they require the highest degree of accuracy from each tool. There are three ways to be wrong at the base level: a base can either be not aligned at all, aligned to the wrong place, or aligned ambiguously to several places.

On both P. falciparum and human libraries, the base level precision is usually high. The great majority of the aligners shows a precision greater than 95% in the most complex library, while on less complex libraries the precision is often greater than 98%. On the other hand, the base level recall shows a more variable trend, highlighting serious issues in some tools. As an example, the base precision and recall for human dataset are shown in Figure 5.1 while the results broken down by different classes of misalignment (aligned to wrong location, aligned ambiguously, and non-aligned) are given in Figure 5.2. Interestingly, the accuracy in terms of base level precision and recall is consistent between the two organisms, meaning that the tools seem robust to different kind of genomes.

On human T1 libraries, the best recall is achieved by MapSplice2 (97.82%), the worst by CRAC (86.07%), while on P. falciparum the best recall is achieved by CLC (99.32%), the worst by CRAC (92.36%). ContextMap2, GSNAP, MapSplice2, STAR appear to be the best

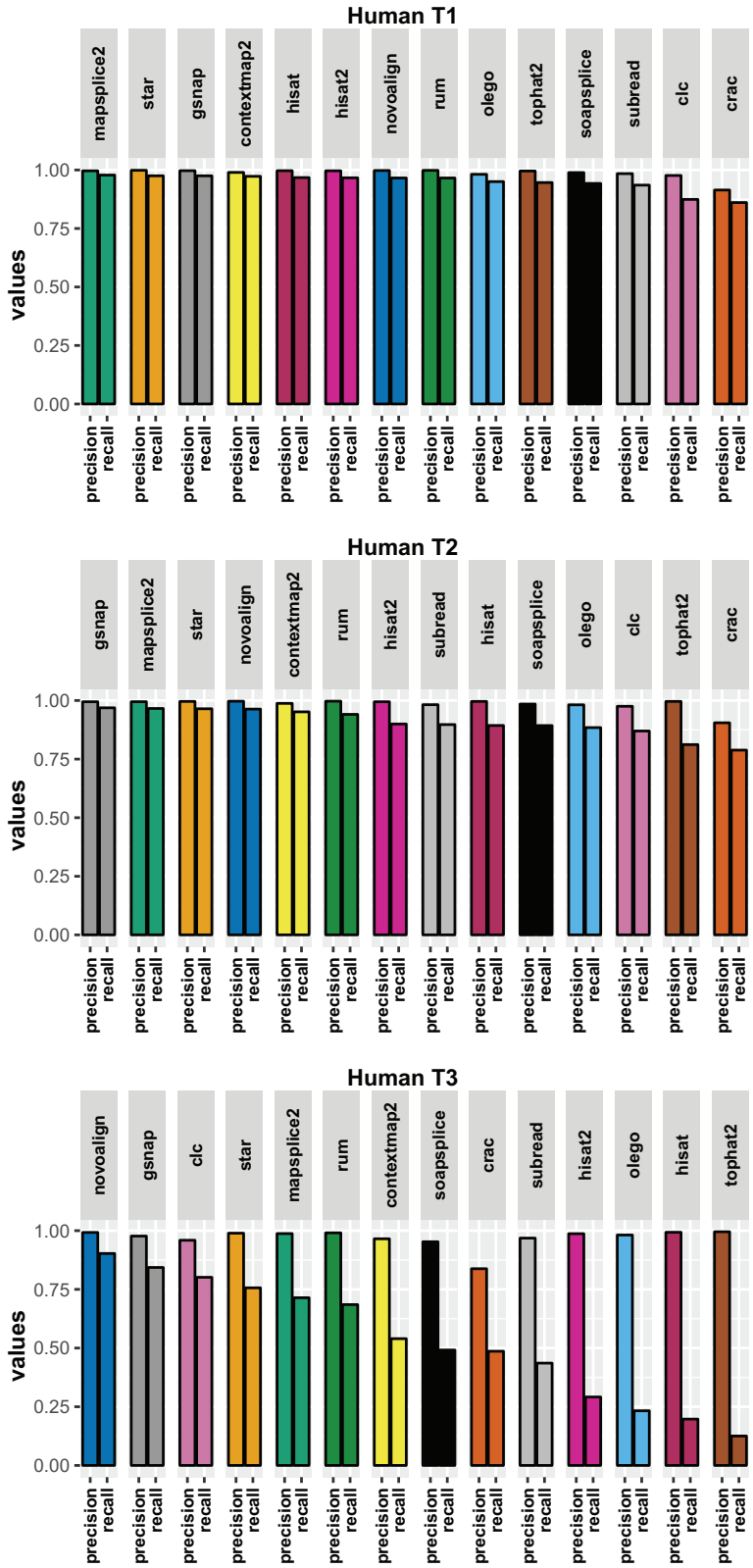

Fig. 5.1 *Default parameters - Base level precision and recall for the human datasets.* The tools are sorted by descending recall. While the precision is almost always greater than 90% across tools and complexity levels, the recall shows a more variable trend.

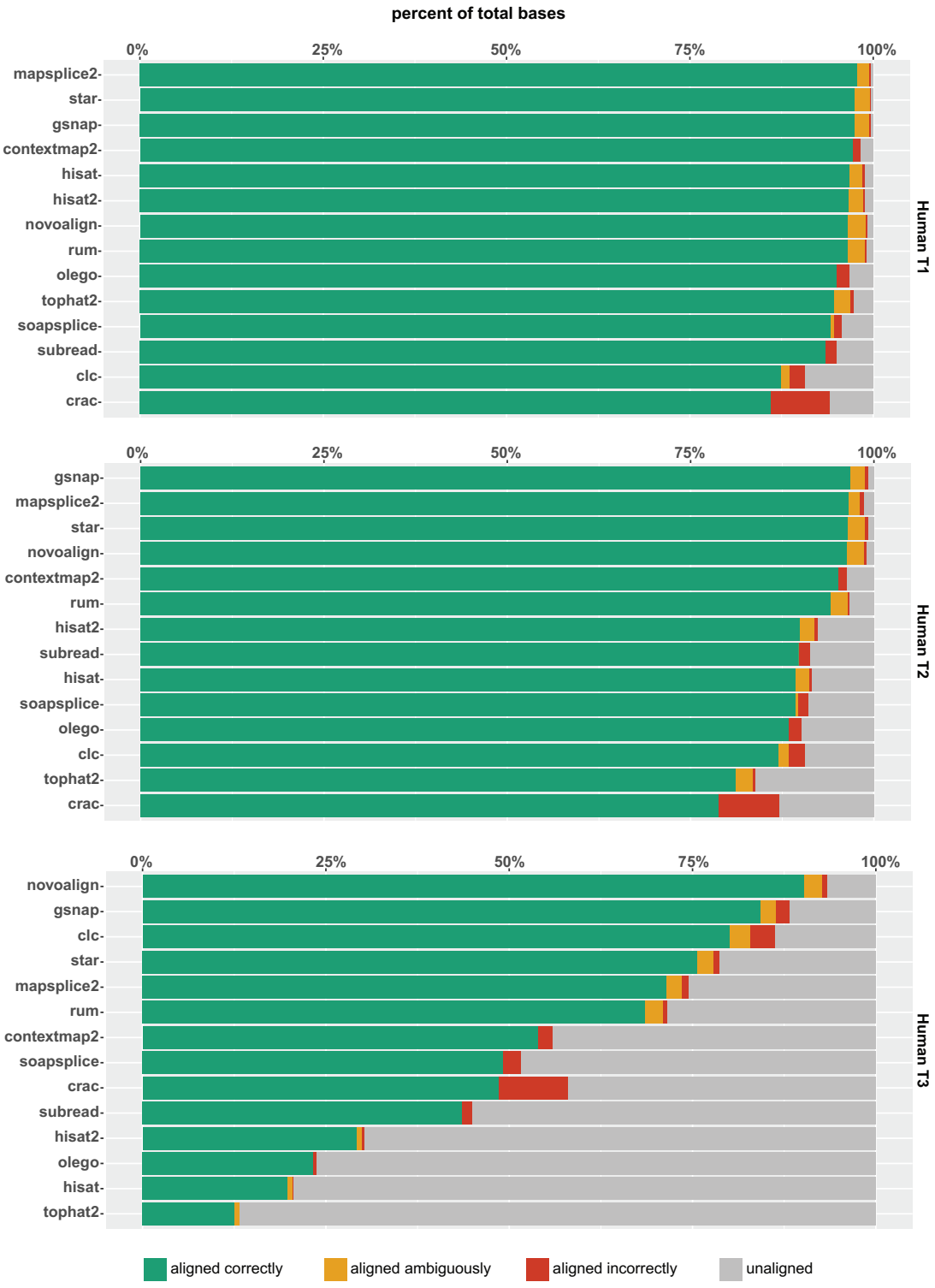

Fig. 5.2 *Default parameters - Base level statistics for the human datasets.* For each dataset, the bars show the percentage of bases aligned correctly, aligned ambiguously, aligned incorrectly and unaligned by each tool. The tools are sorted by descending percentage of bases aligned correctly. Increasing the complexity of the datasets from T1 to T3, the aligning performance changes drastically for the majority of the tools.

for both organisms, followed by RUM and Novoalign. Curiously, for T1 level complexity CLC is the best aligner on P. falciparum but second from the worst for human. In contrast to T1, P. falciparum and human T2 libraries show a significant difference in performance. GSNAP shows the best recall on human (98.88%), while the worst are CRAC (78.83%) and TopHat2 (81.19%). Five tools maintain a recall greater than 95% (ContextMap2, GSNAP, MapSplice2, Novoalign, STAR). On P. falciparum the best recall is achieved by CLC (98.94%), the worst by TopHat2 (72.15%). GSNAP, MapSplice2 and Novoalign perform best on T2 complexity on both organisms, followed by ContextMap2 and STAR. The T3 libraries show a large difference amongst the tools. On human the best recall is achieved by Novoalign (90.27%) while the worst belongs to TopHat2 (12.53%). Similarly, on P. falciparum TopHat2 has the worst recall at 2.07%, while CLC has the best at 91.81%. Novoalign, GSNAP, CLC, STAR, MapSplice2, and RUM are the only tools able to exceed 50% on both organisms.

In conclusion, on all P. falciparum datasets CLC is consistently the best at the base level, while Novoalign and GSNAP also perform well. For human libraries, Novoalign, GSNAP, MapSplice2 and STAR are the best options. A very important result is the TopHat2 performance on libraries T2 and T3, where the tool is consistently among the worst performers on both human and P. falciparum libraries. This is notable because TopHat2 is being used in more than 35% of all publications.

5.2.2 Read level

At the read level, a read is considered to be properly aligned as long as it is not a multi-mapper and at least one of its bases is aligned to the correct position. Read level analysis is most relevant for gene level quantification, because a read that has at least one base aligned correctly will usually increment the correct gene.

Read level results reveal a trend similar to the base level. For both organisms, the read level precision is almost always greater than 97-98% across tools and complexity levels. Due to the greater flexibility in the metric definition, read level precision is usually higher compared to base level precision. Read level recall shows the same variable trend identified at base level. On both organisms, there is a huge variability between tool accuracies, especially for the most complex libraries. The different read level performance of the tools on human can be seen in Figure 5.3 and Figure 5.4.

For human and P. falciparum T1 libraries, all the tools except CLC map more than 96% of the reads and show an average precision of ~99%. Even though T1 libraries already show variability in the read level recall, this trend is more evident in T2 libraries. In human, the read level recall range is between 97.83%(GSNAP) and 81.20%(TopHat2), while in P. falciparum the range goes from 99.43%(CLC) to 72.18%(TopHat2). The high level of

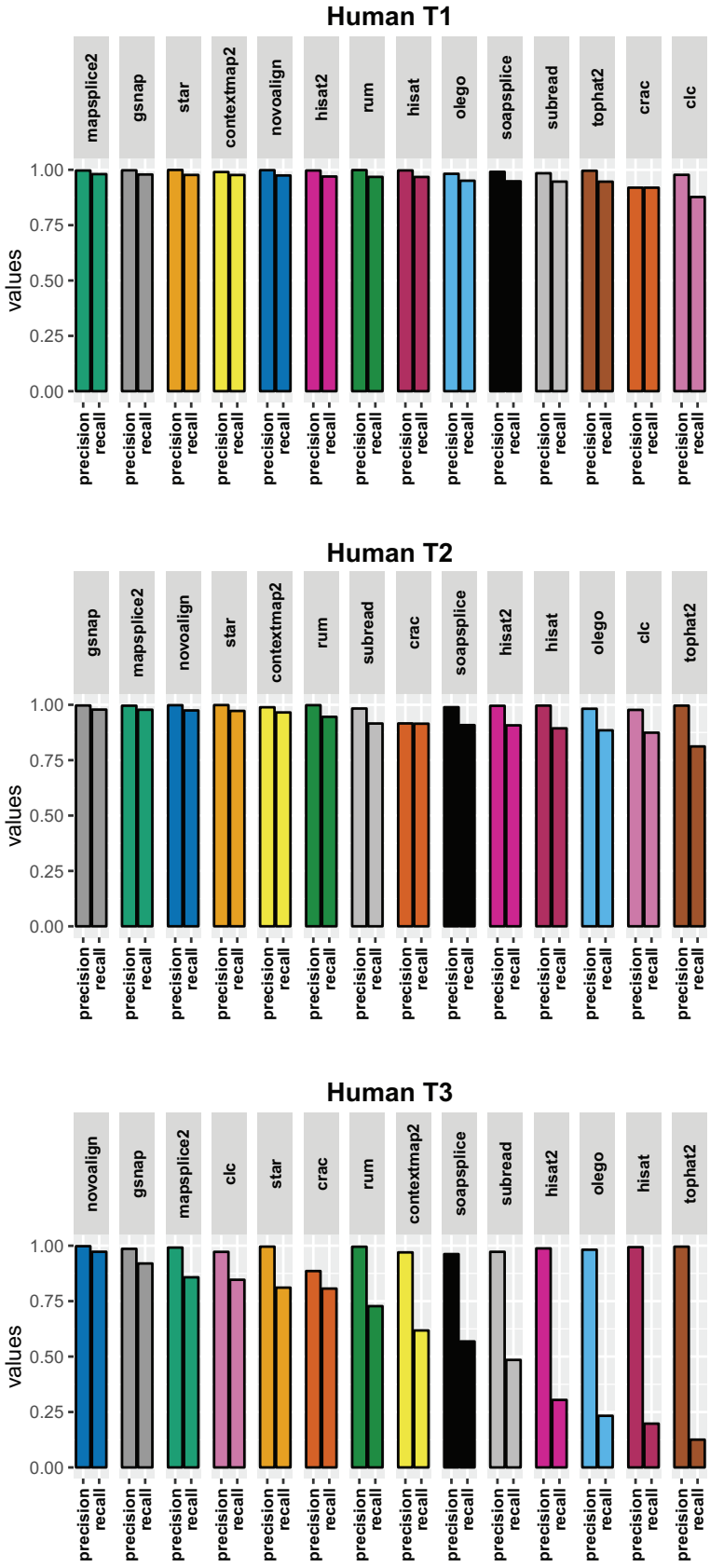

Fig. 5.3 *Default parameters - Read level precision and recall for the human datasets.* The tools are sorted by descending recall.

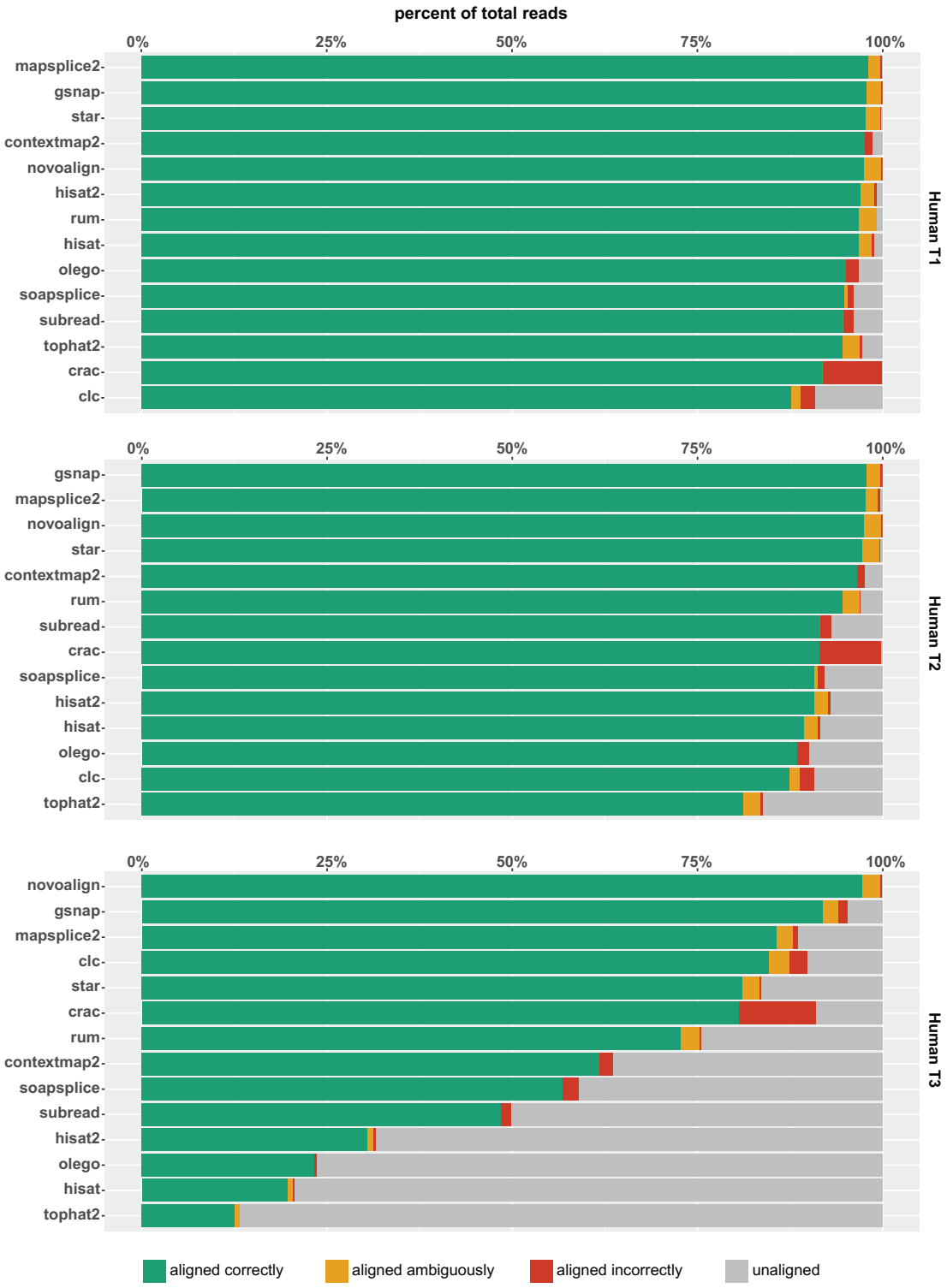

Fig. 5.4 *Default parameters - Read level statistics for the human datasets.* For each dataset, the bars show the percentage of reads aligned correctly, aligned ambiguously, aligned incorrectly and unaligned by each tool. The tools are sorted by descending percentage of reads aligned correctly.

complexity of T3 libraries amplifies the differences between tools and highlights the poor performance of some tools in handling complex data. In T3 libraries the read level accuracy ranges from 2.08%(TopHat2) to 98.27%(Novoalign) for human, while for P. falciparum the range goes from 12.53%(TopHat2) to 97.26%(Novoaling).

These results highlight that the default settings of many tools are not flexible for handling data of different complexity, suggesting that these tools are not feasible options without tweaking the input parameters. Consistent with base level analysis, Novoalign, CLC, STAR, GSNAP and MapSplice2 result as the most robust and best performing tools on both organisms also at the read level.

5.2.3 Junction level

Accuracy of junctions' identification is one of the most important aspects of a good splice aware aligner since this information is used extensively in many downstream analysis programs. In particular, software for reconstructing alternative splicing events depend heavily on these junction calls. In junction level analysis, an algorithm is defined to call an event correctly if it aligns the read not ambiguously and it properly identifies the left and right boundaries of the intron. The results for the human datasets are shown in Figure 5.5.

As for the base and read level analyses, precision is fairly high for most algorithms. On the other hand, junction recall is often low even in the less complex libraries, highlighting that spliced alignments is one of the most challenging tasks in the mapping process. For example, in dataset T2 the majority of tools show a junction recall less than or equal to 75%, while for the same dataset a value of ~75% defines the lower bound for read recall. CRAC and SOAPsplice are consistently the worst performers on T1 and T2 libraries, while CRAC and TopHat2 achieve the poorest recall on the T3 library. Again, human and P. falciparum dataset show consistent results, even if the average recall is higher in the P. falciparum genome. Curiously, CLC is the top performer in all datasets except for human T1 and T2 which are two of the least complex datasets. Among the different complexity level and organisms, the most consistently accurate performers are CLC, STAR and Novoalign.

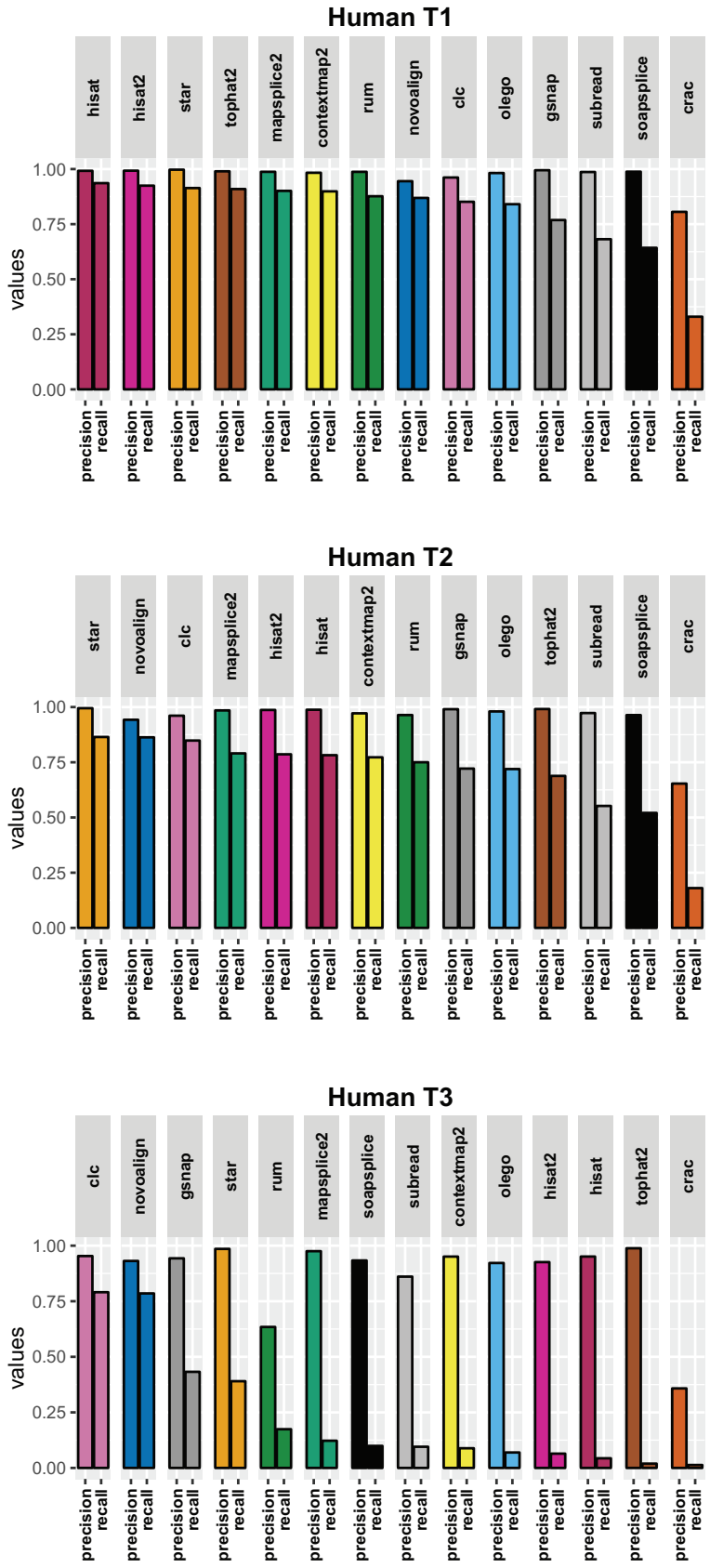

Fig. 5.5 *Default parameters - Junction level precision and recall for the human datasets.* The tools are sorted by descending recall. The precision is almost always greater than 97-98% across tools and complexity levels, the recall shows a more variable trend.

5.3 Effect of input annotation

In order to improve the alignment accuracy, many tools allow providing an annotation as input. In particular, the use of a high quality annotation could increase the ability to identify splicing junctions. A set of tests both providing and omitting the annotation was performed to quantify the improvement due to the extra information available.

The two conditions were not compared on CLC Genomic Workbench and Novoalign, because these tools are able to perform spliced alignment only with annotation. Conversely, CRAC, SOAPsplice and Subread do not allow to provide any kind of annotation as input.

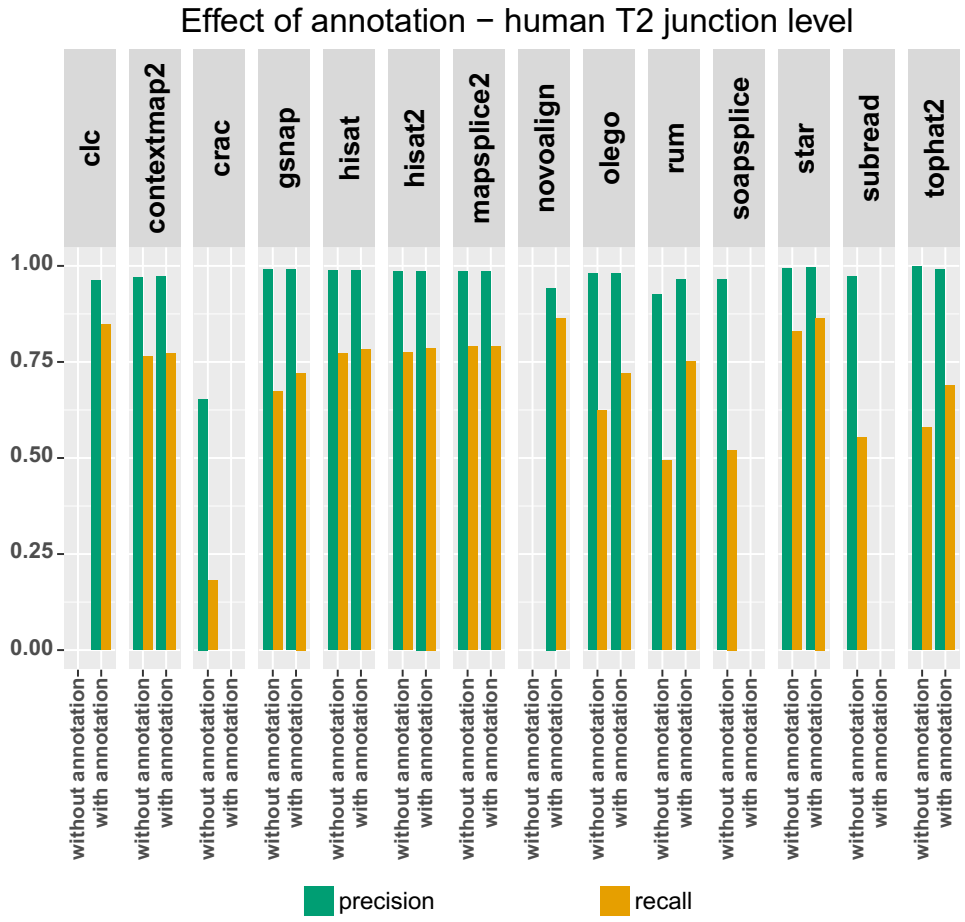

Fig. 5.6 *Effect of annotation - Junction level precision and recall for the human T2 dataset.* The improvement due to the annotation is negligible for many methods, only GSNAP, OLego, RUM, STAR and TopHat2 show some relevant differences

As expected, the results highlight that the use of the annotation rarely provides a significant improvement at the read and base level, both in P. falciparum and human. Indeed, at the base level the annotation improves performance only for those bases near splice junctions.

The further away a base is from a junction the less annotation will help and if the read does not cross a splice junction at all then annotation is unlikely to make a difference to any of its bases. Similarly, the annotation would rarely affect the read level accuracy. As a proof, the most significant difference at the read and base level on human T1 is shown by RUM and TopHat2, where providing the annotation brings a recall improvement of ~1.5% and ~1%, respectively.

On the other hand, the use of annotation shows its most relevant effects at the junction level, both in P. falciparum and human. Often, using the annotation allows increasing the junction recall while has a minor effect on the precision. In both organisms, the only relevant differences were identified in GSNAP, OLego, RUM, STAR and TopHat2. An example of this improvement is shown in Figure 5.6.

5.3.1 Short anchored read

Reads extended by only few bases in one of the spanned exons (less than 10bp), can be very difficult to align, keeping in mind that introns typically are thousands if not tens of thousands of bases in length. Indeed, these short sequences (heretofore referred to as "anchors") may be easily aligned to the adjacent next intron, soft/hard clipped, or aligned in a wrong position, depending on the algorithm. If an anchor has length one, then there is in fact a 25% chance it will align perfectly to the adjacent intron and in general an anchor of length n has a $1/4^n$ chance of aligning to the adjacent intron perfectly. In this context, the input annotation could help define the exon boundaries consequently improving the ability to accurately align short anchors. In order to study the ability of the tools to handle short anchors, a set of accuracy metrics was collected for the reads containing such anchors. First, reads having an anchor of length less than or equal to 8bp were identified, filtering for reads that spanned only one junction and that had no indels, which represents the easiest case. In this way, the results would be mainly affected more by the ability of each tool to handle the short anchored reads, than by the ability to manage other alignment issues.

The results show a highly variable performance between tools, as show in Figure 5.7. CRAC, GSNAP, SOAPsplice have the most trouble with short anchors, while STAR and MapSplice have trouble with anchors of one or two bases, but perform well on longer anchors. HISAT, HISAT2 and ContextMap2 were remarkably accurate even on the shortest anchors, even without annotation. The tools that better exploit the extra information available through the input annotation are RUM and TopHat2.

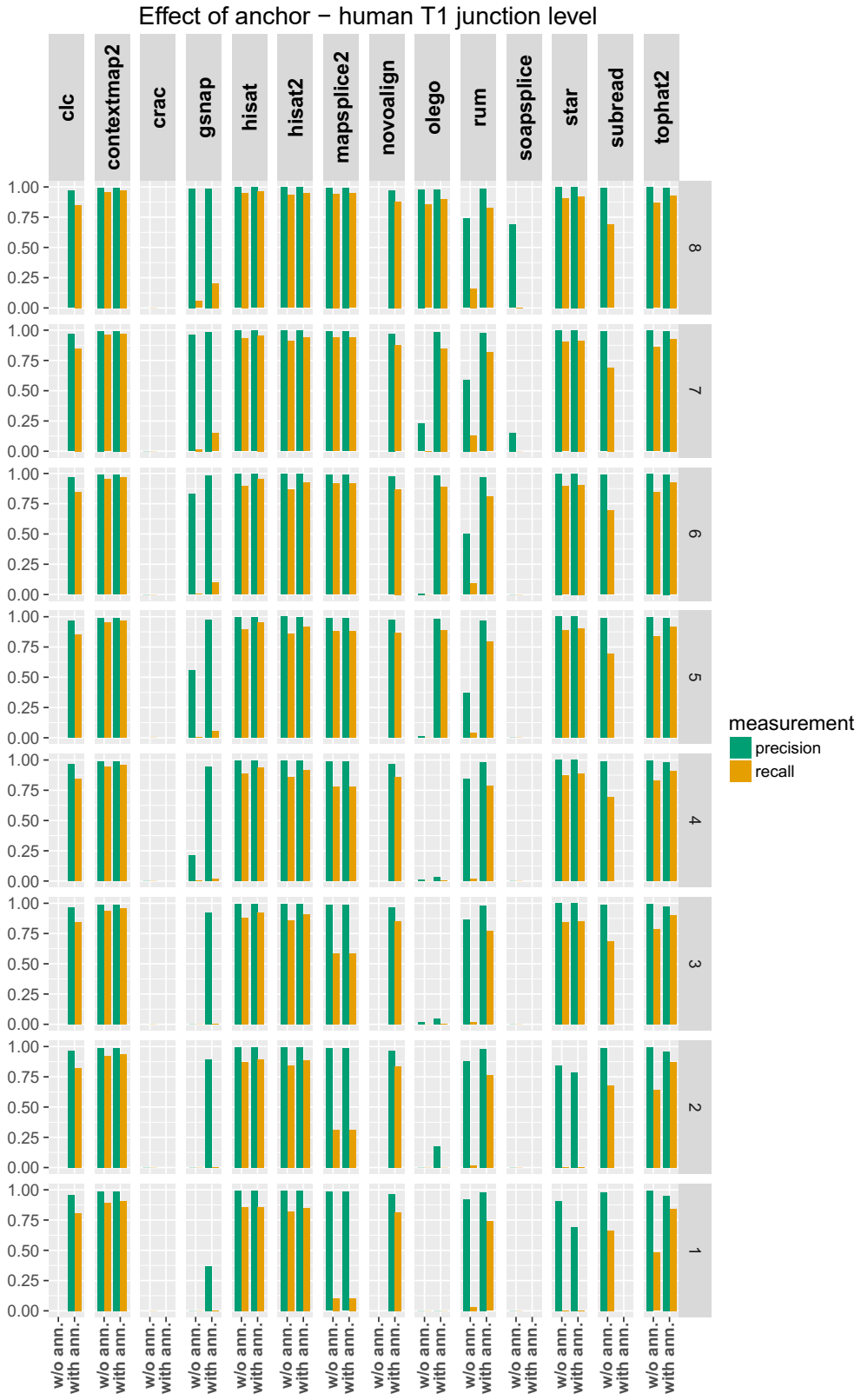

Fig. 5.7 *Analysis of small anchors in junction calls - human T1 dataset.* Precision and recall are shown as a function of anchor size from one to eight bases. Tests are performed both providing (*with ann.*) and omitting *(w/o ann.*) annotation as input.

5.3.2 Canonical vs. non canonical junctions

In the identification of a spliced alignment, one of the most challenging case is when the first base of an intron matches the first base of the following exon. Such cases can often be resolved by prioritizing canonical signals, but there is considerable latitude in how aligners deal with this issue in general.

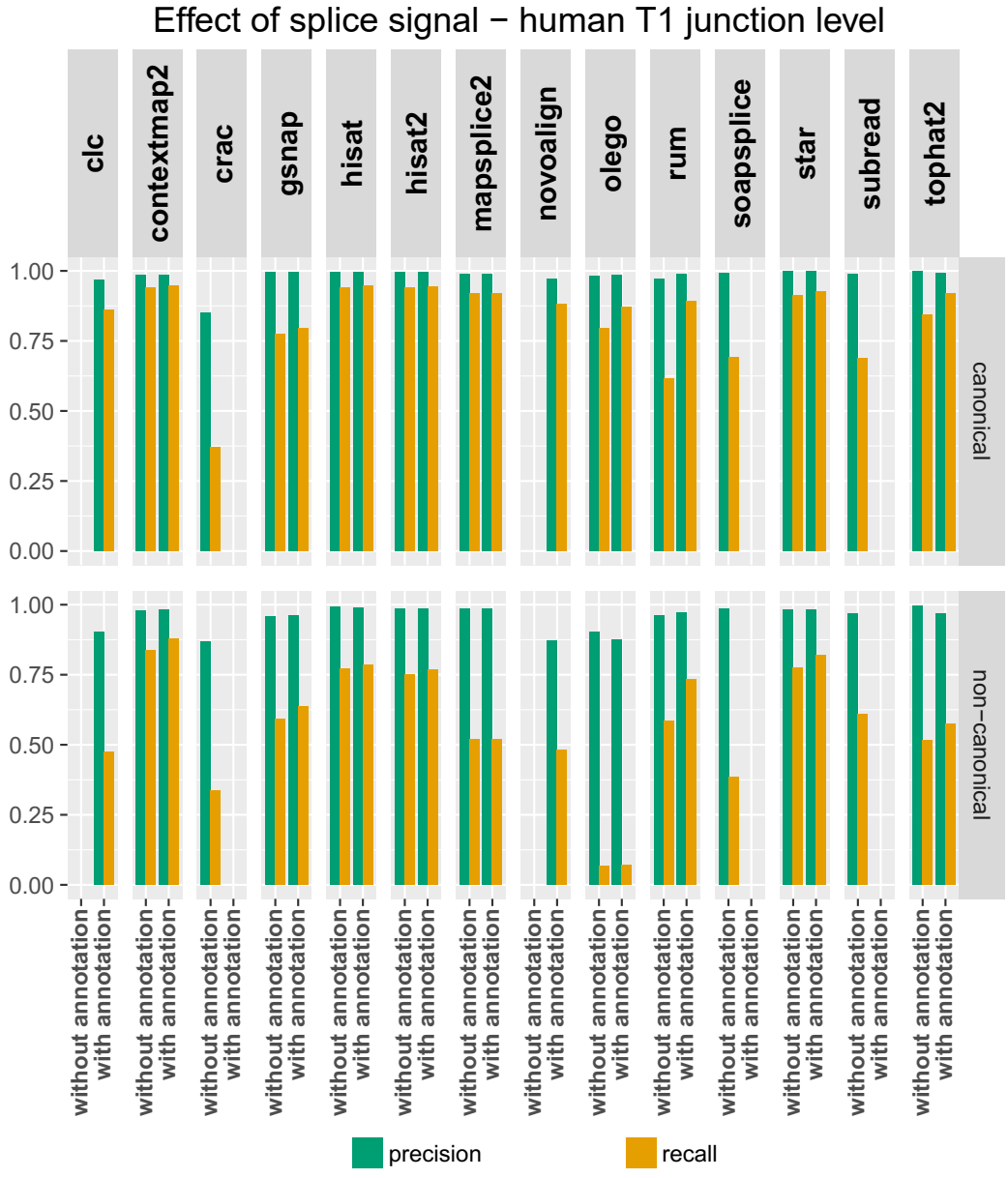

Fig. 5.8 *Analysis of splice signal in junction calls - human T1 dataset.* Precision and recall are shown separately for canonical and non-canonical splice junctions. Identification of non-canonical junctions is more challenging than canonical junctions.

As the gene models were taken from real annotation, the vast majority of splice junctions are canonical (between 98% and 99%). As such, combining all splice junctions into one metric hides any differential performance on non-canonical junctions. Therefore, analyses of both canonical and non-canonical junctions were performed separately.

The results show that all algorithms have significantly lower accuracy on non-canonical junctions as compared to canonical junctions, and no algorithms' performance on non-canonical junctions improves very much with annotation. The effect of the annotation in this scenario for human T1 dataset is shown in Figure 5.8

ContextMap2, HISAT, HISAT2 and STAR do the best overall with GSNAP, RUM and TopHat2 show moderate performance. OLego, CRAC and SOAPsplice have the worst performance on non-canonical junctions.

5.4 Effect of parameters tweaking

Each algorithm has at least several if not dozens of parameters and many of them could directly affect the alignment accuracy. However, it is almost impossible to perform an appropriate parameter tweaking in the real scenario, since there is no way to assess the final accuracy. Indeed, it was determined from the literature search that a large number of investigators do not modify the parameters and simply use the defaults. It is important therefore to explore the effect of the parameters on performance and to favor those algorithms which perform well with defaults. At the same time, collecting some indicators about the role of each parameter could be useful to develop a set of best practices for each aligner.

For each algorithm the parameter space is enormous and each alignment can require many hours to run even with many nodes on a compute cluster, thus a heuristic strategy to search the parameter space must be used which may not necessarily produce a global optimum. Therefore, for each tool, the subset of parameters and values to test were designed following these criteria:

- Indications found on the tool manual about the role and importance of each parameter
- Suggestions from the tool's authors about the role and importance of each parameter
- Use of common parameters which were demonstrated to have a major role in the alignment (e.g. number of allowed mismatches, seed/k-mer length, etc.).

Parameter optimization was performed on the T3 complexity datasets because those are the datasets where there is the greatest room for improvement. Unfortunately, it is generally not possible to optimize both precision and recall at the same time, or even to optimize either one at the base, read and junction level simultaneously. Since precision is already high in most cases, the focus was on optimizing the recall, which was done independently for the base, read and junction level. Globally, a total of 10900 different parameter configurations were tested during the tuning process. For each tool, the parameter configurations which achieved the highest recall at the base, read and junction level and the list of tweaked parameters are shown in Appendix C. The complete list of tested configurations and a set of indicators about the role of each parameter are presented in *Baruzzo et al.* [128].

For human data, the effects of parameter tweaking are shown in Figure 5.9, Figure 5.10 and Figure 5.11. Except for RUM the parameter tweaking always improved the recall at the base, read and junction level in both human and P. falciparum. ContextMap2, HISAT, HISAT2 and TopHat2 show the highest recall improvements at any level, followed by SOAPsplice and Subread. Interestingly, at the junction level ContextMap2 and TopHat2 show poor performance using the default settings, while after tuning they become some of the best

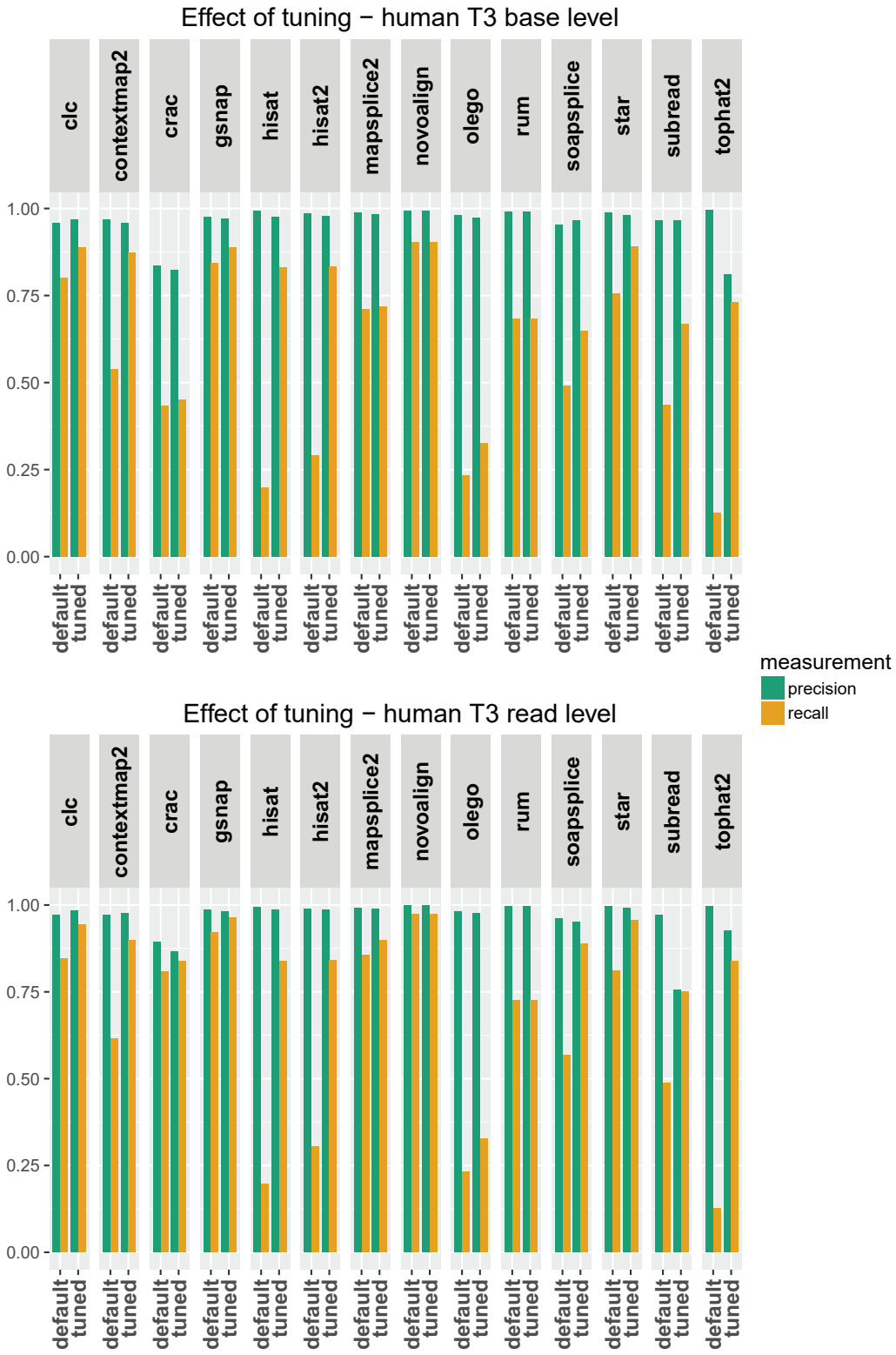

Fig. 5.9 *Effect of parameter tweaking - Precision and recall at the base and read level for the human T3 datasets.* For each tool, the figures show the precision and recall at the base level (top) and read level (bottom) for the "default" and the "tuned" alignments. The "tuned" alignment is the best configuration (in terms of base recall (top) and read recall (bottom)) achieved by the tweaking process.

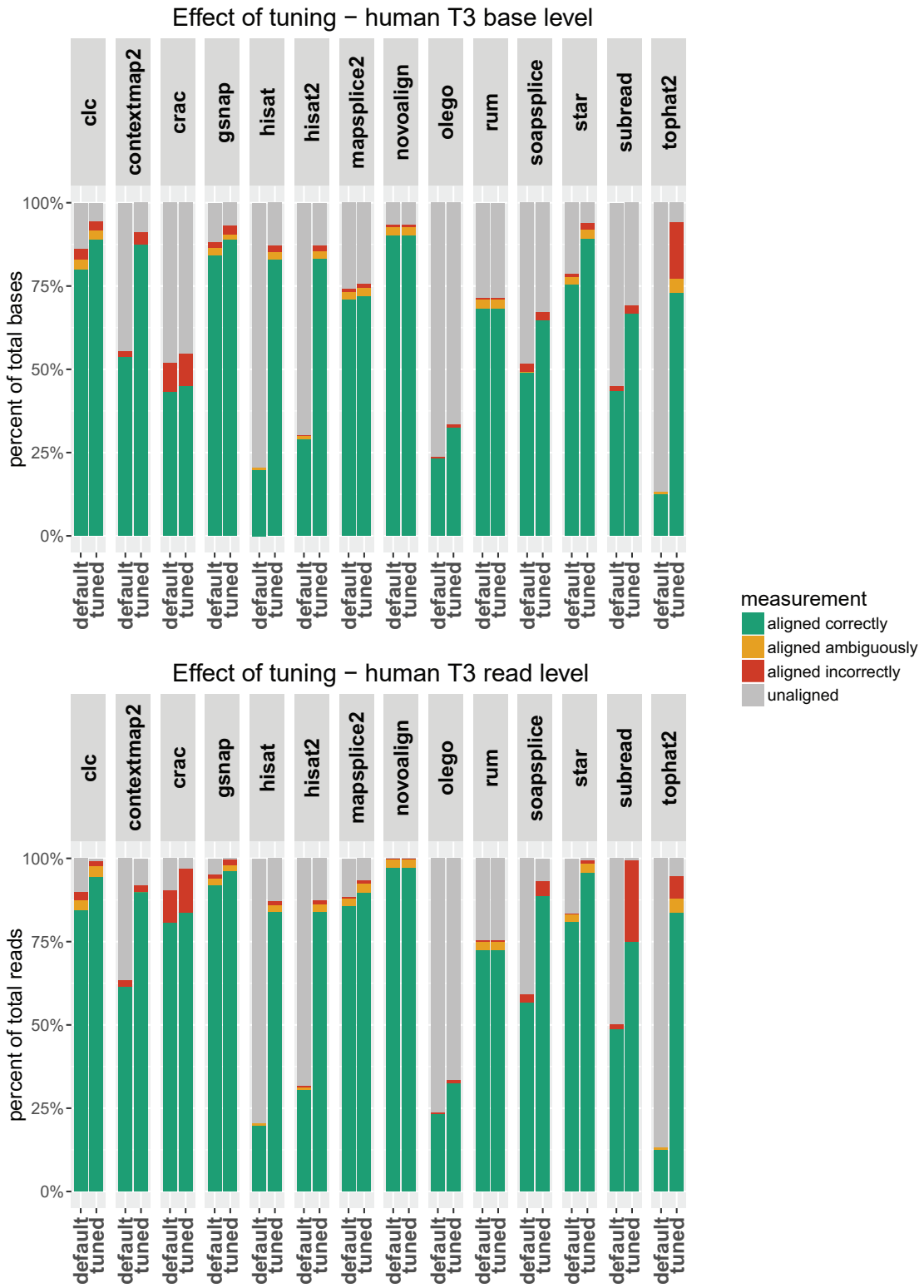

Fig. 5.10 *Effect of parameter tweaking at the base and read level for the human T3 dataset.* The bars show the percentage of bases/reads aligned correctly, aligned ambiguously, aligned incorrectly and unaligned by each tool. For each tool, the figures show the alignment statistics for the "default" and the "tuned" alignments. The "tuned" alignment is the best configuration (in terms of base recall (top) and read recall (bottom)) achieved by the tweaking process.

options. On the contrary, the best performers with the default settings (CLC, Novoalign, GSNAP and STAR) show only small improvement from parameters tweaking, still achieving some of the highest recalls. In addition, in the great majority of cases the precision achieved after parameter tuning are comparable to the ones using the default parameters. In just a few cases, the greater number of bases/reads/junctions aligned correctly brings also an increasing number of incorrectly aligned ones, resulting in lower precision.

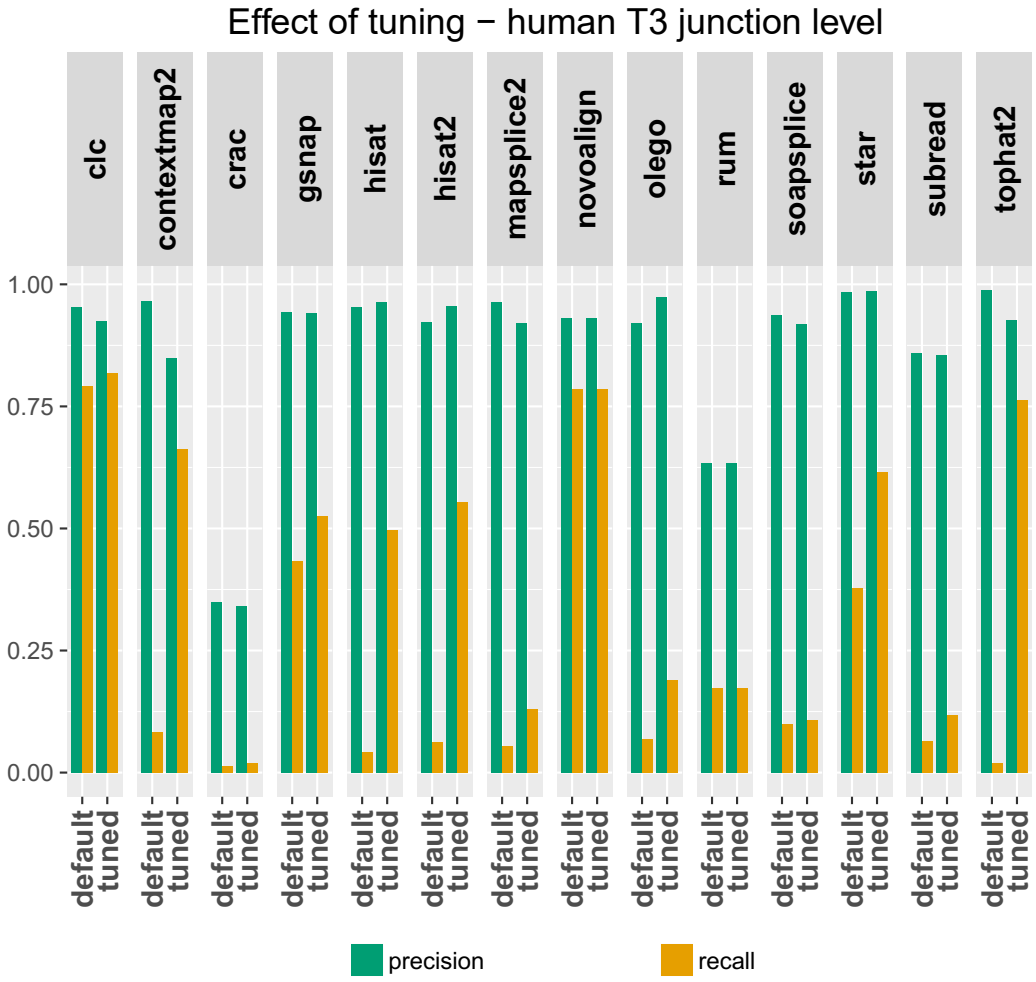

Fig. 5.11 *Effect of parameter tweaking at the junction level for the human T3 dataset.* For each tool, the figure shows the precision and recall at the junction level for the "default" and the "tuned" alignments. The "tuned" alignment is the best configuration (in terms of junction recall) achieved by the tweaking process.

5.5 Effect of read preprocessing

A common task in many RNA-Seq studies consists in performing a read preprocessing step. Reads preprocessing is performed just before the alignment process and consists in discarding low quality reads or trimming them, both to remove low quality tails or adapter sequences. In order to study the role of this step on the mapping accuracy, a specific analysis was designed simulating the scenario of reads containing adapter sequences. Starting from 500000 read pairs in the human T1 dataset, adapter sequences of length 10, 20, 30, 40 and 50 bp were simulated at fragment ends, using the Illumina Universal Adapter as template. In addition, errors were introduced in the added adapter sequences at the same rate as the T1 dataset. Then, the popular tool CutAdapt was employed to identify and remove the adapter sequences. Finally, both the trimmed and untrimmed version of the dataset were aligned by each tool, using the same default alignment settings described in the previous sections.

Results are shown in Figure 5.12. The performances at the read and base level show similar trends, even if precision and recall at the base level are lower due to stricter metric definitions. The results show that most algorithms are robust to short adapters, while medium length adapters are more challenging. Once the adapter sequences reach 50 bases they cause considerable problems for all algorithms, both with or without trimming. On the other hand, trimming is always necessary for HISAT, OLego and TopHat2, even on short length adapters. The poor performance of HISAT, OLego and TopHat2 on the untrimmed datasets are probably related to the inability to perform local alignment, since these tools are the only ones that do not implement the soft-clipping feature. GSNAP, Novoalign and STAR are consistently the best performers on both preprocessed and unpreprocessed data.

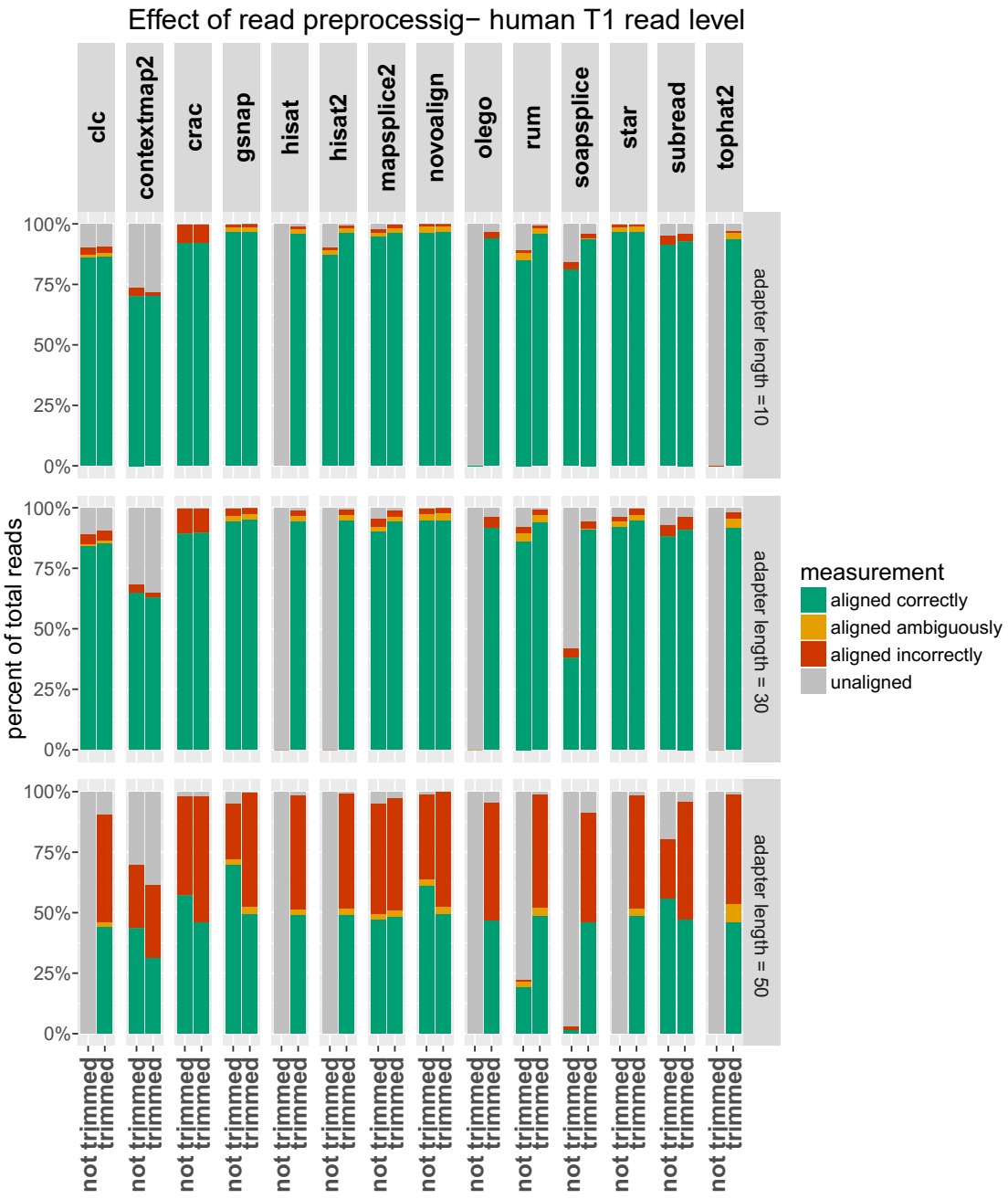

Fig. 5.12 *Effect of preprocessing at the read level for the human T1 "adapter" dataset.* The bars show the percentage of reads aligned correctly, aligned ambiguously, aligned incorrectly and unaligned by each tool on datasets containing adapter sequence of lengths 10, 30 and 50 bases. The labels "not trimmed" and "trimmed" refer to the trimmed and untrimmed versions of the dataset.

5.6 Multimappers

The percentage of reads mapped to multiple locations depends heavily on the chosen aligner, the read lengths and the organism. For example, on a typical RNA-Seq study involving human the percentage of 100bp paired-end reads mapped to multiple locations is usually lower than 5-10% [127, 131]. Due to the ambiguity deriving from these reads, the common approach in many downstream analyses consists in discarding multimappers. However, multimappers are crucial for the correct operation of some popular tools such as RSEM, Cufflinks and eXpress. Therefore, if one of the multiple alignments is the correct one, one of these post-mapping processing tools may be able to "rescue" it. To include the multimapping reads in the analysis, the recall/precision were redefined as follows. For each multimapping read, the alignment with the most correct bases is considered the "best" and is used in the precision/recall computation as if it were a unique alignment of that read.

Figure 5.13 and Figure 5.14 show the results on the human datasets. The tools show consistent results across different organisms and complexity levels, both at the read and base level. The "best" multimappers is very often the correct one, highlighting the ability of the alignment tools to identify the correct location in the set of the candidate ones. Generally, including the best multimapper results in a higher recall and a slightly lower precision. The different improvements achieved by the tools depend both on the initial percentage of multireads, which differs between tools, and the ability to identify the correct alignment even in the most ambiguous cases. On the T1 and T2 human libraries, the percentage of "best" multireads that are correctly mapped is always greater that 90%, except for CLC (<60%) and SOAPsplice (~78%). On the T3 library, the percentage of rescued multireads correctly mapped drops by ~10%, compared to the T1 and T2 libraries. Again, CLC shows the worst performance, aligning correctly only ~68% of the "best" multimappers. The same trends are shown also in the P. falciparum libraries, even though the initial percentage of ambiguously mapped reads is generally lower compared to human libraries.

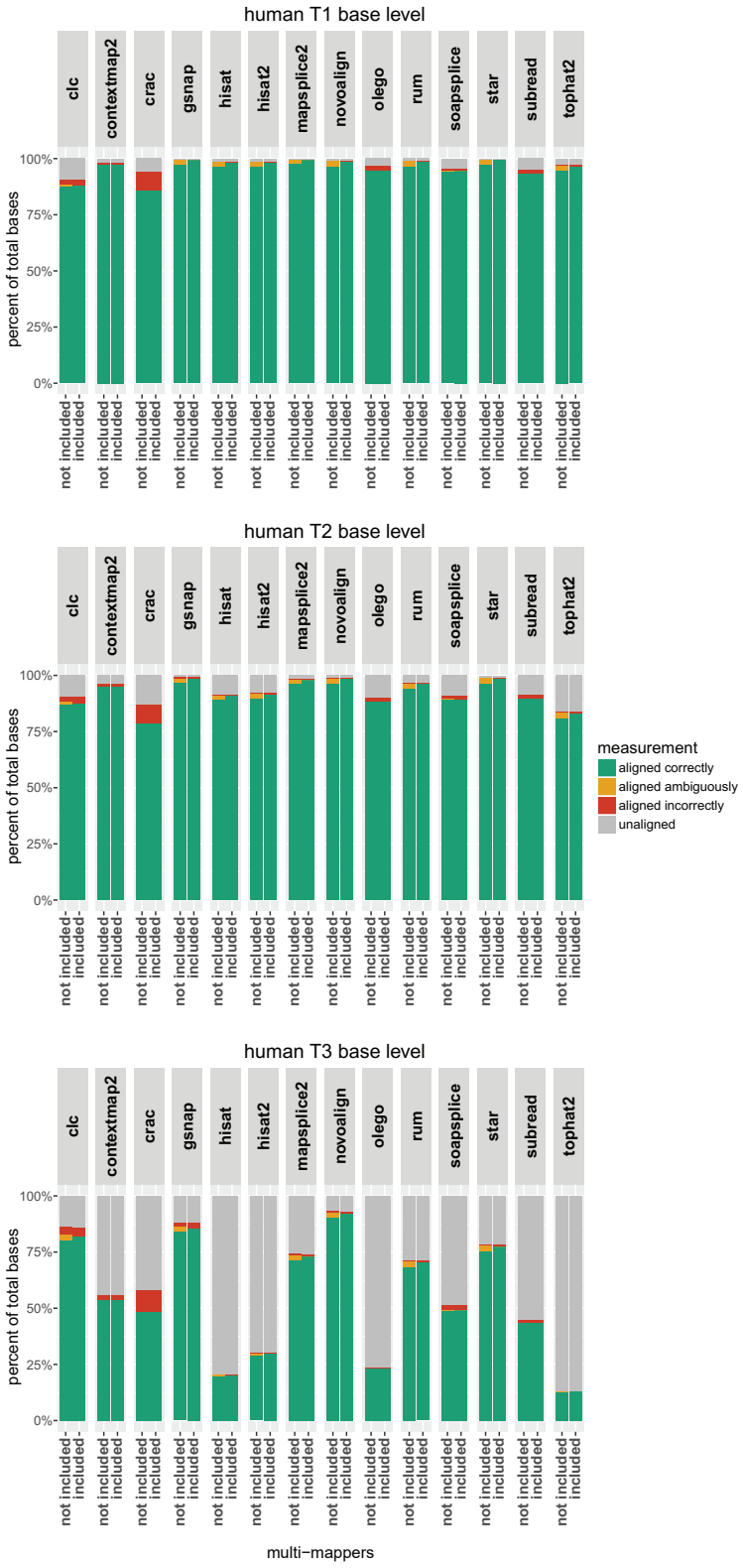

Fig. 5.13 *Multimapper analysis at the base level for the human datasets.* The bars show the percentage of bases aligned correctly, aligned ambiguously, aligned incorrectly and unaligned by each tool. The labels "not included" and "included" refer to considering the best multimapper as if it were aligned unambiguously and including it in the analysis.

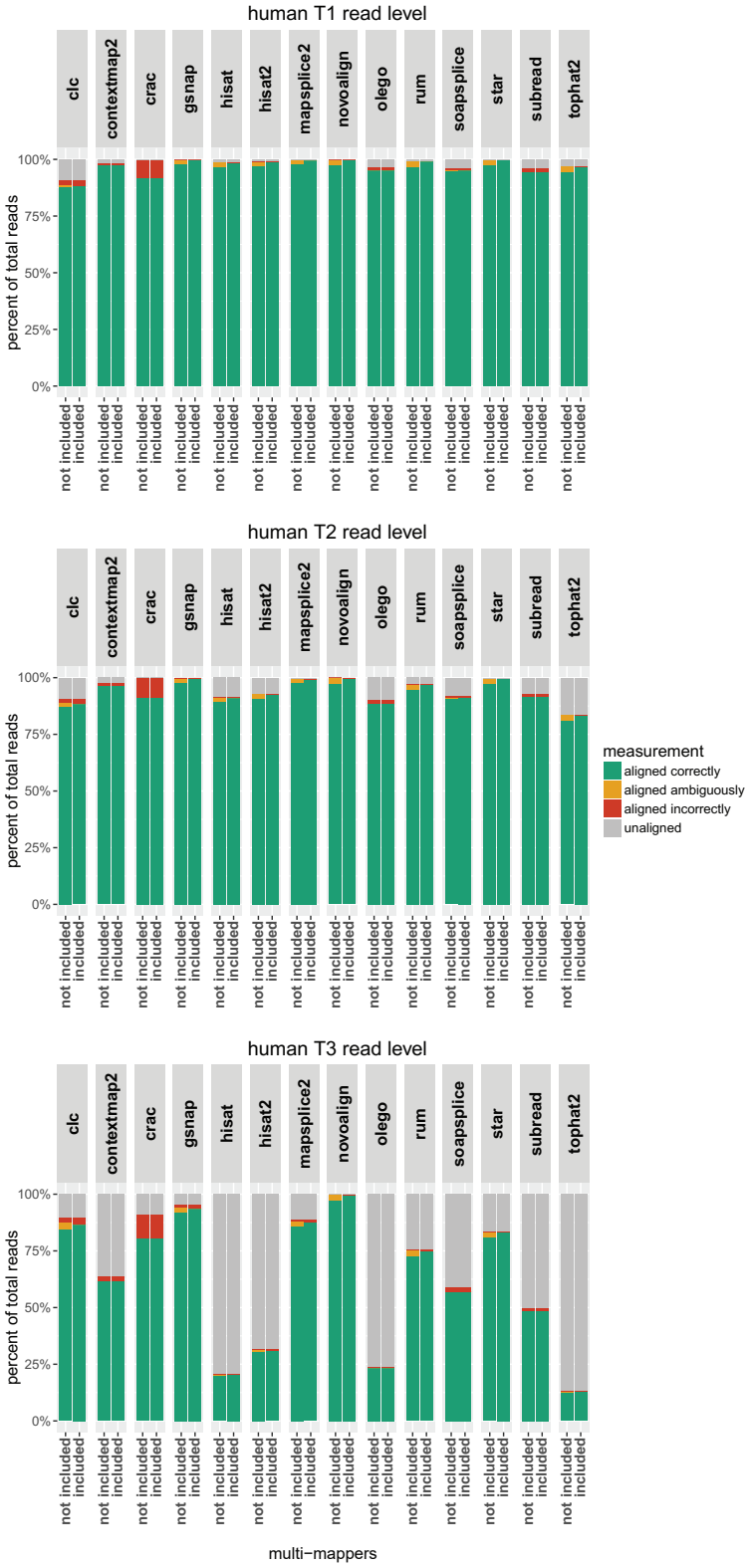

Fig. 5.14 *Multimapper analysis at the read level for the human datasets.* The bars show the percentage of reads aligned correctly, aligned ambiguously, aligned incorrectly and unaligned by each tool. The labels "not included" and "included" refer to considering the best multimapper as if it were aligned unambiguously and including it in the analysis.

5.7 Insertions and deletions

The identification of insertions and deletions represents one of the most challenging tasks in the alignment process. First, indel detection requires complex scoring functions and similarity/distance measures to be properly handled. For example, the commonly used Hamming distance does not manage indels properly, while the use of a scoring function requires the definition of a set of penalties for gap open, gap extension and gap close. Second, indel detection is computationally expensive since it requires time consuming backtracking and dynamic programming procedures. Last, the wide range of indel lengths (e.g. 1bp-10000bp in human [132, 133]) requires flexible procedures to correctly handle insertions and deletions of different sizes, even though large indels are usually easier to identify compared to small indels.

Despite these issues, the correct identification of indels is a primary goals in many sequencing studies. Indels have a major role in the onset and progression of several diseases [134–136] and are a major determinant of human biological diversity [137, 138]. The number of indels in human genomes is second only to the number of single nucleotide polymorphisms and one of the main challenge in their identification is the low alignment quality [139, 140].

For this reason, a specific set of statistics on insertion and deletion accuracy were collected at the base level. The results on human dataset are shown in Figure 5.15 and Figure 5.16.

Between insertions and deletions the tools show similar trends: algorithms showing poor accuracy with insertions tended to also have poor accuracy with deletions. Deletions seem easier to identify compared to insertions, even if the difference is small. In the less complex libraries T1 and T2, only half of the tools achieve an indel recall of at least 50% highlighting the current difficulties of many tool in handling this complex feature. As a consequence, in the most complex dataset T3 only CLC and Novoalign show a recall higher than 50% in both insertions and deletions identification. Among the tested tools, CLC and Novoalign followed by GSNAP and Subread show the best performance on insertion detection. As for deletion identification, the list of best performers is the same with also RUM that performs well.

Again, these results underline that the choice of the right aligner has a pivotal role in alignment accuracy, especially in a challenging task as indel detection. Unlike previous results, indel identification shows a considerable difference between tool performances even in the less complex dataset T1. In the same dataset T1, just a few tools achieve an indel recall higher than 75%, which is far from the base level recall achieved in the same dataset. As for previous analyses, P. falciparum datasets show a similar trend across libraries, achieving only slightly higher recalls compared to human.

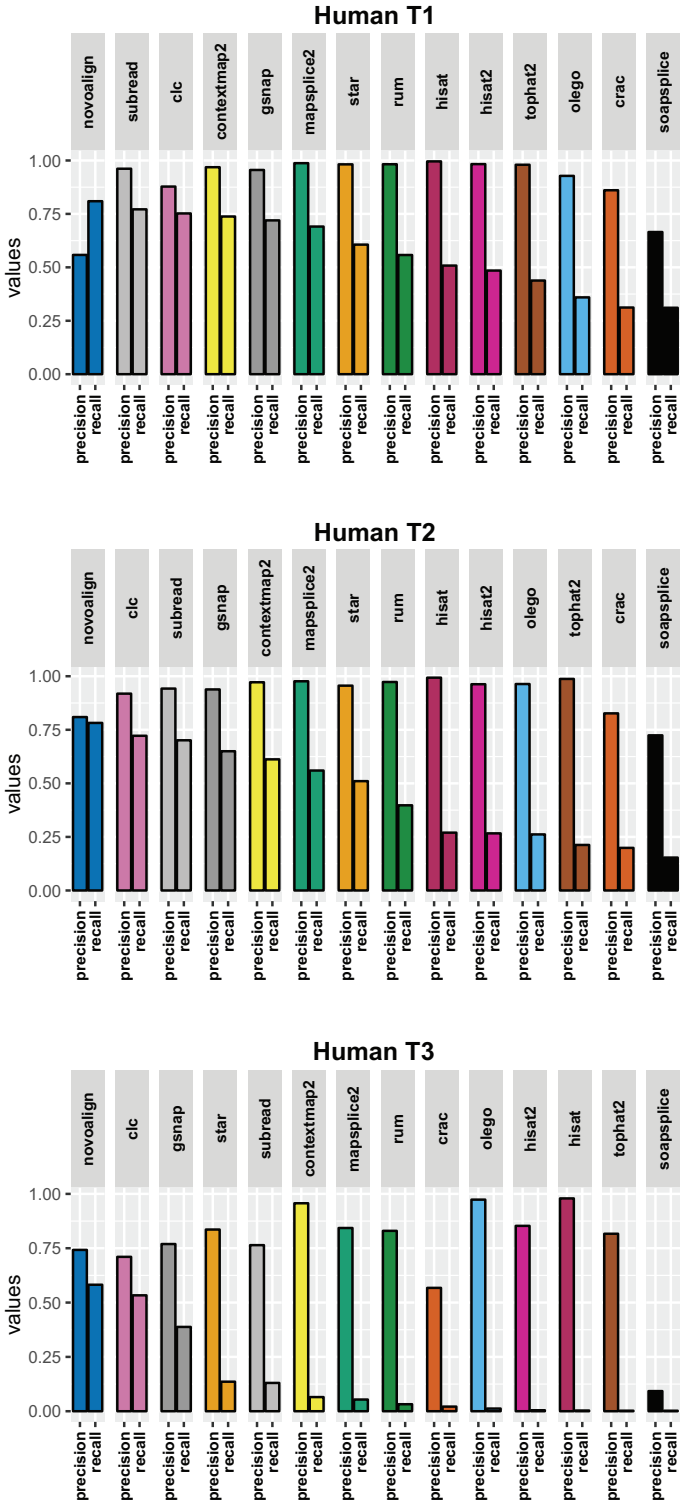

Fig. 5.15 *Insertion level precision and recall for the human datesets.* The tools are sorted by descending recall. For each tool, the figure shows the insertion precision and recall at the base level

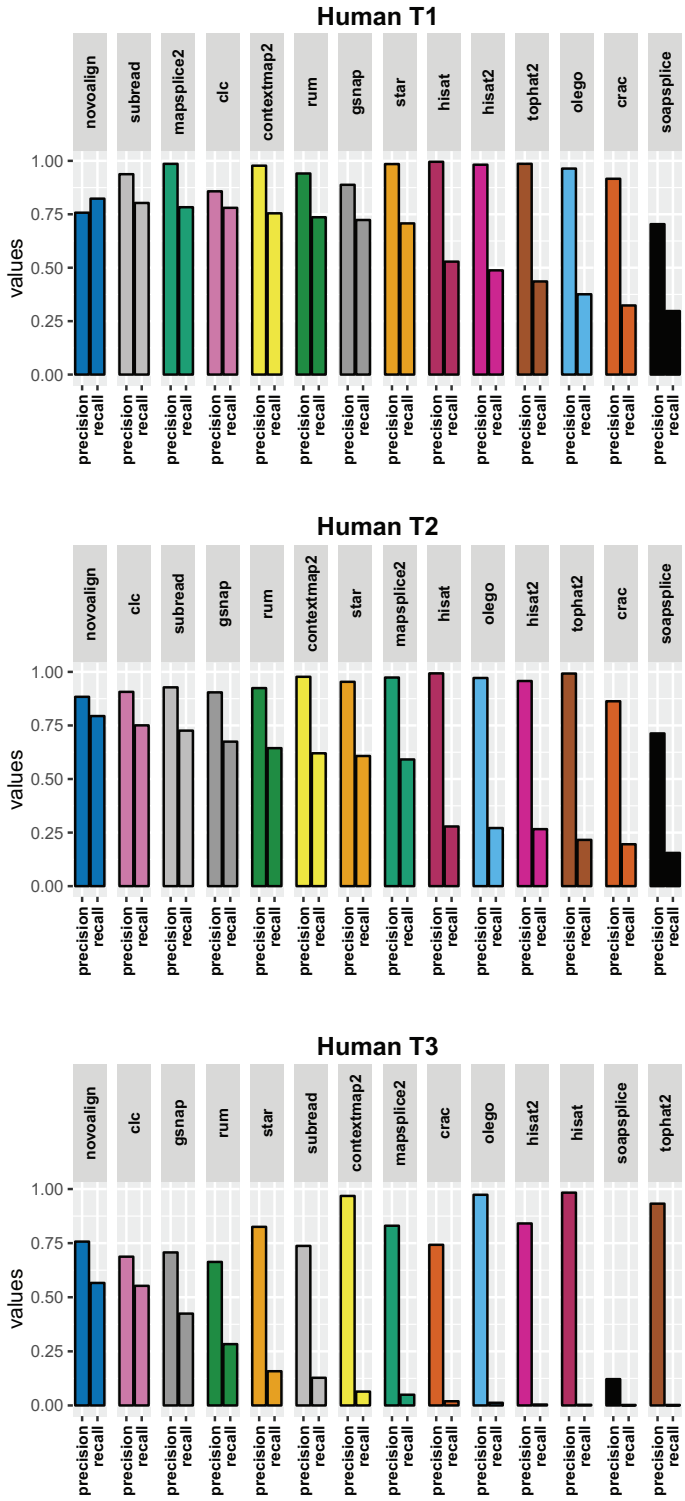

Fig. 5.16 *Deletion level precision and recall for the human datasets.* The tools are sorted by descending recall. For each tool, the figure shows the deletion precision and recall at the base level

5.8 Alignment speed and memory requirement

Aligning tens of millions of reads against a large genome is a computational expensive task which could require several GB of memory and long execution times even on a HPC machine. The aligners try to mitigate these issues by designing specific heuristics or approximations and exploiting efficient data structures and algorithms. However, each aligner's implementation results in a unique way to use the previous elements, resulting in different performances among tools. In this scenario, it is very important to identify both which are the most efficient tools and the minimum amount of memory and execution time required to achieve a specific level of accuracy. For this reason, the metrics defined in section 4.2.4 were employed to measure CPU time, execution time and memory usage on the default alignment described in section 5.2.

The tools were executed on a HPC cluster with a Platform Load Sharing Facility (LSF) job scheduling system. The cluster consists of 144 IBM iDataPlex Nodes (16 physical cores per node; 192 or 256 GB of RAM per node) using a Red Hat Linux 6.4. A job was designed for each alignment run and 16 threads were reserved for each job. The performance metrics were not computed for CLC since the tool works mainly with a graphic interface which makes the performance profiling in a cluster environment difficult to test.

The results on the human datasets are shown in Figure 5.17, Figure 5.18, and Figure 5.19. As for alignment accuracy, the different tools show a highly variable scenario both in terms of execution time and memory usage. The fastest tools on both human and P. falciparum are HISAT2, HISAT and Subread. STAR and Subread have comparable results on all libraries except for P. falciparum T2 and T3, where Subread is substantially faster than STAR. The slowest tools are Mapsplice2, RUM, ContextMap2, OLego and TopHat2. Novoalign has no multi-threading option in its free license version, so it is difficult to precisely assess its performance. Unlike previous analyses, the human and P. falciparum results differ sensibly in terms of execution times. Indeed, the different genome sizes result in faster execution time on P. falciparum datasets compared to human libraries. The fastest tools on human are consistently the best performers also on P. falciparum, while the performance of the other tools sometimes differs between the two organisms. In addition, the execution times increase in the most complex datasets for all tools due to the additional effort required by these libraries. The CPU time metrics underline how the different tools exploit parallelism. Tools having a run time close to the CPU time (divided by 16, as in figure 5.18) highlight a better use of parallelism in terms of work load distribution. The different tool rankings between CPU time and Run time performance are due to this parallelism ability. Interestingly, OLego and ContextMap2 show often a considerable variability on both CPU time and Run

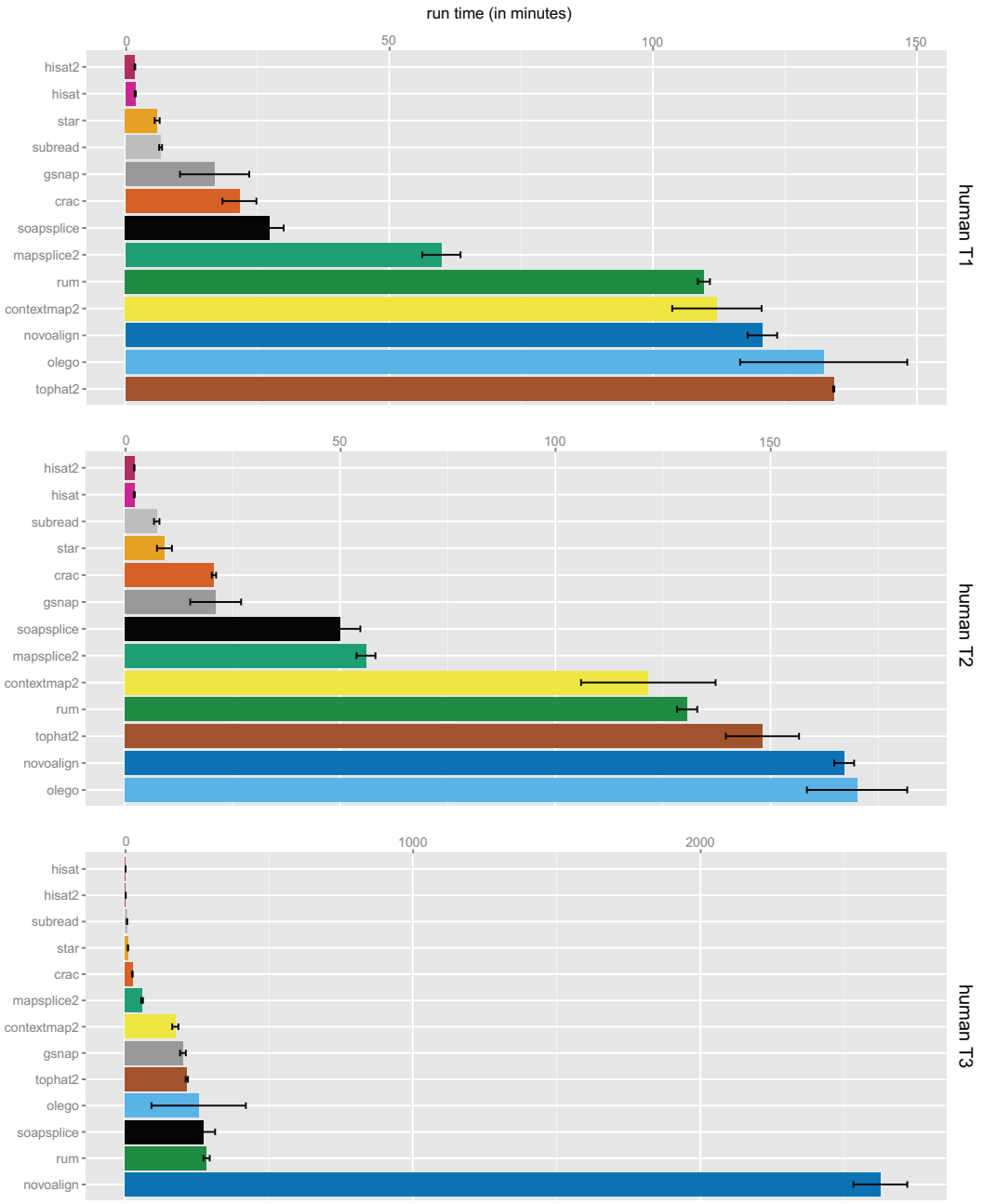

Fig. 5.17 *Performance in terms of run time for the human datasets.* For each dataset and tool, the bars show the average run time (in minutes) computed on the three replicates. The error bars show the variability of this measure. The tools are sorted by ascending average run time. Note: Novoalign has no multithreading in its free license versions. In order to obtain comparable results, the Novoalign run time was divided by the number of used threads (16). However, this is only an underestimation since the real scalability could be lower than the ideal one.

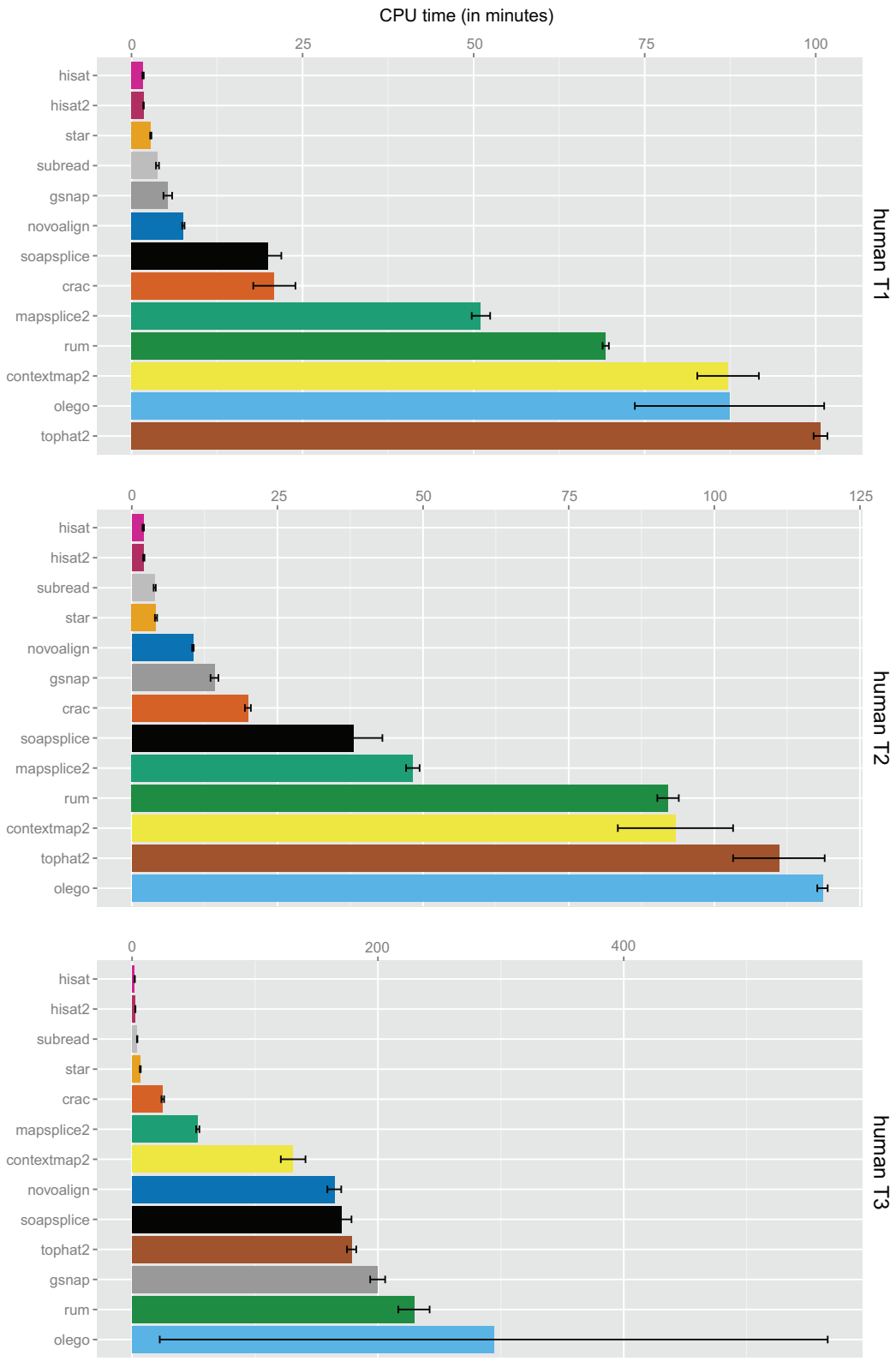

Fig. 5.18 *Performance in terms of CPU time for the human datasets.* For each tool, the figures show the real CPU time divided by the number of threads used (16). For each dataset and tool, the bars show the average CPU time (in minutes) computed on the three replicates. The error bars show the variability of this measure. The tools are sorted by ascending average CPU time.

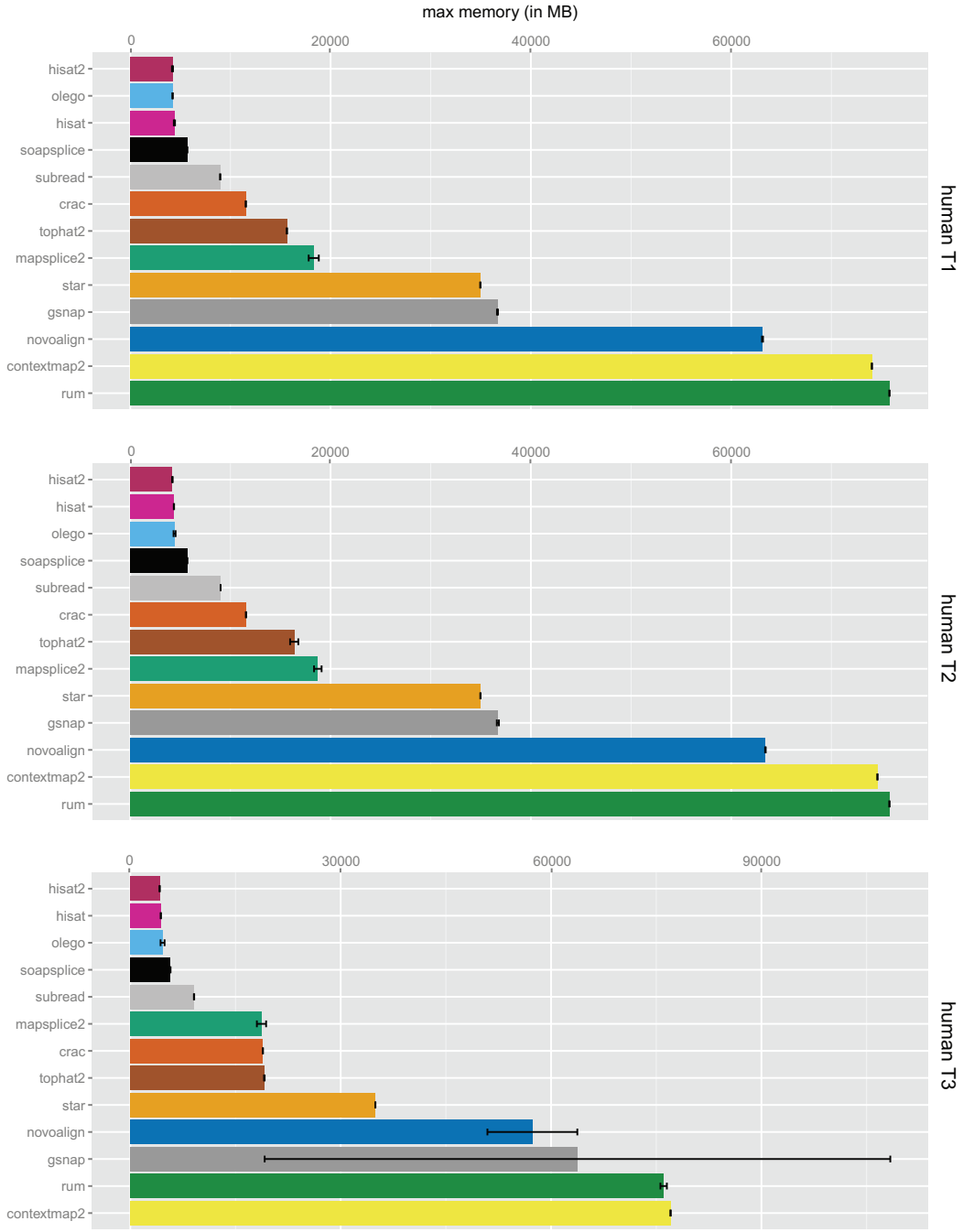

Fig. 5.19 *Performance in terms of maximum memory usage for the human datasets.* For each dataset and tool, the bars show the average maximum memory usage (in MB) computed on the three replicates. The error bars show the variability of this measure. The tools are sorted by ascending average memory usage.

time. This variability was confirmed by additional tests, underling a high sensibility to the particular input dataset.

Finally, identifying a trend between alignment accuracy and alignment speed seems to be non-trivial. Tools achieving a low alignment accuracy at the base, read and junction level show both high alignment speed (HISAT2, HISAT) and long execution time (OLego). On the other hand, STAR is the fastest among the most accurate tools and achieved an excellent alignment speed even compared to less accurate tools. GSNAP and Novoalign are usually slower than STAR, especially on the human datasets.

As for maximum memory requirements, HISAT2, HISAT, OLego and SOAPsplice are consistently the most efficient tools across organisms and complexity levels. ContextMap2 and RUM show the highest requirements in terms of memory in human libraries, followed by Novoalign and GSNAP. In the P. falciparum datesets, ContextMap2 is still the worst performer, followed by Novoalign, RUM and MapSplice2. As for execution time, the P. falciparum datasets show lower requirements even in terms of maximum memory usage. Since the amount of memory employed during the alignment process is mainly affected by the size of the genome index, the lower length of P. falciparum genome compared to human results in a considerable reduction of the required amount of RAM. Interestingly, the most efficient tools in terms of memory (HISAT2, HISAT and OLego) are among the worst performers in terms of accuracy at the base, read and junction level. On the other hand, some of the most accurate tools (STAR, GSNAP and Novoalign) have considerable memory requirements.

Chapter 6

Comparative analysis of splice unaware alignment methods

Splice unaware algorithms solve an easier problem compared to splice aware methods, mainly since no splicing detection is required. For this reason, these algorithms result in fewer alignment parameters and implemented features, compared to splice aware tools.

Unlike splice aware methods, the class of splice unaware aligners is well studied and many comparison analyses have been performed [99, 126, 141–146]. For this reason, the main goal of the following comparative analysis is to assess and update existing results, rather than perform a thorough study of all the alignment aspects.

In this chapter, the simulated libraries described in section 4.1.3 were employed to assess the accuracy and efficiency of the 4 popular splice unaware alignment methods described in section 3.3.2: Bowtie, Bowtie2, BWA and BWAMEM.

6.1 Tested scenarios

The low number of implemented features and the large number of tests already performed in the literature result in a limited number of metrics and analyses compared to splice aware methods. In this scenario, two main aspects were investigated in the study of splice unaware methods:

- **Alignment using the default parameters**: as for splice aware methods, the common approach to read mapping consists in running the tools with the default parameters. Compared to splice aware methods, there are less alignment parameters and their tweaking is rarely necessary. The accuracy of the different tools were assessed in terms of precision and recall using the base level and read level metrics.
- **Alignment speed and memory requirement**: unspliced alignments are less computationally expensive than spliced alignments, both for the lack of splicing junction detection and for the less complex alignment strategies implemented. However, the alignment of several millions of reads could still require many hours and GBs of RAM. To measure the amount of resources employed by each tool, the metrics described in section 4.2.4 were employed to assess the global run time, the single CPU time and the maximum amount of RAM memory.

In the following sections these two analyses are described and the most relevant results are presented and discussed. The details about the alignment parameters are described in Appendix D.

6.2 Results using the aligner tools default parameters

For each tool, the alignment command was designed starting from the default settings. Then, specific options to increase the quality of the alignment, usually at the cost of longer execution time and memory requirement, were set if available. Unlike splice aware tools, none of the tested methods have performance parameters to optimize execution time against memory usage.

6.2.1 Base level

At the base level, each base is considered as aligned correctly, aligned incorrectly or unaligned. Unlike previous analyses, no multireads were reported by the tools using the default parameters. Base level results are shown in Figure 6.1 and Figure 6.2.

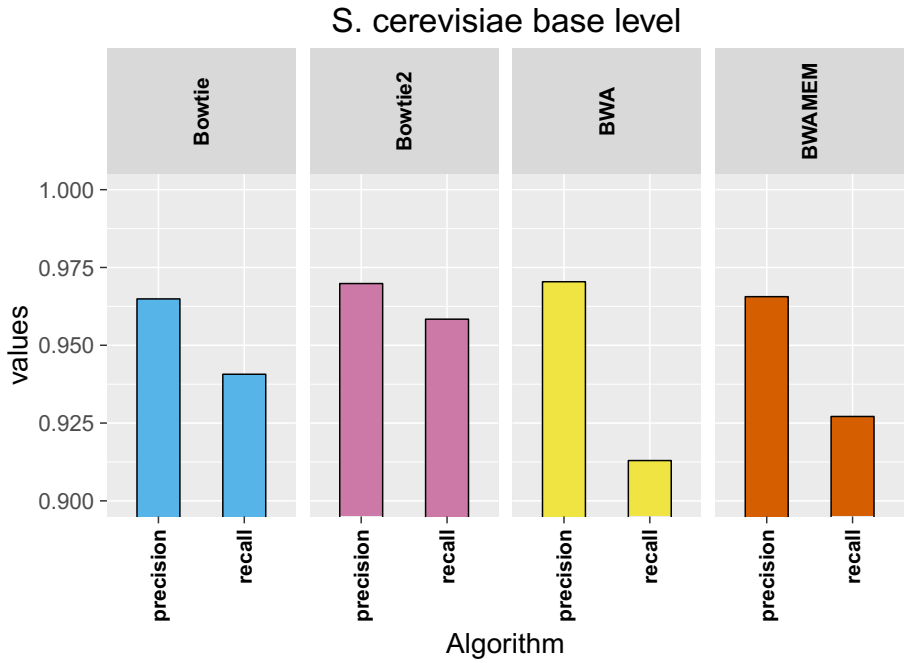

Fig. 6.1 *Default parameters - Base level precision and recall for the S. cerevisiae datasets.* The chart shows the average precision and recall across replicates; the SD across replicates is lower than 0.2%. Since the tools achieve comparable performance, the y-axis is scaled between the values 0.9 and 1.

At the base level, the mapping strategies of Bowtie and Bowtie2 seem to slightly outperform the BWA and BWAMEM ones. The results show comparable performance in terms of precision, while the recall shows some minor differences between tools. Bowtie2 achieves the best performance with a precision of ~97% and a recall of ~96%, followed by Bowtie

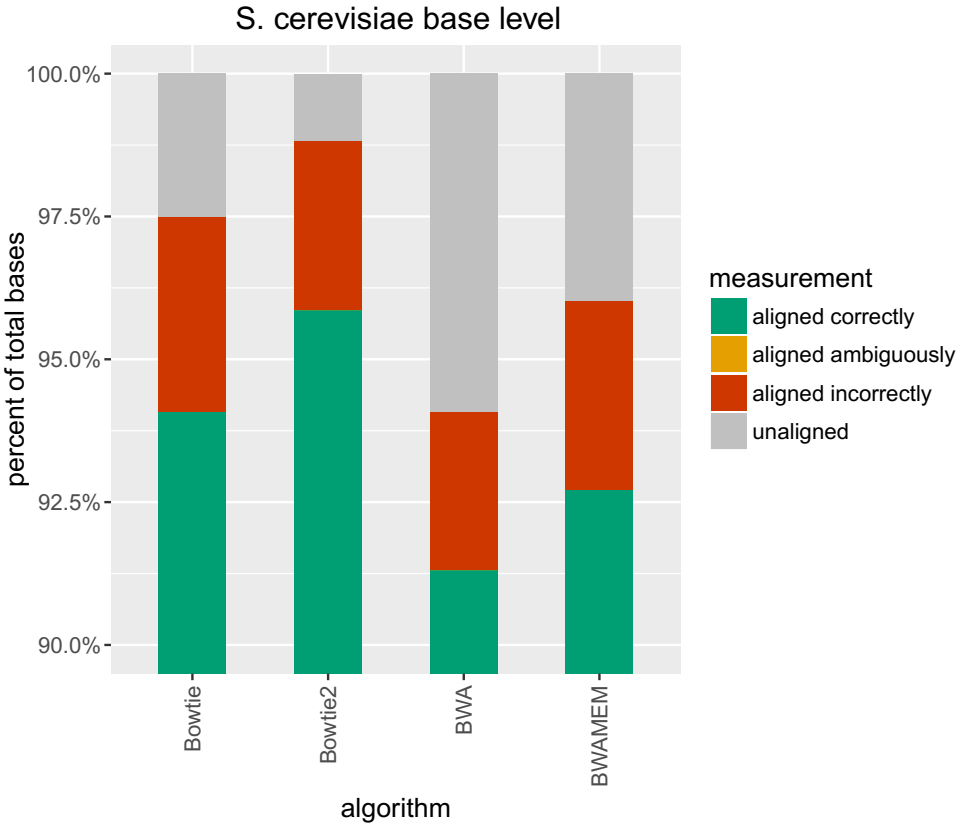

Fig. 6.2 *Default parameters - Base level statistics for the S. cerevisiae datasets.* The bars show the percentage of bases aligned correctly, aligned incorrectly and unaligned by each tool. The bars represent the average percentages across replicates; the SD across replicates is lower than 0.2%. Since the tools achieve comparable performance, the y-axis is scaled between the values 0.9 and 1.

(respectively, ~96% and ~94%) and BWAMEM(~96% and ~93%). A similar trend is shown by the percentage of mapped bases where BWA and Bowtie achieve the lowest percentage of incorrectly mapped bases.

6.2.2 Read level

Since quantification on simple organism is often performed counting the number of reads overlapping an exon, a poor performance in these metrics would result in inaccurate inferences of the expression levels. Read level results are shown in Figure 6.3 and Figure 6.4.

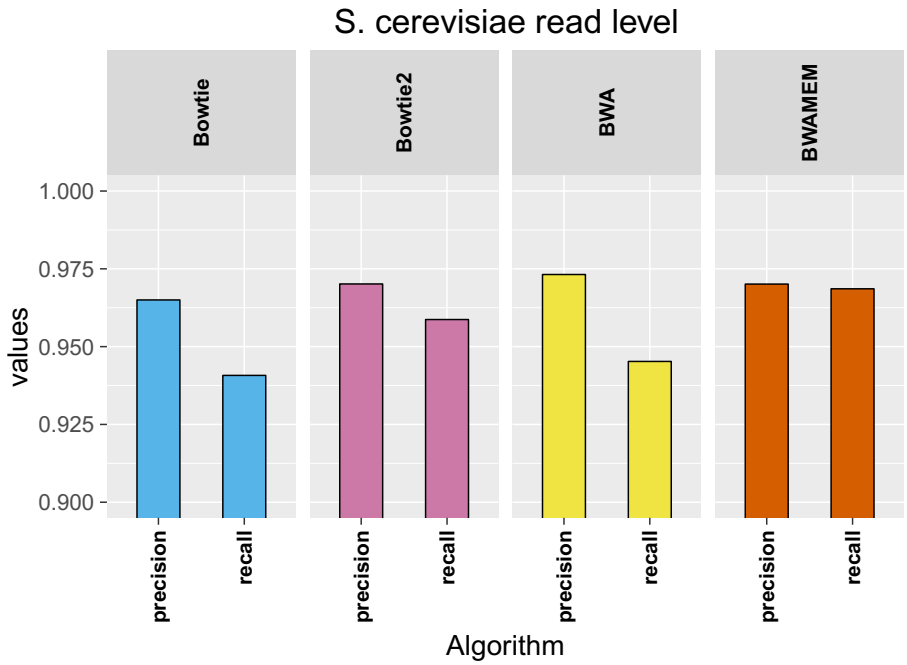

Fig. 6.3 *Default parameters - Read level precision and recall for the S. cerevisiae datasets.* The chart shows the average precision and recall across replicates; the SD across replicates is lower than 0.2%. Since the tools achieve comparable performance, the y-axis is scaled between the values 0.9 and 1.

As for base level, the tools show very good accuracy both in terms of precision and recall. While precision shows a more uniform behavior, it is again the recall that results in a more variable trend. The best tradeoff between precision and recall is achieved by BWAMEM, followed by Bowtie2 and BWA. BWAMEM and Bowtie2 show also the highest number of aligned reads, while BWA achieves the lowest percentage of incorrectly mapped reads.

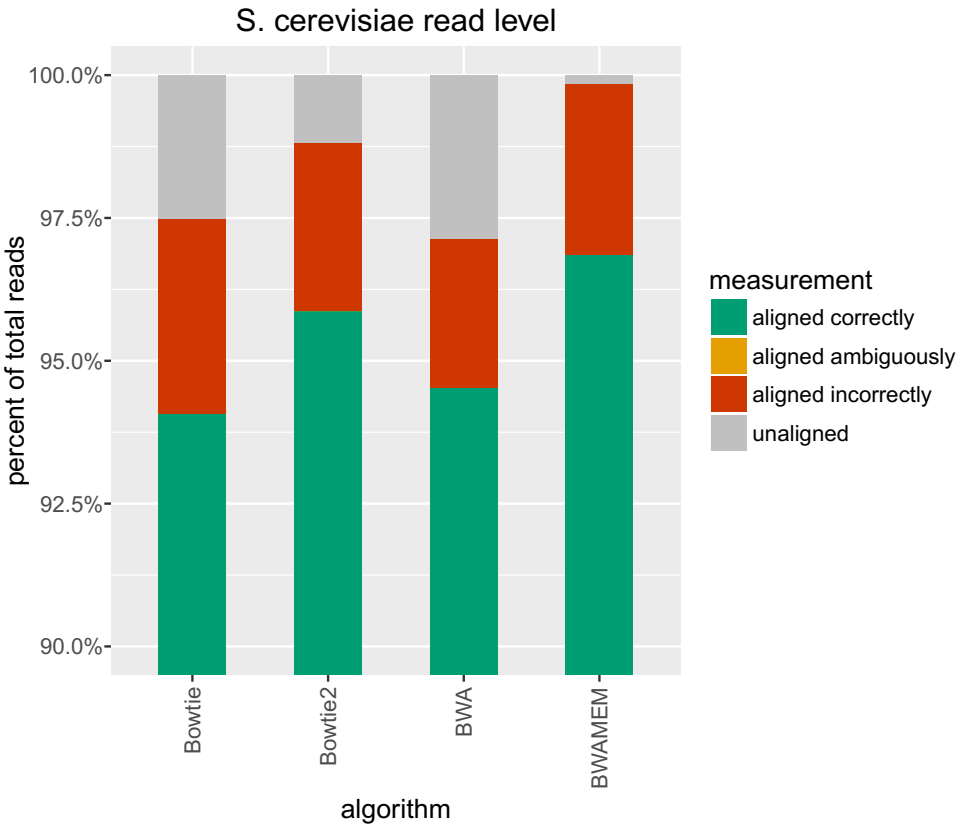

Fig. 6.4 *Default parameters - Read level statistics for the S. cerevisiae datasets.* The bars show the percentage of reads aligned correctly, aligned incorrectly and unaligned by each tool. The bars represent the average percentages across replicates; the SD across replicates is lower than 0.2%. Since the tools achieve comparable performance, the y-axis is scaled between the values 0.9 and 1.

6.3 Alignment speed and memory requirement

Splice unaware tools have usually lower requirements in terms of memory usage compared to splice aware tools. As for the execution time, even though splice unaware tools require less computation due to the easier scenario (i.e. no spliced alignments), the high degree of efficiency achieved in recent years by the splice aware algorithm makes the execution time comparable for some tools. In this scenario, it is important to identify the differences in resource usages since one of the advantages of using splice unaware tools on simple organisms is the lower resource requirements. The performance in terms of alignment speed and memory requirement are assessed measuring the CPU time, run time and memory usage (section 4.2.4).

The tools were executed on a HPC cluster with a Sun Grid Engine (SGE) job scheduling system. The cluster consists of 2 nodes each equipped with four sixteen-core AMD Opteron 6378 (64 core in total) and 256 GB RAM, using a Fedora Linux 20. A job was designed for each alignment run and 4 threads were reserved for each job. The resource profiling results are shown in Figure 6.5, Figure 6.6, and Figure 6.7.

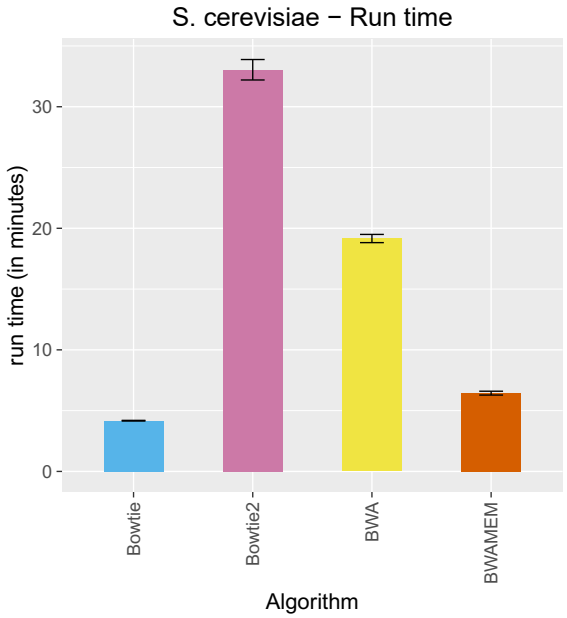

Fig. 6.5 *Performance in terms of run time for the S. cerevisiae datasets.* For each tool, the bars show the average run time (in minutes) computed on the six replicates. The error bars show the variability of this measure.

Bowtie achieves the lowest execution time, showing a run time almost 7 times faster than Bowtie2. BWAMEM is the second best performer, followed by BWA and Bowtie2. The same trend is observed also in CPU time measurements, where all the tools show also a good

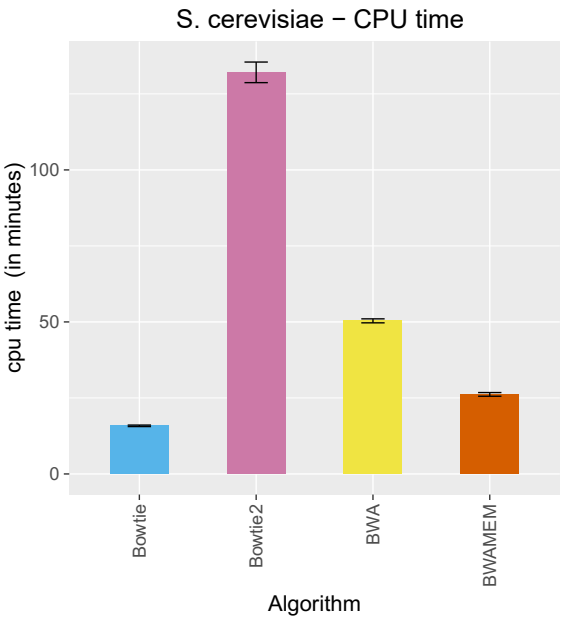

Fig. 6.6 *Performance in terms of CPU time for the S. cerevisiae datasets.* For each dataset and tool, the bars show the average CPU time (in minutes) computed on the six replicates. The error bars show the variability of this measure.

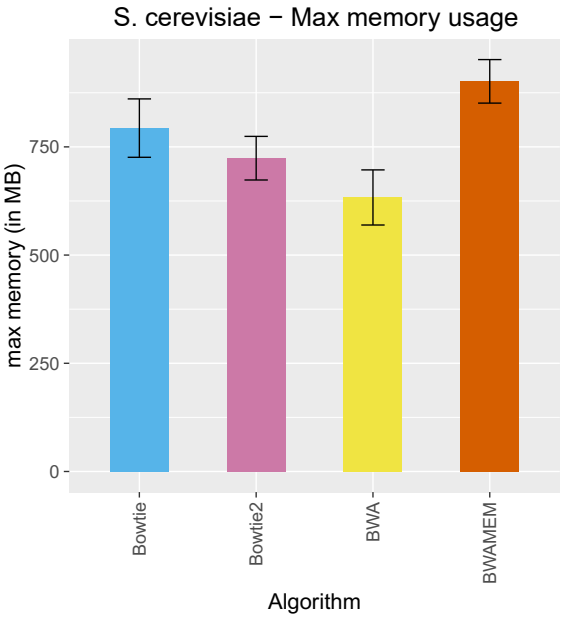

Fig. 6.7 *Performance in terms of maximum memory usage for the S. cerevisiae datasets.* For each dataset and tool, the bars show the average maximum memory usage (in MB) computed on the six replicates. The error bars show the variability of this measure.

level of parallelism. Indeed, except for BWA, the CPU time achieved by each tools is almost 4 times the run time (4 is the number of used threads), highlighting a good scalability in the parallel computation. Comparing the CPU time of splice unaware tools to the CPU time of splice aware methods on the P. falciparum datasets highlights some interesting facts. Even on a bigger genome (the P. falciparum genome is ~23.3Mb while the S. cerevisiae genome is ~12.5Mb), some splice aware tools are able to achieve better than or comparable CPU time compared to the tested splice unaware algorithms. Especially comparing the performance on the less complex P. falciparum libraries T1 and T2, the use of a splice unaware method instead of a splice aware tool seems to not bring any benefit in terms of computational time. As for the memory usage, the four tools show some differences in the required amount of RAM. BWAMEM has the highest memory requirement, employing ~30% more memory compared to BWA. On the other hand, Bowtie2 shows a lower memory requirement compared to its previous version. Comparing to splice aware methods, in the least complex P. falciparum library the great majority of the tools require more than 2GB of RAM to handle a 23.3Mb genome, while splice unaware tools use less than 900MB of RAM to work on a 12.5Mb genome. However, a straightforward comparison with splice aware tools is more complex here, since the different genome sizes have a direct effect on the amount of memory used. In addition, the different indexing (i.e. BWT vs. Hash/SA) and compression strategies employed by the splice aware algorithms make it difficult to compare the two classes of methods.

Chapter 7

Effects of read alignment on expression level quantification

The results of the aligner comparison highlight several differences between tool accuracies, especially among splice aware tools. The performance differences, particularly on more complex datasets, would considerably affect the accuracy of many downstream analyses. On the other hand, even if Bowtie2 and BWAMEM outperform their previous versions, splice unaware tools show smaller differences in terms of precision and recall. For these tools, the different performances could bring only minor effects on the reliability of downstream analyses.

In order to assess the effect of the four splice unaware tools on a common downstream analysis, an expression level quantification study was performed. Expression level quantification is one of the most common downstream analyses in RNA-Seq studies and it is described in section 2.3.2. Briefly, quantification analyses exploit the idea that the number of reads that map to each transcribed sequence (called *counts*) is a good proxy of its expression level. Quantification could be performed at the transcript, gene or exon level, depending on which coding unit is employed to summarize the raw counts of mapped reads. After some processing and normalization, counts provide a digital measure of expression levels.

In this chapter, a set of suitable test datasets for an expression level analysis is identified and described, focusing on real or simulated data which could ensure some sort of ground truth to exploit during the assessment. Second, a set of metrics to assess the expression level quantification accuracy is introduced. Finally, several data processing workflows are defined, including different read preprocessing options, alignment tools and alignment strategies. These different workflows allow a better understanding of the role of each data processing choice in the final quantification accuracy. In the last section, quantification accuracy results are shown and discussed.

7.1 Selection of test datasets

In order to assess quantification accuracy, some ground truth information about the true expression level in the sample is required. These true abundance values would be compared to the measured expression levels and an accuracy index would be produced.

The easy way to obtain a ground truth information about the true expression level is simulating data. Using the right simulator, it is possible to obtain a simulated read library and the expression levels employed during the simulation. The main advantage of employing simulated data is the complete control on data characteristics and the availability of complete ground truth information. Unlike simulation for alignment benchmarking, here some complex effects (e.g. length bias, GC content bias, etc.) must be modeled in order to produce realistic data. However, simulating the entire range of biases which affect real data is challenging and usually simulators employ approximations or simplifications in the data generation process.

On the other hand, using real data makes it impossible to know the ground truth information about the level of expression. Quantitative Real-Time PCR (qPCR) [147] is commonly employed to assess the fold differences in expression levels, providing an indirect relative abundance measure [148–150]. Alternatively, standard transcripts with known sequences and concentrations, called *spike-ins* [151], could be added to sample. Spike-ins try to mimic some properties of the endogenous transcripts, such as different GC content and length, maintaining at the same time a minimal sequence homology with the transcripts in the samples. This characteristic allows minimization of the confounding alignment of spike-in reads to the reference genome, resulting in a more accurate detection of spike-in fragments. Unlike simulated data, the use of spike-in techniques allows accounting of the entire complexity of real data but provides a less accurate estimation of the true expression level.

In order to exploit the strengths of the two class of approaches, both real data containing spike-ins and simulated data were employed in the assessment process.

7.1.1 Real data

Real data consist of 6 libraries of *Mycobacterium tuberculosis* H37RV sequenced using an Illumina HiSeq 1000 platform. An Illumina Truseq stranded mRNA preparation kit with a Ribo-Zero rRNA Removal Kit (Bacteria) was employed during library preparation. The six libraries consist of 100bp strand-specific paired end reads; each library contains between ~30 and ~45 millions read pairs. Before sequencing, two mixtures of Ambion ERCC Spike-In Control Mixes (Thermo Fisher Scientific [152]) were added to samples. The ERCC RNA Spike-In Control Mixes are pre-formulated sets of 92 polyadenylated transcripts from the ERCC plasmid reference library. The two mixtures, called *Mix1* and *Mix2*, contain 92

transcripts spanning a 10^6-fold concentration range (Figure 7.1) and having different GC content and lengths (Figure 7.2).

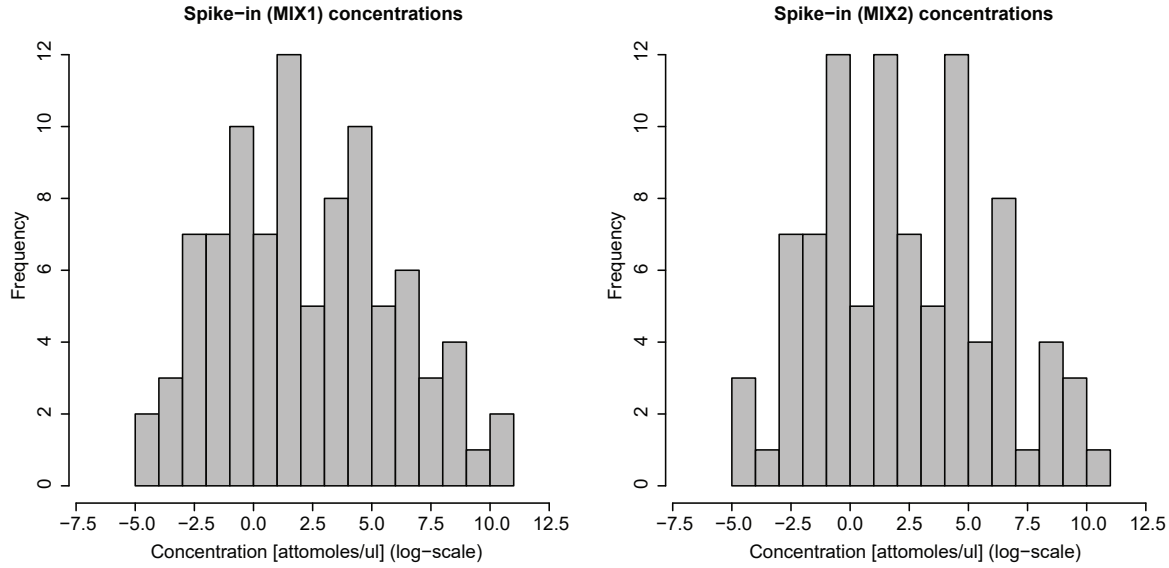

Fig. 7.1 *Spike-in concentrations*

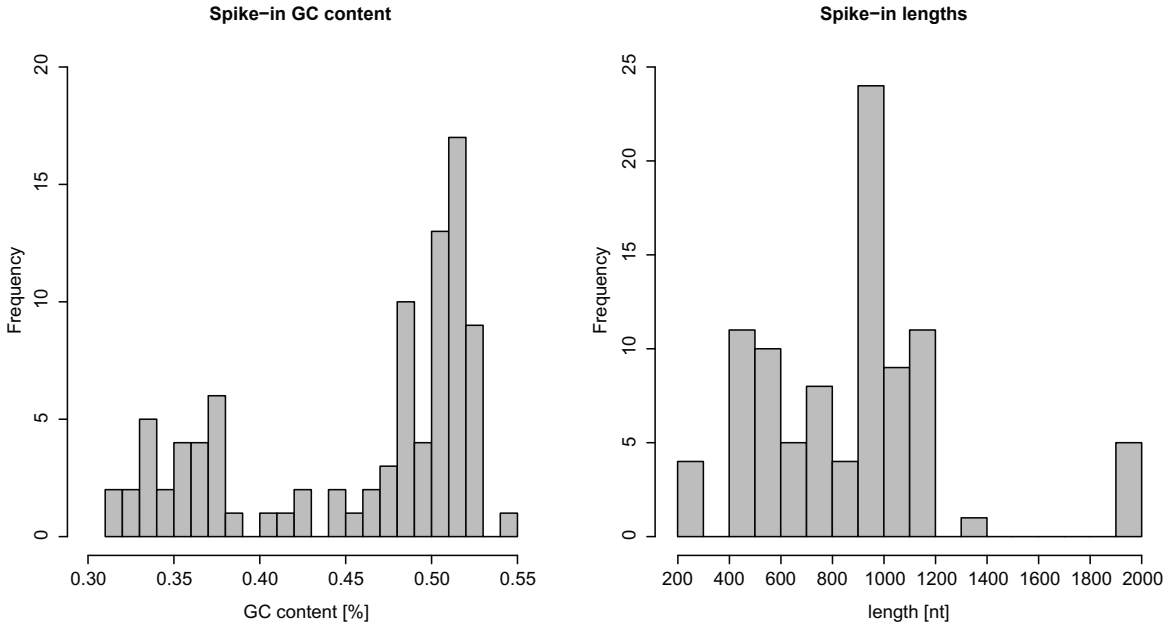

Fig. 7.2 *Spike-in GC content (left) and lengths (right)*

The known spike-in concentration levels were employed as ground truth in the assessment.

7.1.2 Simulated data

A subset of the simulated data used in the benchmarking of splice unaware mapping methods (section 4.1.3) were employed in the current assessment. The dataset consists of 6 S. cerevisiae libraries of ~1 million 100 base paired-end strand-specific reads simulated using the Flux simulator. The use of the S. cerevisiae genome results in only ~0.24% of reads coming from exon junctions.

The Flux Simulator assigns to the transcripts specified in the annotation random expression ranks x which are then converted into relative expression levels y by the following modified Zipf's Law:

$$y = y_0 x^k exp^{-\frac{x}{a}-(\frac{x}{b})^2}$$

where y_0 is the initial number of molecules, x denotes the rank number of a gene, $k < 0$ is the exponent of the intrinsic power law, while a and b control the exponential decay. The number of expressed RNA molecules provided by the simulator was employed as ground truth in the assessment. Data were generated using *in silico* fragmentation and first strand synthesis with random primers. Finally, a size selection step was simulated, employing a normal distribution with mean equal to 300 and standard deviation equal to 50 as probability distribution over fragment sizes to be retained in the final set. In the simulation of the sequencing process, errors in the reads were added using the built-in simulator error model, which employs a Markov model to generate an error profile similar to the one observed in Illumina data.

7.2 Metric definitions

The expression level quantification accuracy was assessed in terms of Pearson coefficient of correlation r. Correlation is widely employed as accuracy metrics in studies involving the comparison between the estimates and the known expression levels [151, 153, 154].

However, correlation alone is not completely informative since is not able to capture effects such as data dispersion. For this reason, the relative error between the inferred expression level and the true expression level was computed. Since the inferred and true expression levels could be expressed in different scales, the values were expressed in terms of fraction over the total amount (e.g. a spike-in concentration value would be divided by the sum of all the concentration values) and the relative error was computed using such computed values.

In addition, a visual inspection of the plots representing the relation between true and inferred expression values was performed to identify possible undetected effects.

7.3 Quality control and read preprocessing

In order to identify the role of read quality control and preprocessing in the final quantification accuracy, both raw and preprocessed reads were analyzed. Read quality control was performed using FASTQC. The quality reports highlight the presence of adapter sequences in the real data and a low quality tail in the read sequence, especially in the simulated data (Figure 7.3 and Figure 7.4).

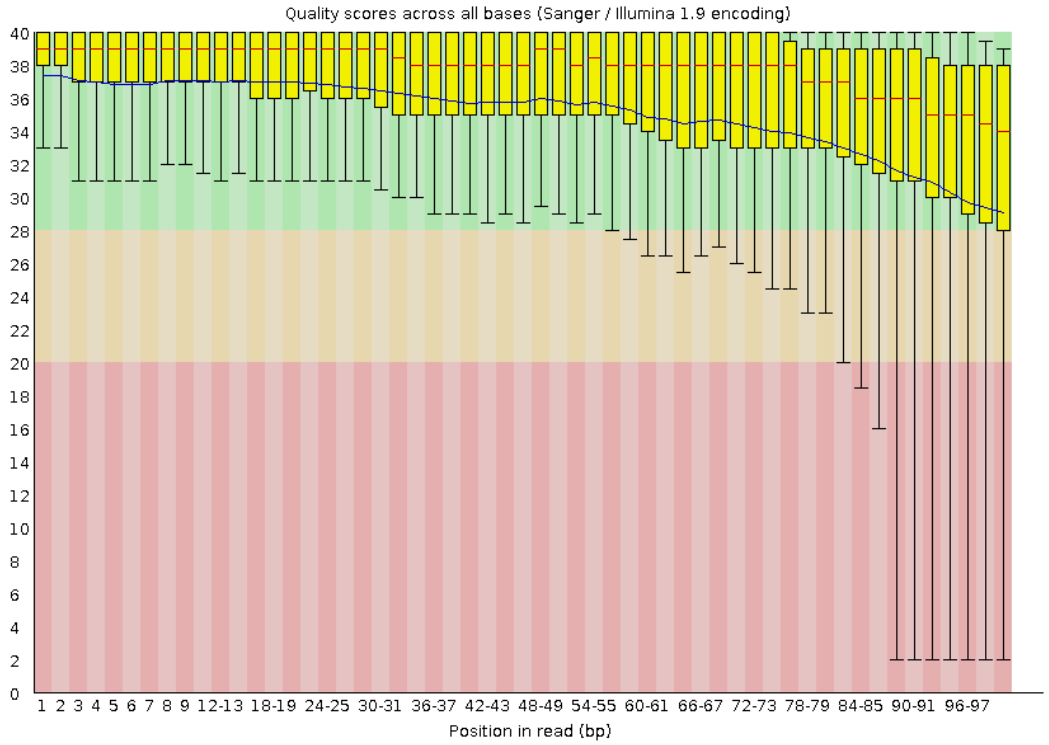

Fig. 7.3 *Base quality distribution for simulated data (library 1).*

The overall base qualities are higher in real data than in simulated data, suggesting that the built-in Flux Simulator error model generates a considerable amount of low quality bases.

In order to improve the quality of the data, some read preprocessing steps were performed using FASTX-Toolkit:

1. **Adapters removal:** adapter sequences at the end of the reads were removed

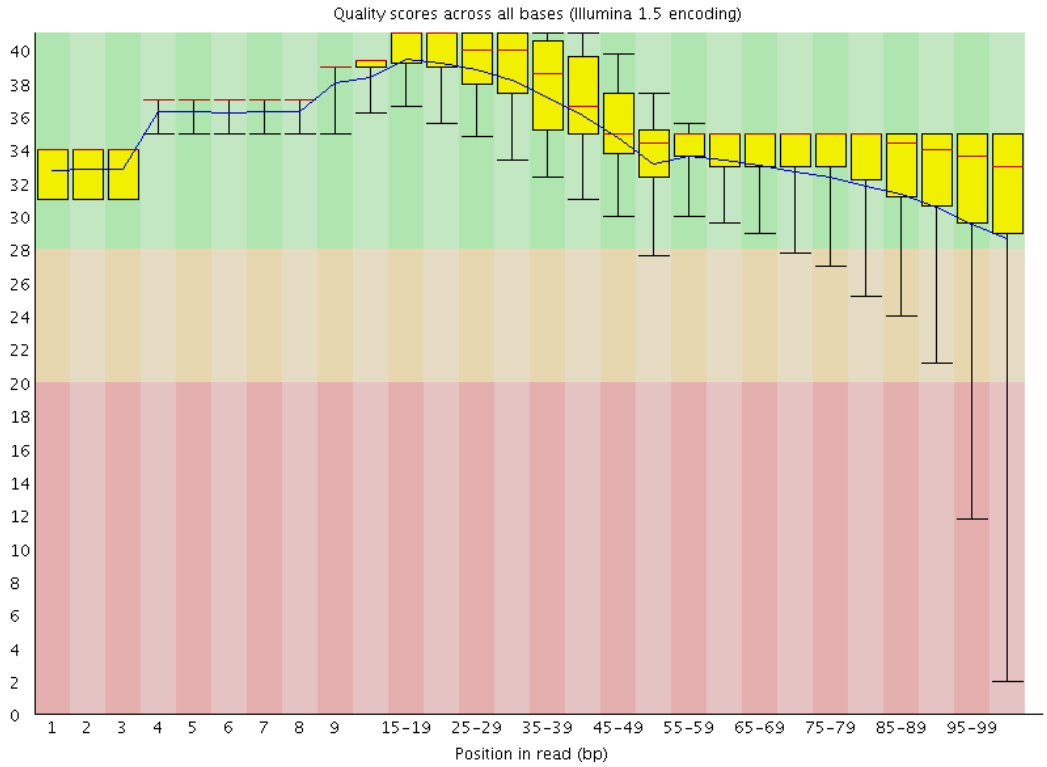

Fig. 7.4 *Base quality distribution for real data (library 1).*

2. **Low quality read tails trimming:** nucleotides showing a quality value lower than 20 were trimmed (from the end of the sequence).

3. **Short read discarding:** reads whose length after trimming was lower than 70 bases were discarded

After these steps, some read pairs lost one of the two mates resulting in several single read sequences (i.e. *singleton*). For each library, singletons were collected in an additional file and exploited during the mapping process. The results of read preprocessing are summarized in Table 7.1 and Table 7.2. The considerable number of low quality bases in the simulated dataset resulted in a high number of discarded reads (~21.8% read pairs lose one mate). On the other hand, the higher quality of real data allowed more than 99.7% of read pairs to pass the preprocessing thresholds.

Table 7.1 Effect of preprocessing on simulated dataset

Dataset	Read pairs	Preproc. read pairs		Removed read pairs		Singleton reads	
		Number	%	Number	%	Number	%
Sim1	1000200	767213	76.71%	15346	1.53%	217641	21.76%
Sim2	1000442	767100	76.68%	15593	1.56%	217749	21.77%
Sim3	1000938	766080	76.54%	15550	1.55%	219308	21.91%
Sim4	1001010	767282	76.65%	15579	1.56%	218149	21.79%
Sim5	1000133	765480	76.54%	15729	1.57%	218924	21.89%
Sim6	1000741	767137	76.66%	15463	1.55%	218141	21.80%

Table 7.2 Effect of preprocessing on real dataset

Dataset	Read pairs	Preproc. read pairs		Removed read pairs		Singleton reads	
		Number	%	Number	%	Number	%
Real1	34978376	34892984	99.76%	796	0.002%	84596	0.24%
Real2	35830466	35756662	99.79%	501	0.001%	73303	0.20%
Real3	40779711	40672542	99.74%	1011	0.002%	106158	0.26%
Real4	30027891	29962557	99.78%	866	0.003%	64468	0.21%
Real5	30835876	30771421	99.79%	404	0.001%	64051	0.21%
Real6	44947217	44836985	99.75%	17323	0.039%	92909	0.21%

7.4 Read mapping

With the aim of assessing the role of read mapping on quantification accuracy, both the raw and preprocessed reads were aligned using several splice unaware tools. The tools involved in section 6.1 comparisons (BWA, BWAMEM, Bowtie and Bowtie2) were employed in this test.

In addition to different alignment methods, the effect of different alignment strategies involving single-end (SE) and paired-end (PE) mapping modes was analyzed. Indeed, the tested tools allow mapping of both single-end and paired-end reads.

Moreover, specific alignment strategies were developed to rescue singleton reads. Since expression level quantification relies on the number of mapped reads in a specific region, the use of singletons would affect the counting process and potentially change the overall accuracy. Obviously, while read pairs could be mapped both in SE and PE mode, singleton could be mapped only in SE mode. The final set of designed alignment strategies on both raw and preprocessed reads is summarized in the following list:

- **M_SE:** raw read pairs (**M**) were mapped in single-end mode (**SE**)
- **M_PE:** raw read pairs (**M**) were mapped in paired-end mode (**PE**)
- **P_SE:** preprocessed read pairs (**P**) were mapped in single-end mode (**SE**)
- **P_PE:** preprocessed read pairs (**P**) were mapped in paired-end mode (**PE**)
- **PS_SE:** preprocessed read pairs (**P**) and singletons (**S**) were mapped in single-end mode (**SE**)
- **PS_PE:** preprocessed read pairs (**P**) were mapped in paired-end mode (**PE**) while singletons (**S**) were mapped in single-end mode

The mapping statistics achieved using the previous mapping modes are shown in Table 7.3 and Table 7.4. BWAMEM and Bowtie2 show the highest percentage of mapped reads, both on real and simulated dataset. On the other hand, Bowtie achieves the lowest percentage of aligned reads on real dataset, while on simulated data Bowtie and BWA show the lower number of mapped reads. Except for Bowtie, all the tools in PE mode achieve a percentage of mapped reads greater than or equal to the corresponding SE mode, even if the differences are negligible. Similarly, raw and preprocessed dataset show comparable percentages of mapped reads. Curiously, BWA in M_SE mode maps an unusual low number of reads, while the same dataset mapped in PE mode achieves the typical percentage of aligned reads.

Table 7.3 Mapping statistics on the simulated datasets. The statistics labeled as (total) show the percentage of mapped reads calculated with respect to the total number of raw reads.

Mode	Aligner	% mapped read		% mapped read (total)	
		avg	SD	avg	SD
M_SE	BWA	85.601%	0.044%	85.601%	0.044%
M_SE	BWAMEM	99.670%	0.005%	99.670%	0.005%
M_SE	Bowtie	98.734%	0.031%	98.734%	0.031%
M_SE	Bowtie2	98.820%	0.021%	98.820%	0.021%
M_PE	BWA	97.129%	0.036%	97.129%	0.036%
M_PE	BWAMEM	99.844%	0.003%	99.844%	0.003%
M_PE	Bowtie	97.487%	0.048%	97.487%	0.048%
M_PE	Bowtie2	98.821%	0.021%	98.821%	0.021%
P_SE	BWA	98.692%	0.029%	75.625%	0.082%
P_SE	BWAMEM	99.997%	0.000%	76.625%	0.073%
P_SE	Bowtie	99.572%	0.028%	76.299%	0.081%
P_SE	Bowtie2	99.855%	0.021%	76.516%	0.078%
P_PE	BWA	99.892%	0.016%	76.545%	0.077%
P_PE	BWAMEM	100.000%	0.000%	76.627%	0.073%
P_PE	Bowtie	99.152%	0.042%	75.978%	0.084%
P_PE	Bowtie2	99.855%	0.021%	76.516%	0.078%
PS_SE	BWA	98.692%	0.028%	86.392%	0.054%
PS_SE	BWAMEM	99.997%	0.000%	87.535%	0.041%
PS_SE	Bowtie	99.573%	0.028%	87.163%	0.053%
PS_SE	Bowtie2	99.855%	0.021%	87.410%	0.048%
PS_PE	BWA	99.743%	0.016%	87.312%	0.048%
PS_PE	BWAMEM	99.999%	0.000%	87.536%	0.041%
PS_PE	Bowtie	99.205%	0.040%	86.841%	0.058%
PS_PE	Bowtie2	99.855%	0.021%	87.410%	0.048%

Table 7.4 Mapping statistics on the real datasets. The statistics labeled as (total) show the percentage of mapped reads calculated with respect to the total number of raw reads.

Mode	Aligner	% mapped read		% mapped read (total)	
		avg	SD	avg	SD
M_SE	BWA	98.218%	0.282%	98.218%	0.282%
M_SE	BWAMEM	99.881%	0.024%	99.881%	0.024%
M_SE	Bowtie	95.823%	0.583%	95.823%	0.583%
M_SE	Bowtie2	99.708%	0.058%	99.708%	0.058%
M_PE	BWA	98.744%	0.300%	98.744%	0.300%
M_PE	BWAMEM	99.894%	0.026%	99.894%	0.026%
M_PE	Bowtie	92.318%	0.919%	92.318%	0.919%
M_PE	Bowtie2	99.708%	0.058%	99.708%	0.058%
P_SE	BWA	98.225%	0.282%	97.998%	0.299%
P_SE	BWAMEM	99.881%	0.024%	99.651%	0.033%
P_SE	Bowtie	95.841%	0.583%	95.620%	0.599%
P_SE	Bowtie2	99.708%	0.058%	99.478%	0.065%
P_PE	BWA	98.745%	0.300%	98.518%	0.316%
P_PE	BWAMEM	99.894%	0.026%	99.664%	0.034%
P_PE	Bowtie	92.352%	0.915%	92.139%	0.930%
P_PE	Bowtie2	99.708%	0.058%	99.478%	0.065%
PS_SE	BWA	98.225%	0.282%	98.107%	0.292%
PS_SE	BWAMEM	99.881%	0.024%	99.762%	0.032%
PS_SE	Bowtie	95.840%	0.583%	95.726%	0.592%
PS_SE	Bowtie2	99.708%	0.058%	99.589%	0.067%
PS_PE	BWA	98.745%	0.300%	98.627%	0.310%
PS_PE	BWAMEM	99.894%	0.026%	99.775%	0.033%
PS_PE	Bowtie	92.355%	0.915%	92.245%	0.927%
PS_PE	Bowtie2	99.708%	0.058%	99.589%	0.067%

7.5 Quantification and normalization

Quantification was performed at the exon level, counting the number of reads overlapping the annotated regions, i.e. exons on simulated dataset and spike-ins on real dataset. The raw number of reads was computed using BEDTools. More sophisticated methods such as Cufflinks, RSEM and eXpress were not employed since the specific scenario (e.g. no multireads, no multiple isoforms, etc.) would not benefit from the advanced features implemented by these methods. In order to mitigate the length bias, the raw read counts were divided by the feature length. No between-library normalization were implemented since the assessment did not require any inter-library count computation.

7.6 Results of quantification assessment

The Person coefficient of correlation was computed between the normalized count and the known expression level for each library, both on real and simulated data. Table 7.5 and Table 7.6 show for each combination of aligner and mapping mode the average correlation among the 6 libraries and the corresponding standard deviation.

The results highlight a high consistency between tool performances. Neither the choice of the aligner nor the adopted mapping mode show any significant difference in terms of correlation (Wilcoxon-test p-value > 0.05). Similarly, the relative error analyses and the visual inspection of the inferred expression level plots do not show any significant difference between the tested options. An example for a single real dataset is shown in Figure 7.5, where the results achieved by two different modes such as M_SE and PS_PE are compared. The figure highlights how using two very different workflows (i.e. M_SE uses raw data and a single-end mapping mode, whereas PS_PE employs preprocessed data and a pair-end mapping mode) does not result in any considerable difference in terms of correlation and overall agreement between spike-ins true concentrations and computed counts.

The comparisons performed in section 6.2 show a ~3% difference at the read level recall between the worst (Bowtie) and the best (BWAMEM) performer. This performance difference seems to not affect the expression level accuracy, suggesting that such small variations could be negligible for a quantification study involving simple organisms.

In addition, this analysis shows that the adoption of a more complex (and more expensive) paired end sequencing protocol does not bring any improvement in a quantification study where splicing in not a primary issue. Similarly, read preprocessing does not seem to significantly improve the overall quantification accuracy.

Table 7.5 Quantification accuracy (Pearson correlation coefficient) on the simulated datasets

Mode	Aligner	Correlation	
		avg	SD
M_SE	BWA	0.923	0.031
M_SE	BWAMEM	0.926	0.029
M_SE	Bowtie	0.924	0.031
M_SE	Bowtie2	0.924	0.030
M_PE	BWA	0.926	0.029
M_PE	BWAMEM	0.926	0.029
M_PE	Bowtie	0.922	0.031
M_PE	Bowtie2	0.925	0.030
P_SE	BWA	0.925	0.030
P_SE	BWAMEM	0.925	0.029
P_SE	Bowtie	0.925	0.030
P_SE	Bowtie2	0.925	0.029
P_PE	BWA	0.926	0.029
P_PE	BWAMEM	0.926	0.029
P_PE	Bowtie	0.924	0.030
P_PE	Bowtie2	0.926	0.030
PS_SE	BWA	0.924	0.030
PS_SE	BWAMEM	0.925	0.030
PS_SE	Bowtie	0.924	0.030
PS_SE	Bowtie2	0.925	0.030
PS_PE	BWA	0.925	0.030
PS_PE	BWAMEM	0.926	0.030
PS_PE	Bowtie	0.923	0.030
PS_PE	Bowtie2	0.925	0.030

Table 7.6 Quantification accuracy (Pearson correlation coefficient) on the real datasets

Mode	Aligner	Correlation	
		avg	SD
M_SE	BWA	0.969	0.013
M_SE	BWAMEM	0.970	0.012
M_SE	Bowtie	0.966	0.015
M_SE	Bowtie2	0.970	0.012
M_PE	BWA	0.970	0.012
M_PE	BWAMEM	0.970	0.012
M_PE	Bowtie	0.964	0.016
M_PE	Bowtie2	0.970	0.012
P_SE	BWA	0.969	0.013
P_SE	BWAMEM	0.970	0.012
P_SE	Bowtie	0.966	0.015
P_SE	Bowtie2	0.970	0.012
P_PE	BWA	0.970	0.012
P_PE	BWAMEM	0.970	0.012
P_PE	Bowtie	0.964	0.016
P_PE	Bowtie2	0.970	0.012
PS_SE	BWA	0.969	0.013
PS_SE	BWAMEM	0.970	0.012
PS_SE	Bowtie	0.966	0.015
PS_SE	Bowtie2	0.970	0.012
PS_PE	BWA	0.970	0.012
PS_PE	BWAMEM	0.970	0.012
PS_PE	Bowtie	0.964	0.016
PS_PE	Bowtie2	0.970	0.012

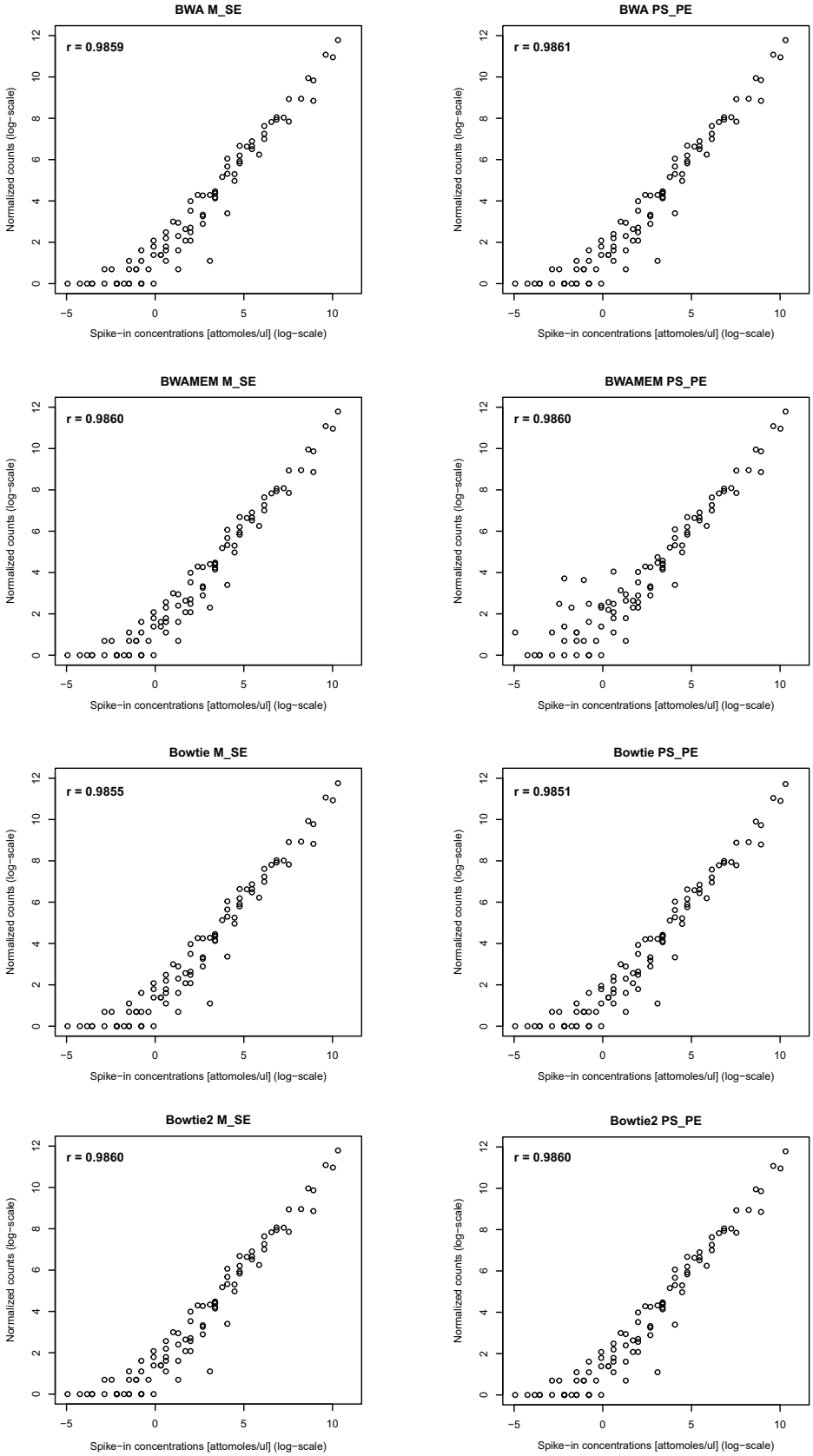

Fig. 7.5 *Comparison of spike-in true concentrations and RNA-Seq normalized counts in log-log scale.* The figure shows the result on a real dataset using the mode M_SE (left) and PS_PE (right). Pearson's correlation *r* between spike-in true concentrations and normalized counts is also reported.

Chapter 8

Conclusions

RNA-Seq has become one of the most adopted methodologies in transcriptomic studies, providing nearly unlimited possibilities in modern transcriptome analyses. Exploiting the incredible progress achieved by NGS technologies, RNA-Seq allows sequencing at single base resolution all the RNA species present in a sample, characterizing their sequences and quantifying their abundances at the same time. NGS platforms produce millions of short transcript sub-sequences, called *reads*, sequenced from random positions of the input RNAs. Unfortunately, the sequencing process does not provide any information about which transcripts have generated the reads or from which part of the transcripts they come from. In order to identify the relation between the sequencing output (i.e. reads) and the sequenced transcripts, the common approach consists in aligning the reads against a reference genome and then inferring which transcripts have generated them by analyzing the read locations. At first glance, the read alignment tasks may seem very simple, but its implementation is in fact complex and still not well defined. In recent years there was a considerable effort of the research community in the development of RNA-Seq read alignment methods, resulting in more than 20 published tools since 2009. However, the large number of available methods has made the definition of a robust and unified computational pipeline difficult, mainly due to the lack of comprehensive assessment studies.

The main result of this thesis is the systematic assessment of the read alignment step with the goal of improving recommendations for RNA-Seq pipelines. The relevance of this topic is testified by a recent publication in *Nature Methods* (*Baruzzo et. al* [128]). Three main objectives were designed and achieved in order to accomplish the final goal: *i)* a thorough definition of the read alignment problem, *ii)* the identification of the state of the art RNA-Seq alignment methods and the assessment of their performance and *iii)* the assessment of the role of RNA-Seq read alignment in the accuracy of expression level quantification analyses.

Definition of the read alignment problem

The thorough definition of the read alignment problem was achieved by identifying the most important characteristics and challenges. The analysis revealed that intron size gap and indel are the most challenging situations in the alignment process and that the chosen data structures and algorithms have a major role in the final accuracy. With regards to the available tools, both hash based and BWT based approaches have strengths and weaknesses, so it is difficult to identify a priori the best solution. Hash based methods allow handling of errors during the alignment process, at least when many seeds are employed for each read. However, the alignment accuracy and efficiency are strongly affected by the non-trivial choice of seed sizes and positions. On the other hand, BWT methods are very fast and memory efficient, but they have some limitations in handling inexact matches. Mismatches and gaps in the reads are managed through seed&extend strategies or heuristics, whose accuracies strongly depend on the number of allowed errors. Many of the most recent NGS read mappers are based on BTW methods rather than hash based methods. The first generation of short DNA read aligners, tools like BLAST, BLAT, MAQ, Eland and SOAP, were based on hashing methods while after the publication of tools like BWA and Bowtie, BWT methods have become more popular. Even in the context of RNA-Seq aligners, the current trend shows more BWT based methods (e.g. ContextMap2, HISAT, HISAT2, MapSplice2, TopHat2, etc.) than Hash based methods (Novoalign and Subjunc). Finally, few methods employ uncompressed suffix arrays instead of Burrows–Wheeler transform (CLC Genomic Workbench and STAR) to achieve higher alignment speed at the cost of larger memory requirements.

Identification and assessment of state of the art RNA-Seq alignment methods

The second objective involved the identification of the state of the art RNA-Seq alignment methods and the assessment of their performance. State of the art methods for RNA-Seq read alignment were identified by a thorough literature search that involved more than 2000 peer reviewed publications in the context of RNA-Seq. The literature search highlighted that the majority of RNA-Seq studies employ TopHat/TopHat2, followed by Bowtie/Bowtie2 and BWA/BWAMEM. While TopHat2 is a splice aware alignment method, both Bowtie/Bowtie2 and BWA/BWAMEM are splice unaware methods so their use should be limited to simple organisms or transcriptome alignment. However, the literature search revealed that the use of these tools is very often not limited to these scenarios, resulting in many unsuitable applications of these methods. Finally, the literature search highlighted that RNA-Seq read alignment remains in a state of confusion mainly due to a lack of accurate and systematic benchmarking studies. With the purpose of addressing this issue, a thorough benchmark

analysis of splice aware and splice unaware read alignment methods was designed and performed.

The analysis of state of the art splice aware algorithms highlighted some interesting results. First, there is a considerable difference between tool's accuracy, making the choice of the right aligner a pivotal step in many RNA-Seq studies. Section 5.2 shows that the differences between aligner performances are detectable even in the less complex datasets and a great variability of methods performance is observed in the case of complex datasets. Generally, the tools show a high precision in the different metrics, meaning that the reads that they are able to align are usually in the right position. On the other hand, it is the recall that better discriminates the tools performance, with many tools that are able to align only few reads when the dataset are complex. In many applications where the number of aligned reads is important (e.g. quantification and DE studies), this would have a major role. The results are consistent between the tested organisms, human and P. falciparum, suggesting that the tool's performance do not depend on the particular organism and that the alignment strategies are general enough to handle very different genomes. At the base, read and junction level, it is possible to identify in CLC, Novoalign, STAR and GSNAP the best performing tools in terms of accuracy. These same tools show effective default settings compared with other tools, which instead require a most complex parameters tweaking in order to achieve acceptable results in the most complex datasets (Figure 5.9). A tool with a smart and effective default is preferable due to the impossibility of performing an accurate tuning in the real scenario, so these tools guarantee reliable results on a larger variety of input datasets. Interestingly, the most widely used RNA-Seq aligner (i.e. TopHat2) shows poor performance with the default parameters as well as in many other tested scenarios.

Regarding the employment of a reliable annotation as input to the alignment process, it seems to not considerably affect the alignment accuracy, even if it helps in some particular scenario. Specifically, the use of annotation does not result in any considerable improvement at the read and base level, whereas a few tools show minor improvements at the junction level (Figure 5.6). Generally, the 2-pass aligners (e.g. STAR) achieve comparable results with and without annotations, while some 1-pass aligners (e.g GSNAP and STAR-1-pass) show some benefits from the presence of an input annotation. Since a reliable annotation is not always available, the small performance differences between providing and omitting an input annotation is a remarkable result. In this case, it is important to underline that CLC and Novoalign have the limitation to rely on an input annotation to detect splice junctions, even if these tools achieve excellent results when it is provided as input. On the other hand, the results shown in section 5.5 underscore that a major role in the alignment accuracy is played by the read preprocessing. The results underscore how few tools are not able to handle

unpreprocessed reads, resulting in poor results when a preprocessing step is not performed. This limitation is mainly due to the lack of soft-clipping and local alignment features, that would be even more important with future longer reads. As for multimappers, the tests show that many tools are able to identify the right location among the reported ones. This result suggests that the use of post-alignment tools able to properly exploit the multimapper information, instead of discarding them, could result in more accurate downstream analyses.

Another resulting trend is the one that relates the tools accuracy to the algorithm and data structure employed. Among the four best performing tools, CLC and STAR use an uncompressed suffix array, Novoalign exploits a hash table while GSNAP use both a suffix array and a hash table. None of these tools relies entirely on the Burrows Wheeler transform, which is instead exploited by almost all the remaining tools. BWT based methods have the main advantage of requiring a low amount of memory and allowing fast execution time. On the other hand, the use of suffix array (especially with no compression) requires a considerable amount of memory. The use of a BWT with a FM-Index was probably a good choice for the first generation of RNA-Seq aligners, where the main constraint was the low amount of available RAM. However, since RAM price decreased in recent years and many machine today mount at least 32GB of memory, the use of uncompressed data structure has become more popular. In 2013, STAR was one of the first tools to exploit the higher amount of memory resulting in lower execution time. Similarly, GSNAP updated its core alignment engine to exploit a suffix array. The last generation of BWT based algorithms (e.g. HISAT and HISAT2) still rely on memory efficient data structure and are able to achieve faster alignment than STAR and GSNAP. However, these tools seem to still not guarantee a comparable level of accuracy. Even though the progress in terms of efficiency and accuracy allows the identification of a set of reliable aligners, our results highlight that there are still two main challenges that need to be addressed: junction identification and indel detection. Junction level metrics show the worst results compared to base and read level metrics, highlighting that junction detection is still one of most complex task in the alignment process. Short anchored reads and non-canonical junctions are two of the most complex situations in which current state of the art aligners need to be improved (Figures 5.7 and 5.8). The second challenge involves indel detection and it is probably the one which needs more attention. Indel detection is intrinsically a complex task and the results in section 5.7 highlight that just few aligners reach an acceptable accuracy in this respect. Indel discovery has a major role in the study of many diseases, including cancer, and a higher accuracy would be necessary to fully explore these phenomena.

While the splice aware tools analysis highlights several new outcomes, the performed comparison of splice unaware methods largely confirms the literature results. BWAMEM

and Bowtie2 outperform their previous versions in terms of accuracy, showing comparable precision but achieving higher recall both at the base and read level. Unlike previous studies, the distinct analysis at the read and base level allows highlighting some interesting results. Bowtie and Bowtie2 achieve better results at the base level (Figure 6.1), while at the read level (Figure 6.3) BWAMEM and BWA outperform Bowtie2 and Bowtie, respectively. This is an unusual trend, especially compared to splice aware tools where the best performers at the base level are usually the best performers even at the read level. The different behaviors between base and read levels are probably due to the different alignment policies adopted by the tools. Bowtie and Bowtie2 perform an end-to-end alignment by default, while BWA and BWAMEM adopt a local alignment policy. An end-to-end strategy requires finding a full alignment for each read, discarding reads that are not completely aligned. This approach would result in both a lower number of mapped reads (due to the constraint on the full alignment) and a higher execution time (due to the extra amount of effort to completely align complex reads). As a consequence, this approach obtains a higher number of correctly aligned bases. On the contrary, local alignment strategies allow both mapping a higher number of reads (due to the soft-clipping of complex read portion) and a lower amount of computation (due to the lower number of bases to align). In this way, the latter approach has a higher number of correctly mapped reads. By analyzing Bowtie2 and BWAMEM mapping strategies and the related results, it is easy to identify in the above alignment policies (i.e. end-to-end vs. local) the explanation for the different speeds and alignment accuracies at the read and base level. The results in terms of accuracy are largely consistent with the literature on alignment benchmarking. Moreover, this thesis results highlight that the gap between splice aware and splice unaware methods in terms of resource usage is continuously diminishing. In terms of computational time, many splice aware methods achieve the same speed of the tested unspliced aligners, whereas splice aware tools are still more expensive in terms of memory usage. Since the main application of splice unaware tools on simple organism is due to the lower computational requirements compared with splice aware methods, these results suggest that soon the benefits in terms of speed and memory will be negligible.

A questionable limitation of the alignment methods assessment presented in this thesis is that the conclusions are based only on simulated data. Obviously, the conclusions based on data produced from a RNA-Seq read simulation are as accurate as the simulated data resemble real data. However, there is no ground truth with real data, resulting in the impossibility of performing any accuracy analysis. Metrics such as the percentage of mapped/unmapped reads are not reliable indices of the accuracy of the alignment process. For example, increasing the number of allowed mismatches would probably result in a higher number of aligned reads but at the same time it could increase the probability of mapping the reads to the wrong

positions. An alternative approach could be to compare the alignment outputs of different methods and to infer the correct read position based on the agreement between the several tools [155]. However, this approach provides only a heuristics way to assess the alignment accuracy and it is limited by the accuracy of the analyzed methods itself. In this scenario, the use of simulated data is the most reasonable option, even if the choice of the right RNA-Seq data simulator has a pivotal role in the reliability of the final results. While many simulators are available in the context of NGS data [156], just few simulators are specifically developed for RNA-Seq application. In the context of RNA-Seq alignment benchmarking, BEERS and Flux are the most reliable simulators and indeed they are the only two tools that were specifically designed to generate realistic RNA-Seq reads.

Assessment of RNA-Seq read alignment in the accuracy of quantification

The last objective was related to the assessment of the role of RNA-Seq read alignment in the accuracy of a very common downstream analysis such as expression level quantification. The performance in terms of quantification accuracy of BWA, BWAMEM, Bowtie and Bowtie2 were studied on both real and simulated data. Different from splice aware methods, these tools showed small performance differences in the alignment accuracy, allowing to identify if expression level quantification accuracy is sensitive to minor changes in the alignment output. The results in section 7.6 reveal that the different alignment methods do not affect the quantification accuracy, suggesting that such small alignment accuracy variations are negligible in the context of expression level analyses. Even employing different read preprocessing policies and several alignment strategies seems to not affect the final quantification result. Therefore, in the context of expression level quantification on simple organisms the adoption of different analysis workflows seems to be irrelevant.

Arguably, the considerable differences between tools performance observed in the context of splice aware alignment methods would instead result in significant changes in quantification accuracy. However, the assessment on expression level accuracy in the context of complex organisms is difficult due to the presence of alternative splicing and the need of a reliable gene model. In addition, the assessment would require the adoption of complex quantification methods in order to achieve a gene level or transcript level quantification. As results, different quantification methods should be tested introducing another level of variability. Although such assessment represents a complex challenge, an effort in designing this analysis will be part of future works.

In the study of quantification accuracy, an interesting methodological question arises regarding the definition of accuracy metrics. Currently, the correlation between the inferred and the known expression levels is the most common way to assess quantification accuracy.

However, correlation is not able to fully describe quantification accuracy and for this reason other auxiliary metrics are usually employed. Even combining different metrics, the definition of a robust index for the assessment of quantification accuracy is challenging. To further investigate this open problem, the study of the available accuracy indices and the design of novel metrics will be part of future works.

Final remarks

The current trend of increasing read lengths and the rising of new sequencing technologies make it difficult to foresee what will be the leading alignment approach in the future. Even keeping unvaried the current error rates, longer reads would result in a larger number of errors per read and consequently new challenges to address. In the near future, the advent of longer reads would probably require to redefine some of the mapping strategies and alignment policies currently employed.

Based on the idea of redefining the mapping process, concepts of "*lightweight alignment*", "*alignment free*" and "*pseudo alignment*" have been introduced in the last two years. The intuition behind these terms is that in some context the mapping process could be redefined so that the base-by-base matching between the read sequence and the reference sequence is no longer required. Instead, information theory techniques, k-mer frequencies and similarity measures are used to define mapping criteria between the read and the reference sequence. These concepts were first implemented by tools such Kallisto [157], Salmon [158] and Sailfish [159] in the field of transcript quantification abundance, where a perfect end-to-end match between the read and the reference is often not necessary. Indeed, quantification studies simply need to assign the reads to the correct genomic features, in order to count how many reads overlap that genomic region. For this reason, no information about the precise position of the read inside the feature is usually necessary since the knowledge of the relation between the read and the feature is enough for the objectives of this kind of studies. Compared to a typical expression level analysis (i.e. read alignment and quantification), pseudo alignment methods achieve at the same time a comparable level of accuracy and a speed-up factor between 20 and 100 in many scenarios [157, 159]. For applications where a base-by-base matching between the read and the reference sequence is not necessary, these methods have the potential to revolutionize the mapping and quantification process.

Appendix A

SAM file format

SAM (Sequence Alignment/Map) format is the standard file format for read alignment output. The information about the read mappings is presented in a TAB delimited text format consisting of a header section (optional) and an alignment section. This appendix contains only a brief description of the SAM file format in the context of RNA-Seq read alignment; the complete file format specification is presented in [63].

A.1 Header section

Header lines start with '@' and contain several auxiliary information about the alignment process such as the name and version of the employed aligner, the mapping tool command used, the reference sequences names and lengths, the sorting order of the alignments, etc. If present, the header section must precede the alignments section.

A.2 Alignment section

In the alignment section, the alignment of each read is reported in a single line. Each line contains 11 mandatory fields describing several alignment information (Table A.1).

Table A.1 Overview of SAM file mandatory fields. Table adapted from [63]

Field#	Field name	Type	Regexp/Range	Brief description
1	QNAME	String	[!-?A-~]{1,254}	Read NAME
2	FLAG	Int	$[0,2^{16}-1]$	bitwise FLAG
3	RNAME	String	*|[!-()+-<>-~][!-~]*	Reference sequence NAME
4	POS	Int	$[0,2^{31}-1]$	1-based leftmost mapping POSition
5	MAPQ	Int	$[0,2^{8}-1]$	MAPping Quality
6	CIGAR	String	*|([0-9]+[MIDNSHPX=])+	CIGAR string
7	RNEXT	String	*|=|[!-()+-<>-~][!-~]*	Ref. name of the mate/next read
8	PNEXT	Int	$[0,2^{31}-1]$	Position of the mate/next read
9	TLEN	Int	$[-2^{31}+1,2^{31}-1]$	Observed Template LENgth
10	SEQ	String	*|[A-Za-z=.]+	Read SEQuence
11	QUAL	String	[!-~]+	ASCII of Phred-scaled base QUALity+33

1. **QNAME**: Read identifier; reads having the same QNAME are regarded to come from the same fragment (e.g. paired-end reads). If a read is a multimapper, then it may occupy multiple alignment lines.

2. **FLAG**: Combination of bitwise flags; each bit is associated to a specific alignment information (Table A.2).

Table A.2 Overview of bitwise FLAGs. Table adapted from [63]

Bit	Bit(hex)	Description
1	0x1	read paired
2	0x2	read paired in paired properly according to the aligner
4	0x4	read unmapped
8	0x8	mate unmapped
16	0x10	read reverse strand
32	0x20	mate reverse strand
64	0x40	the first read in the read pair
128	0x80	the second read in the read pair
256	0x100	secondary alignment
512	0x200	not passing filters, such as platform/vendor quality controls
1024	0x400	PCR or optical duplicate
2048	0x800	supplementary alignment

3. **RNAME**: Reference sequence name; usually it contains the chromosome name.

4. **POS**: Leftmost mapping POSition of the first matching base. Position coordinate is 1-based (i.e. the first base in the reference sequence has coordinate 1).

5. **MAPQ**: MAPping Quality; it equals $ceil(-10log_{10}P(\text{mapping position is wrong}))$

6. **CIGAR**: CIGAR string; it contains a sequence of CIGAR operations (Table A.3).

Table A.3 CIGAR operations. Table adapted from [63]

Operation	Description
M	alignment match (sequence match or mismatch)
I	insertion to the reference
D	deletion from the reference
N	skipped region from the reference (intron)
S	soft clipping (clipped sequences present in SEQ)
H	hard clipping (clipped sequences NOT present in SEQ)
P	padding (silent deletion from padded reference)
=	sequence match
X	sequence mismatch

7. **RNEXT**: Reference sequence name of the primary alignment of the NEXT read (read mate).

8. **PNEXT**: Position in the RNEXT sequence of the primary alignment of the NEXT read (read mate).

9. **TLEN**: Signed observed Template LENgth; for read pairs, if both the reads are mapped to the same reference, the TLEN equals the number of bases from the leftmost mapped base to the rightmost mapped base.

10. **SEQ**: Read SEQuence; the length of the sequence must equal the sum of lengths of M/I/S/=/X operations in CIGAR.

11. **QUAL**: ASCII encoding of base QUALity plus 33; it is the same as the quality string in the Sanger FASTQ format.

Appendix B

P. falciparum results figures

B.1 Results using the aligner tools default parameters

B.1.1 Base level

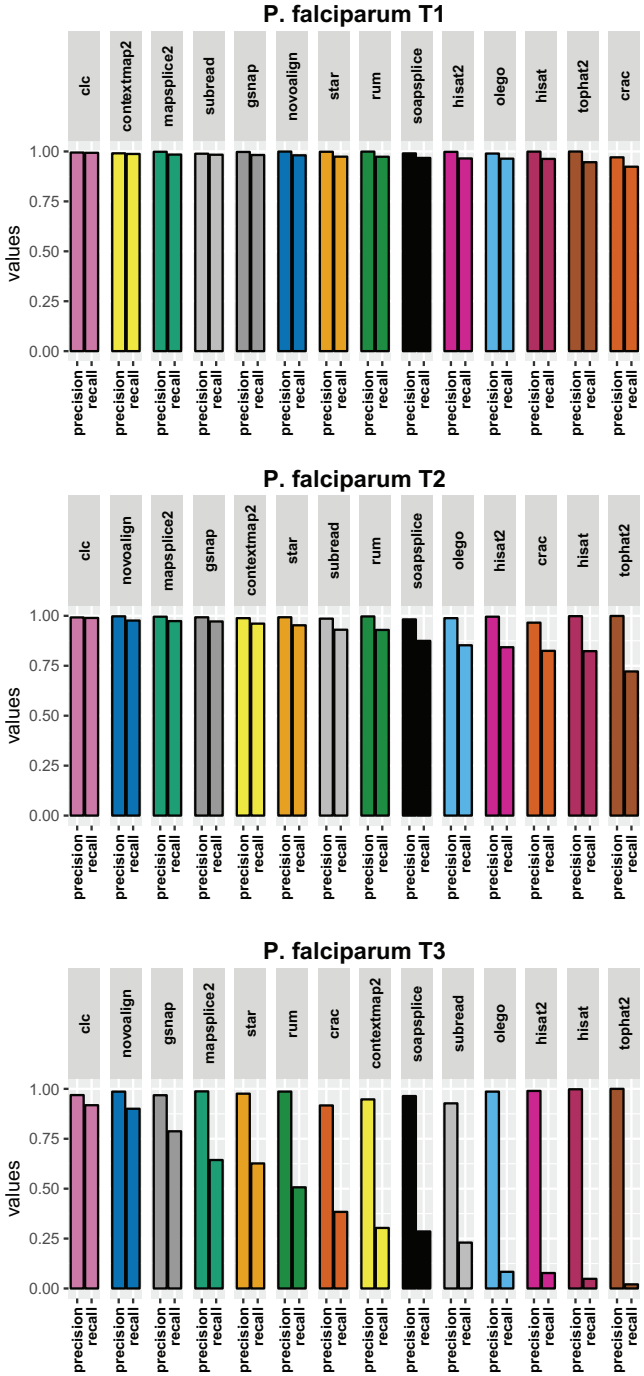

Fig. B.1 *Default parameters - Base level precision and recall for the P. falciparum datasets.* The tools are sorted by descending recall.

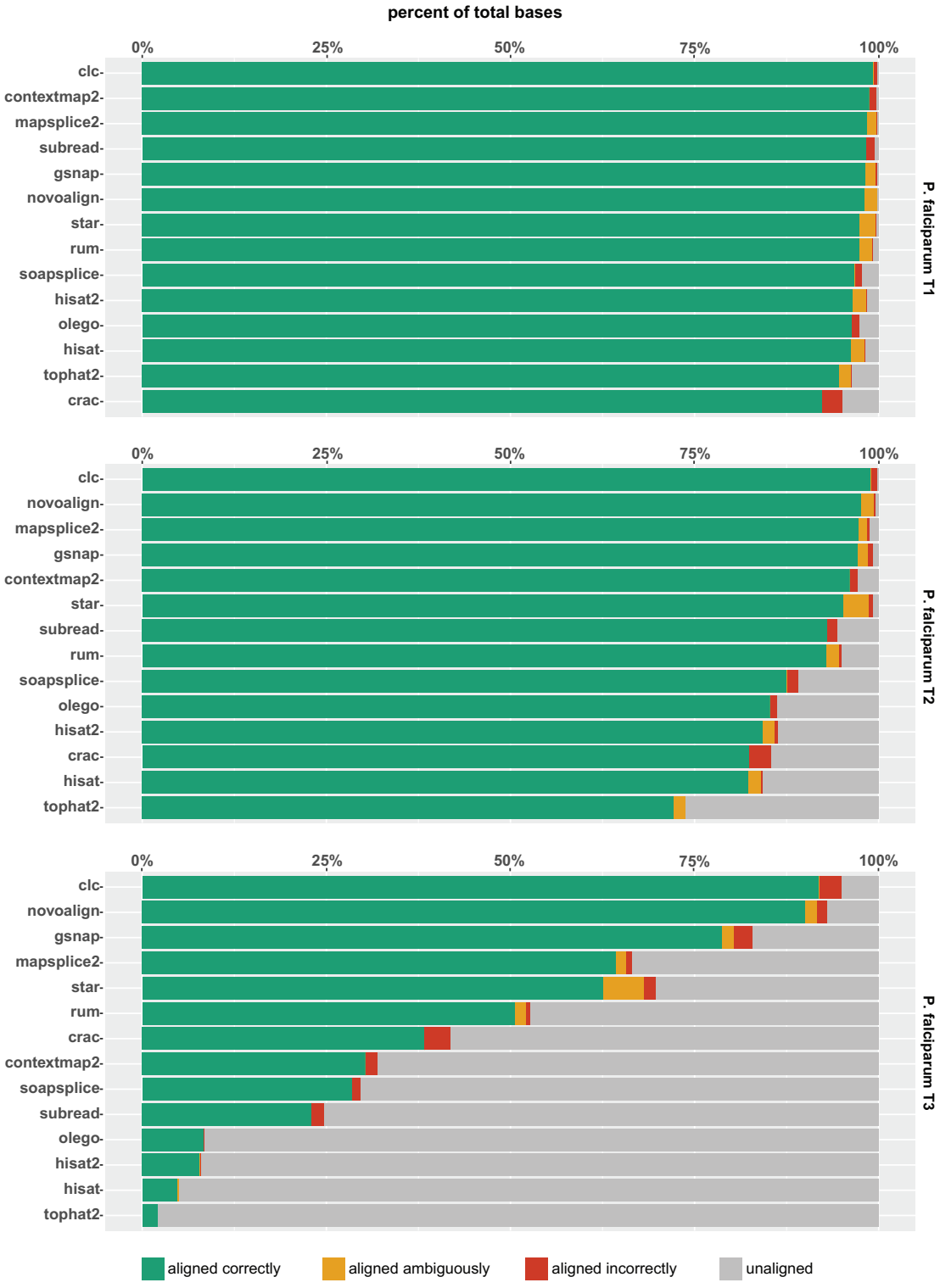

Fig. B.2 *Default parameters - Base level statistics for the P. falciparum datasets.* For each dataset, the bars show the percentage of bases aligned correctly, aligned ambiguously, aligned incorrectly and unaligned by each tool. The tools are sorted by descending percentage of bases aligned correctly.

B.1.2 Read level

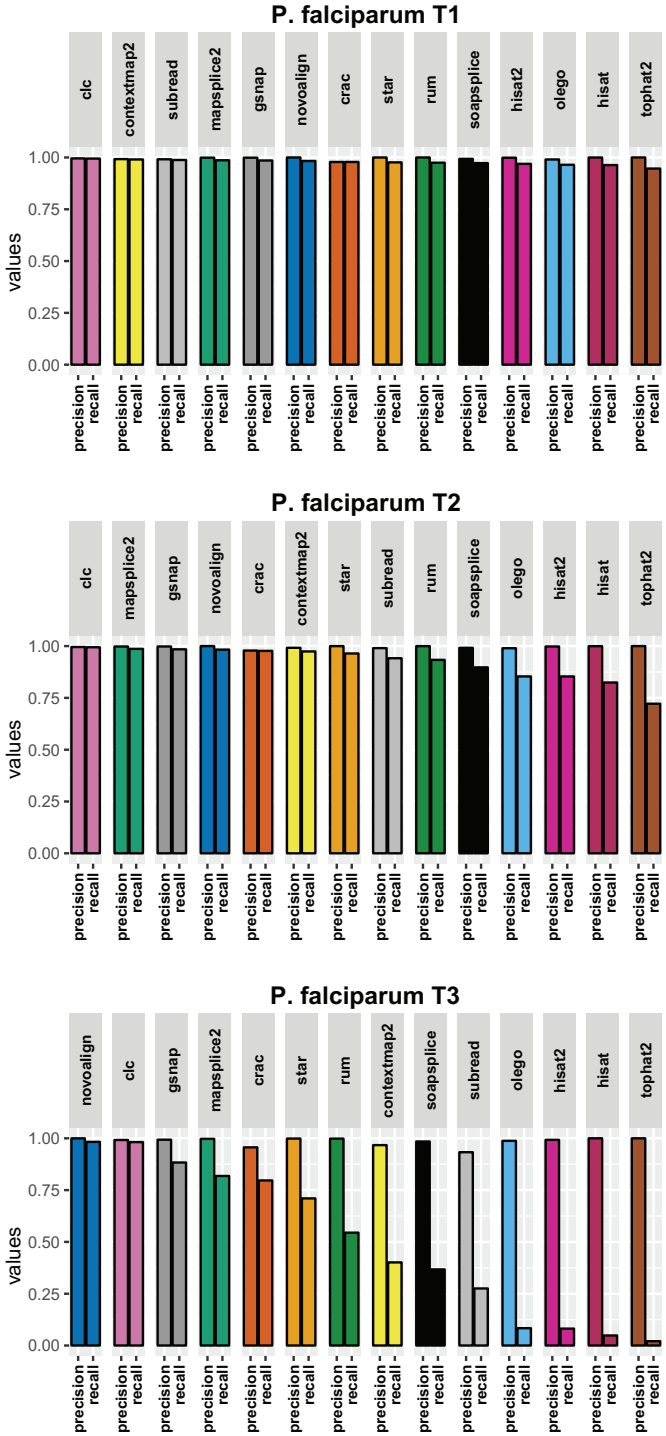

Fig. B.3 *Default parameters - Read level precision and recall for the P. falciparum datasets.* The tools are sorted by descending recall.

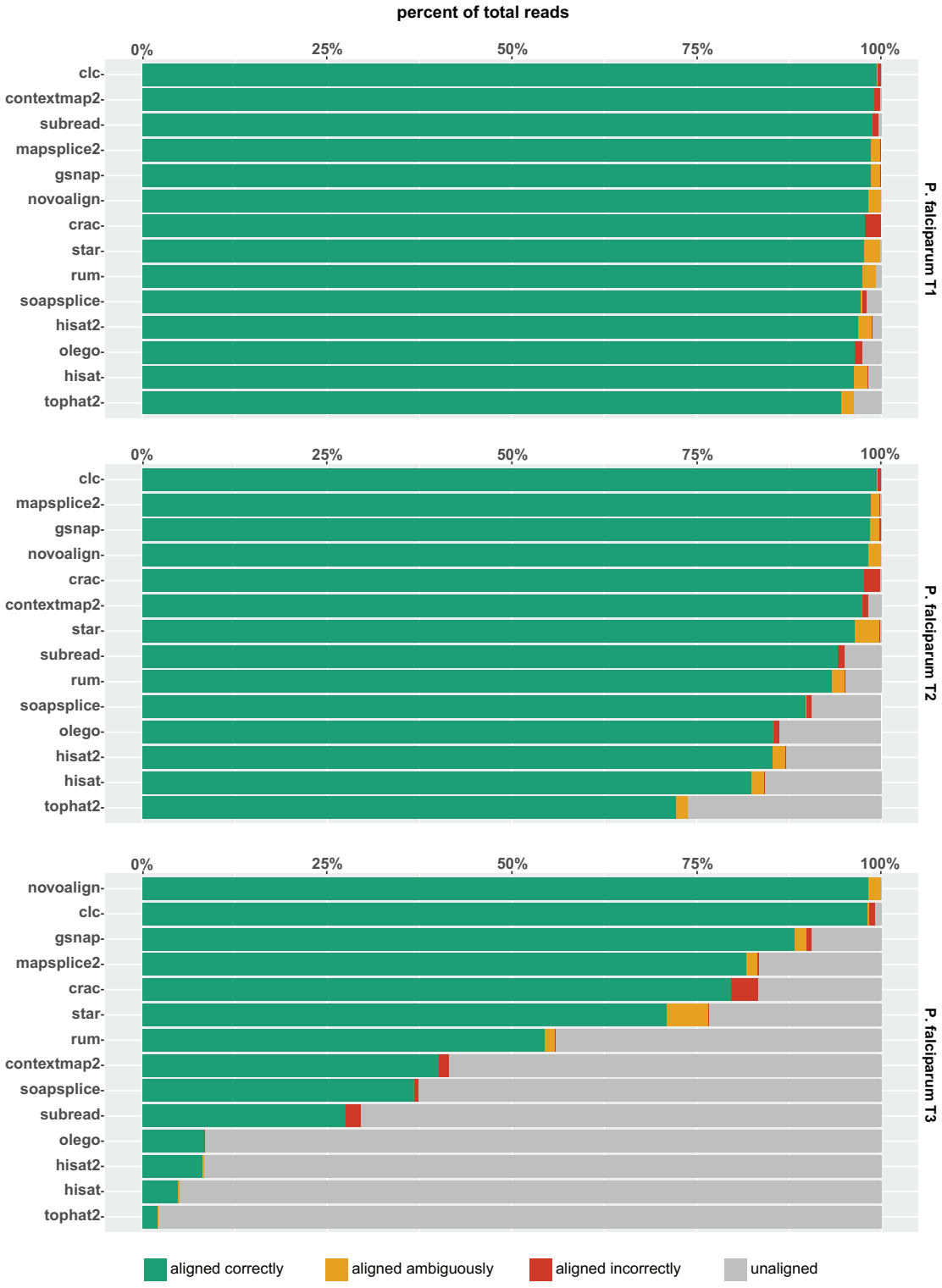

Fig. B.4 *Default parameters - Read level statistics for the P. falciparum datasets.* For each dataset, the bars show the percentage of reads aligned correctly, aligned ambiguously, aligned incorrectly and unaligned by each tool. The tools are sorted by descending percentage of reads aligned correctly.

B.1.3 Junction level

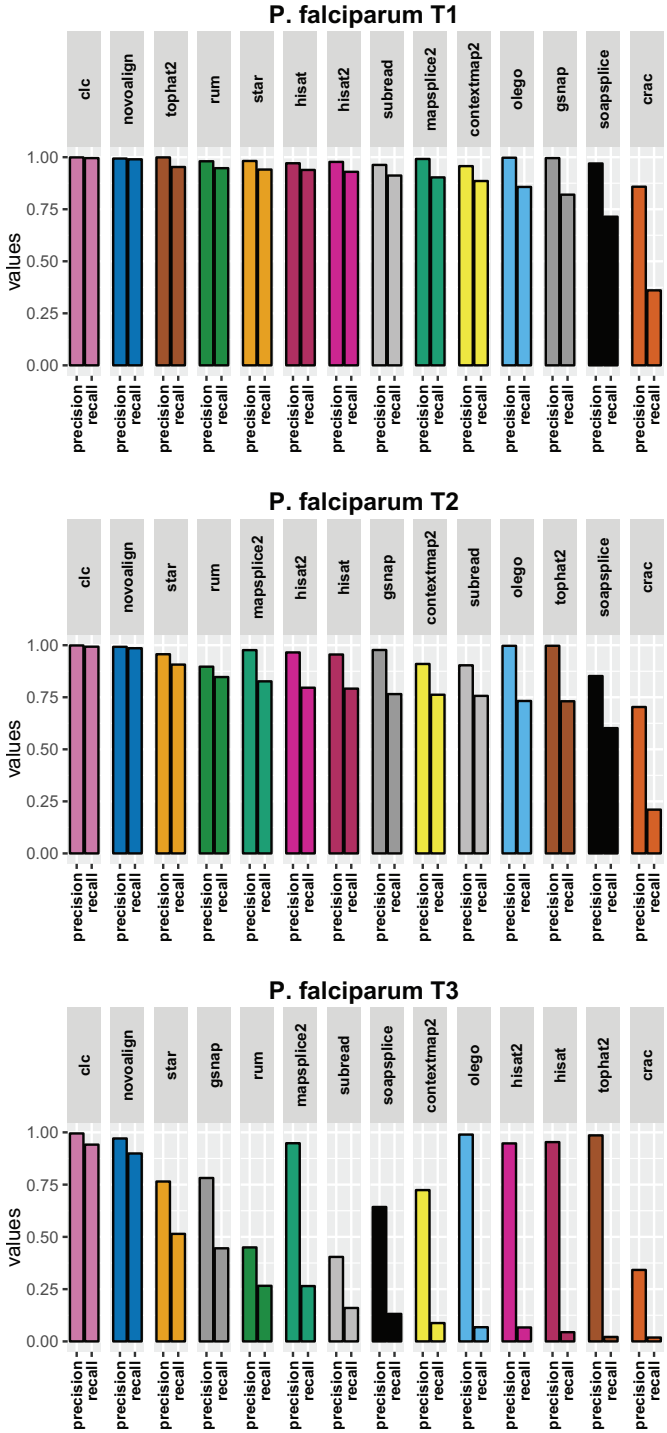

Fig. B.5 *Default parameters - Junction level precision and recall for the P. falciparum datasets.* The tools are sorted by descending recall.

B.2 Effect of input annotation

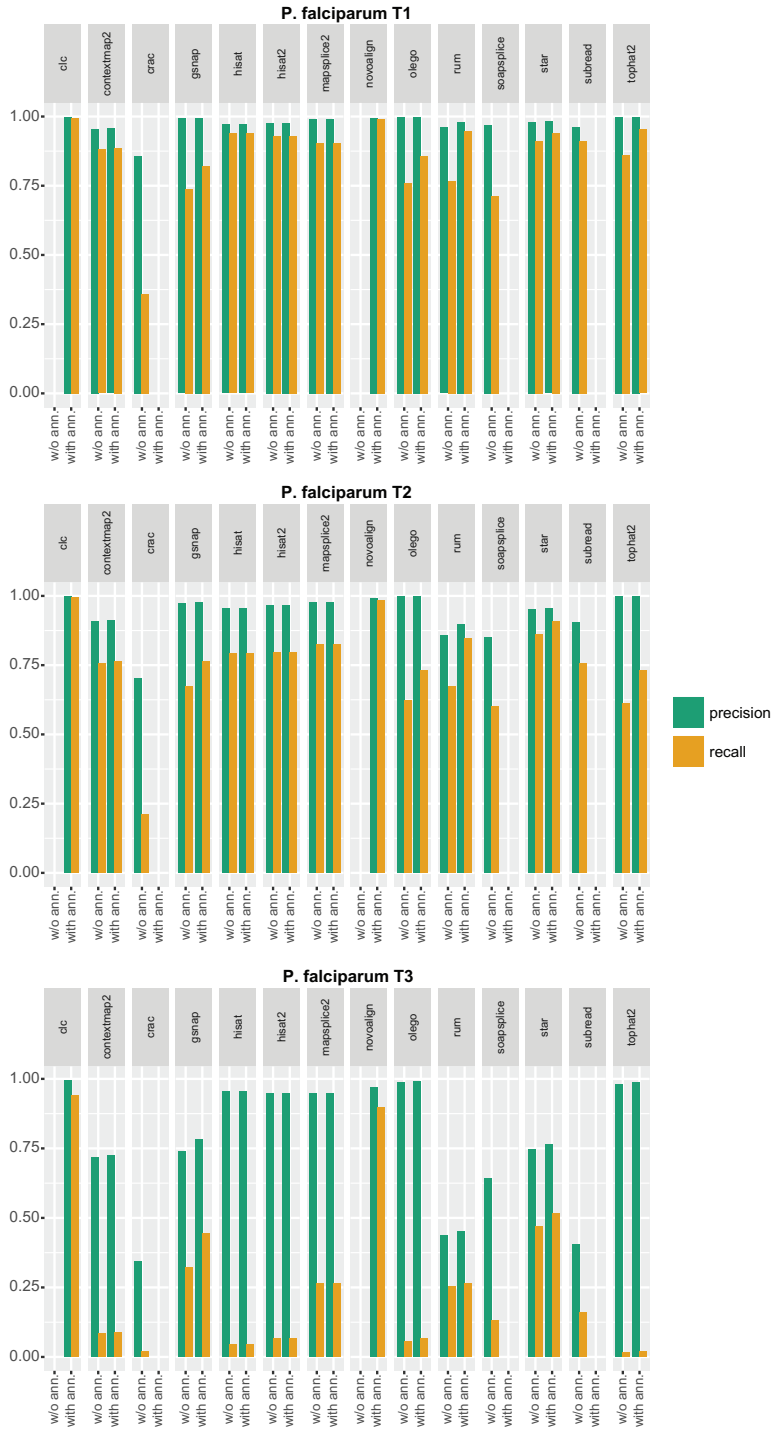

Fig. B.6 *Effect of annotation - Junction level precision and recall for the P. falciparum datasets.*

B.3 Effect of parameters tweaking

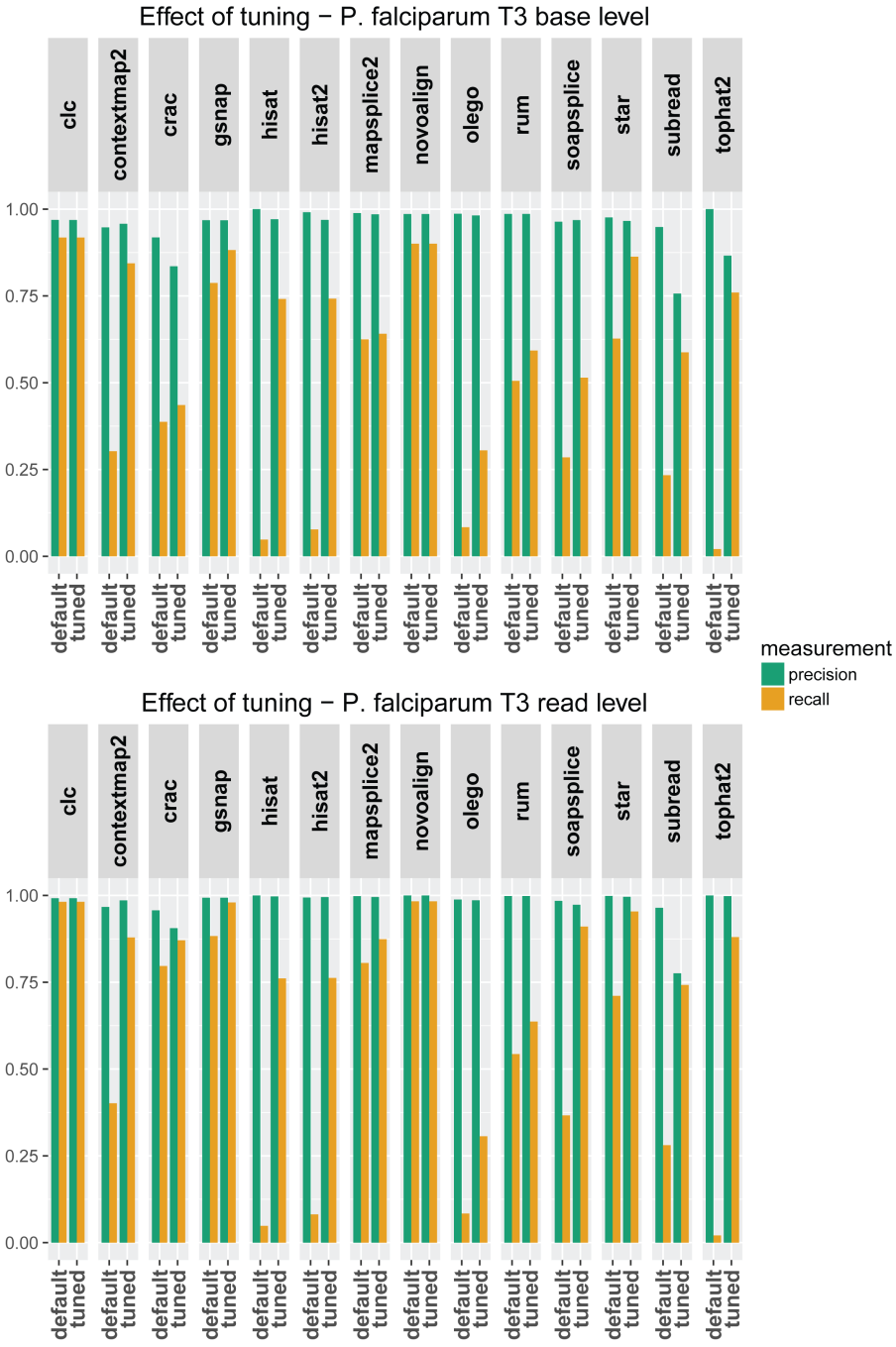

Fig. B.7 *Effect of parameter tweaking - Precision and recall at the base and read level for the P. falciparum T3 dataset.* For each tool, the figures show the precision and recall at the base level (top) and read level (bottom) for the "default" and the "tuned" alignments. The "tuned" alignment is the best configuration (in terms of base recall (top) and read recall (bottom)) achieved by the tweaking process.

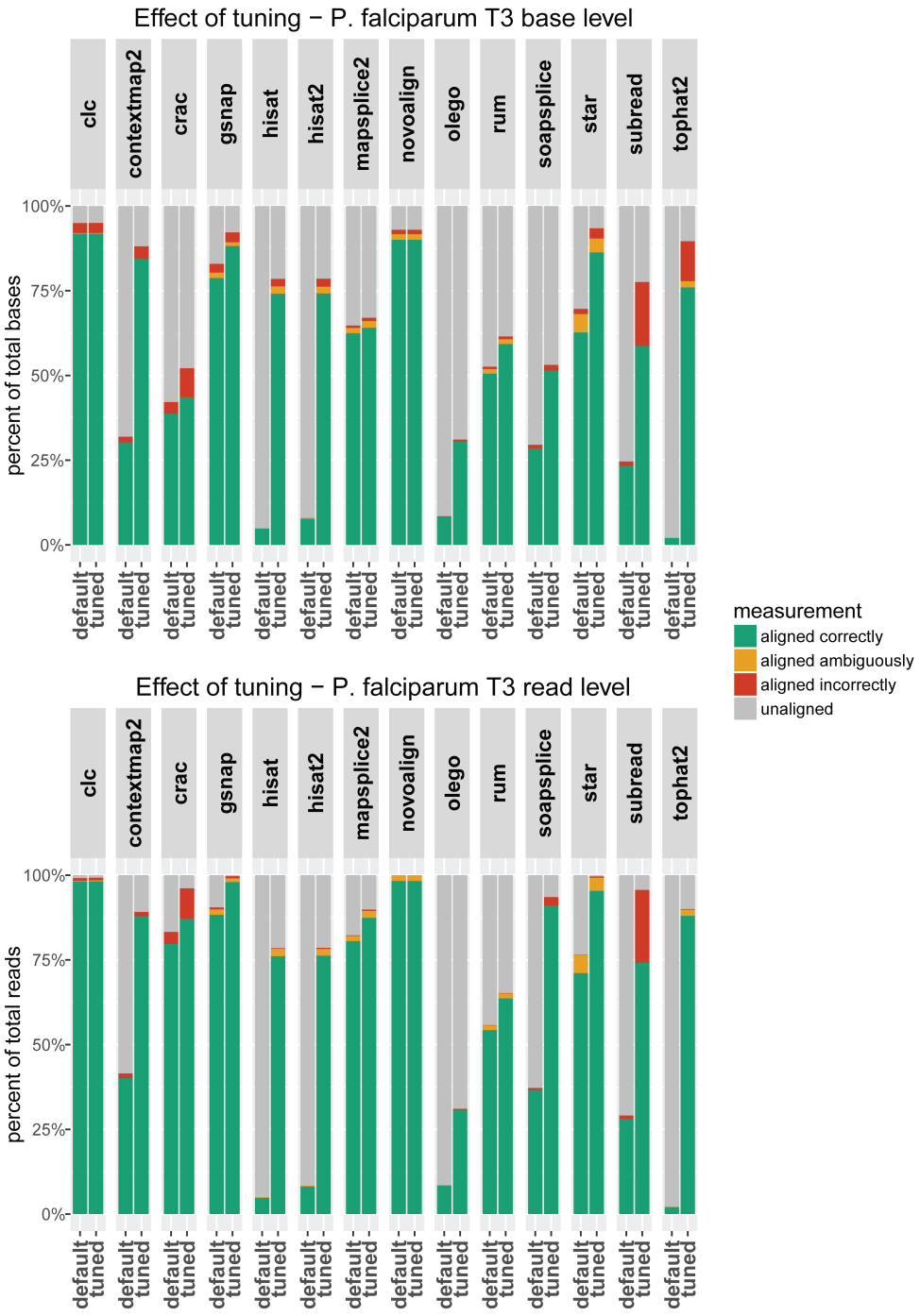

Fig. B.8 *Effect of parameter tweaking at the base and read level for the P. falciparum T3 dataset.* The bars show the percentage of bases/reads aligned correctly, aligned ambiguously, aligned incorrectly and unaligned by each tool. For each tool, the figures show the alignment statistics for the "default" and the "tuned" alignments. The "tuned" alignment is the best configuration (in terms of base recall (top) and read recall (bottom)) achieved by the tweaking process.

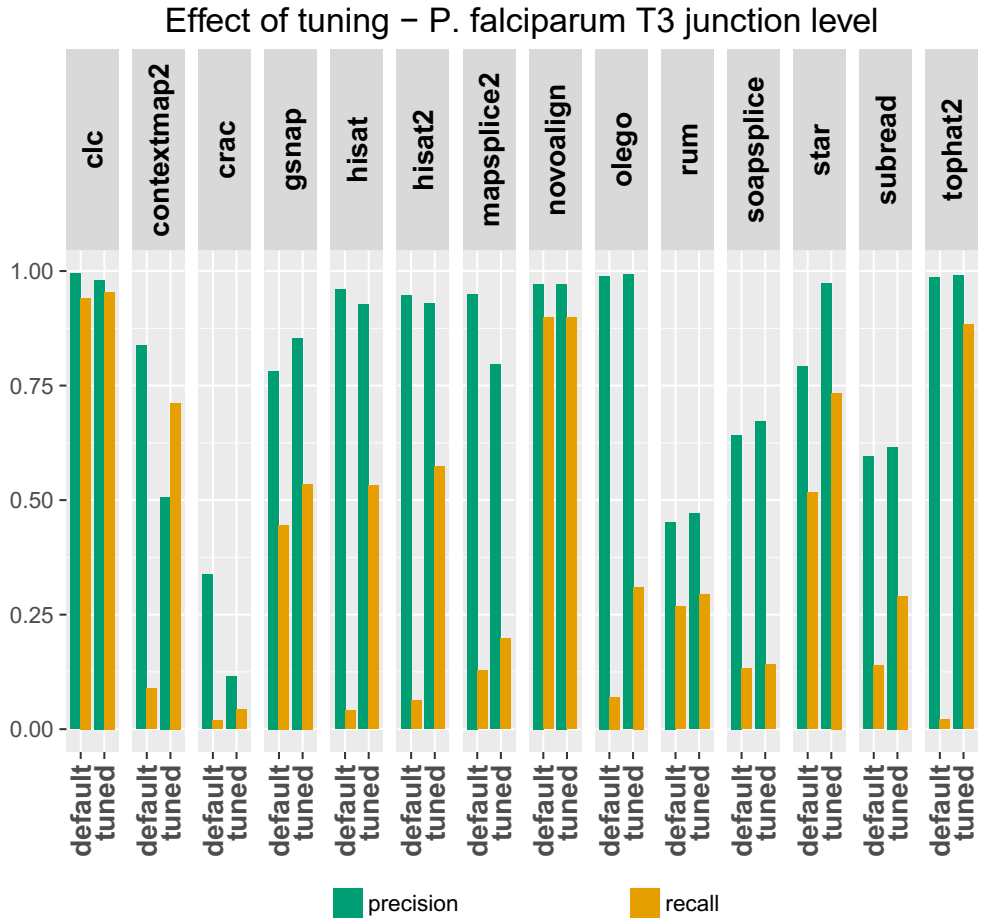

Fig. B.9 *Effect of parameter tweaking at the junction level for the P. falciparum T3 dataset.* For each tool, the figure shows the precision and recall at the junction level for the "default" and the "tuned" alignments. The "tuned" alignment is the best configuration (in terms of junction recall) achieved by the tweaking process.

B.4 Insertions and deletions

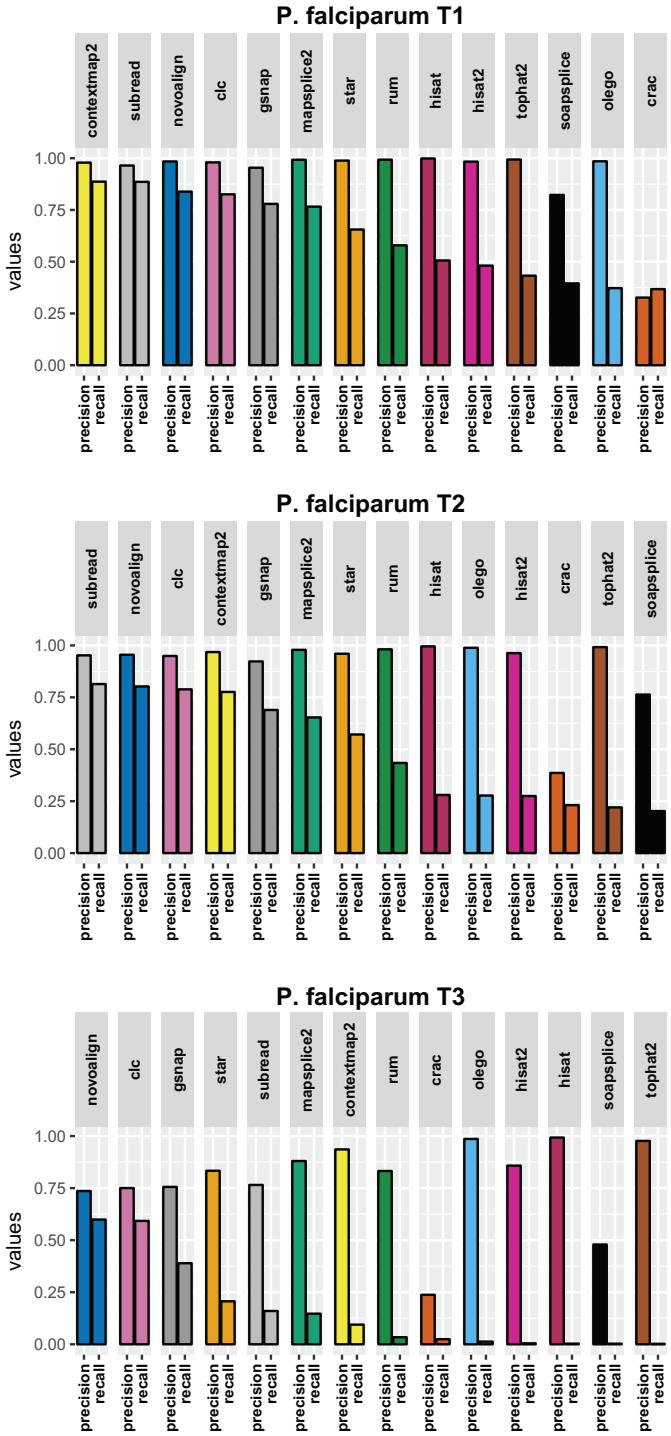

Fig. B.10 *Insertion level precision and recall for the P. falciparum datasets.* For each tool, the figure shows the insertion precision and recall at the base level. The tools are sorted by descending recall.

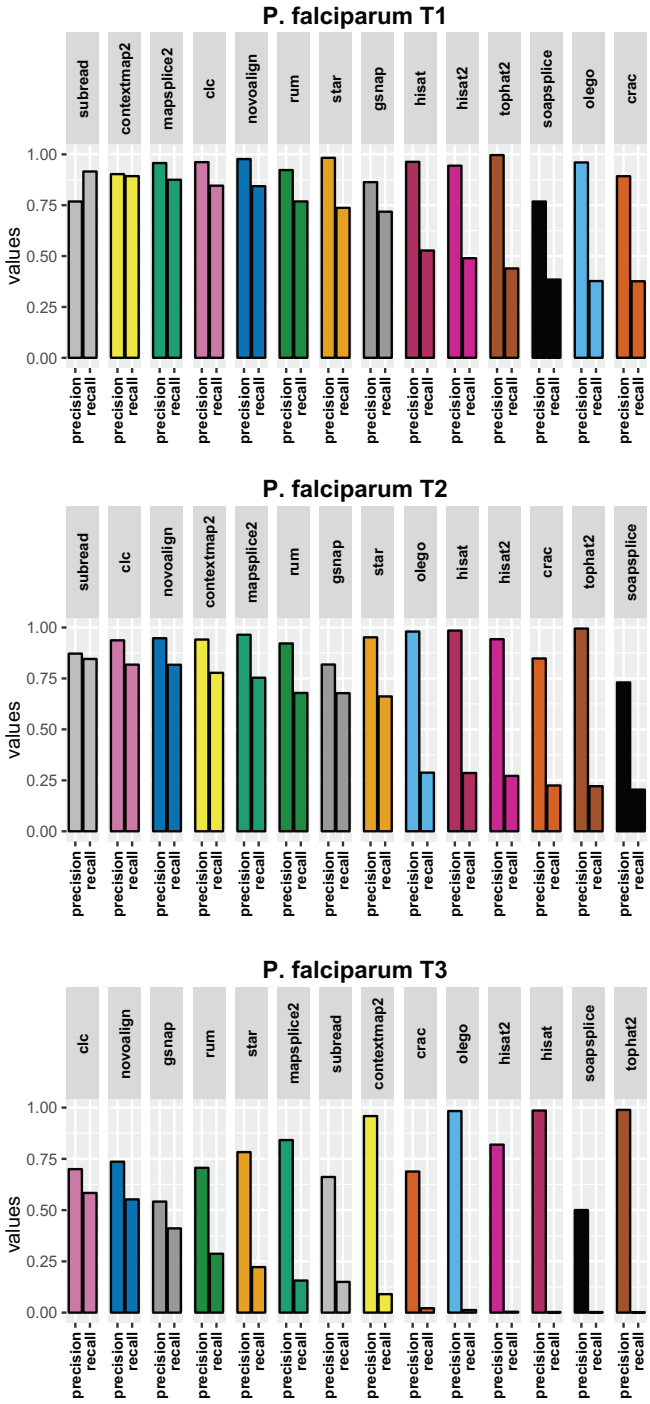

Fig. B.11 *Deletion level precision and recall for the P. falciparum datasets.* For each tool, the figure shows the deletion precision and recall at the base level. The tools are sorted by descending recall.

B.5 Alignment speed and memory requirement

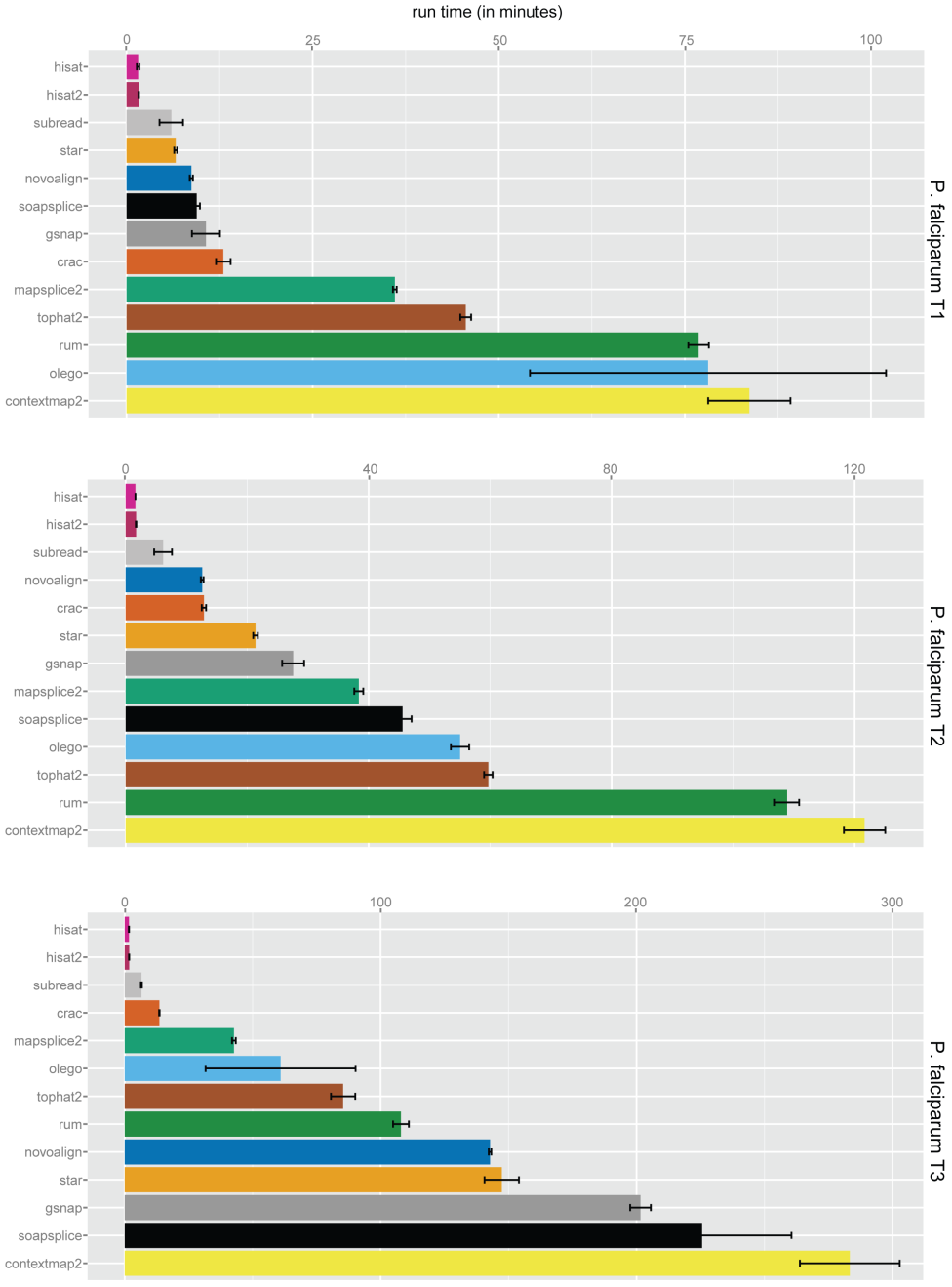

Fig. B.12 *Performance in terms of run time for the P. falciparum datasets.* For each dataset and tool, the bars show the average run time (in minutes) computed on the three replicates. The error bars show the variability of this measure. The tools are sorted by ascending average run time. Note: Novoalign has no multithreading in its free license versions. In order to obtain comparable results, the Novoalign run time was divided by the number of used threads (16).

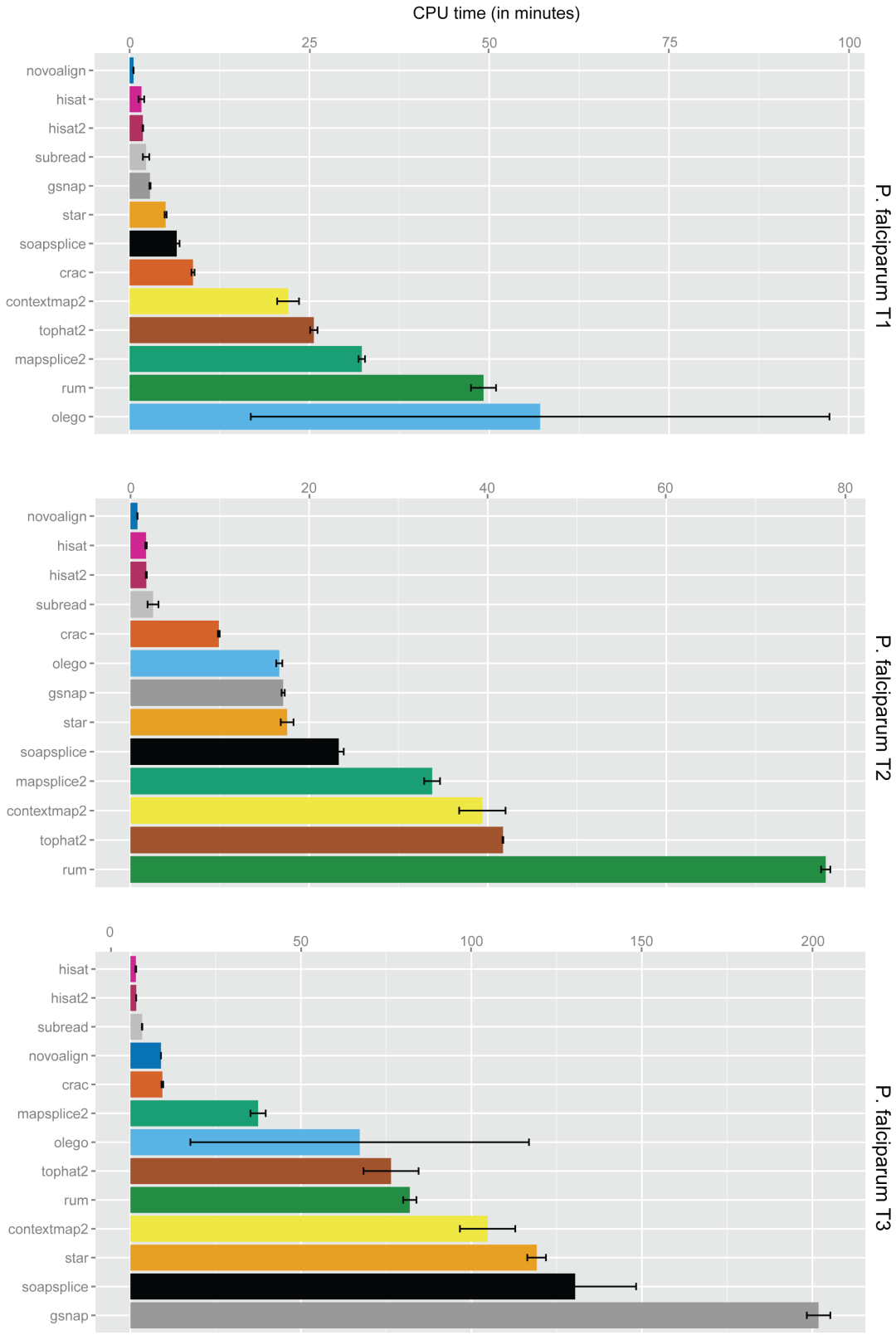

Fig. B.13 *Performance in terms of CPU time for the P. falciparum datasets.* For each tool, the real CPU time was divided by the number of threads used (16). For each dataset and tool, the bars show the average CPU time (in minutes) computed on the three replicates. The error bars show the variability of this measure. The tools are sorted by ascending average CPU time.

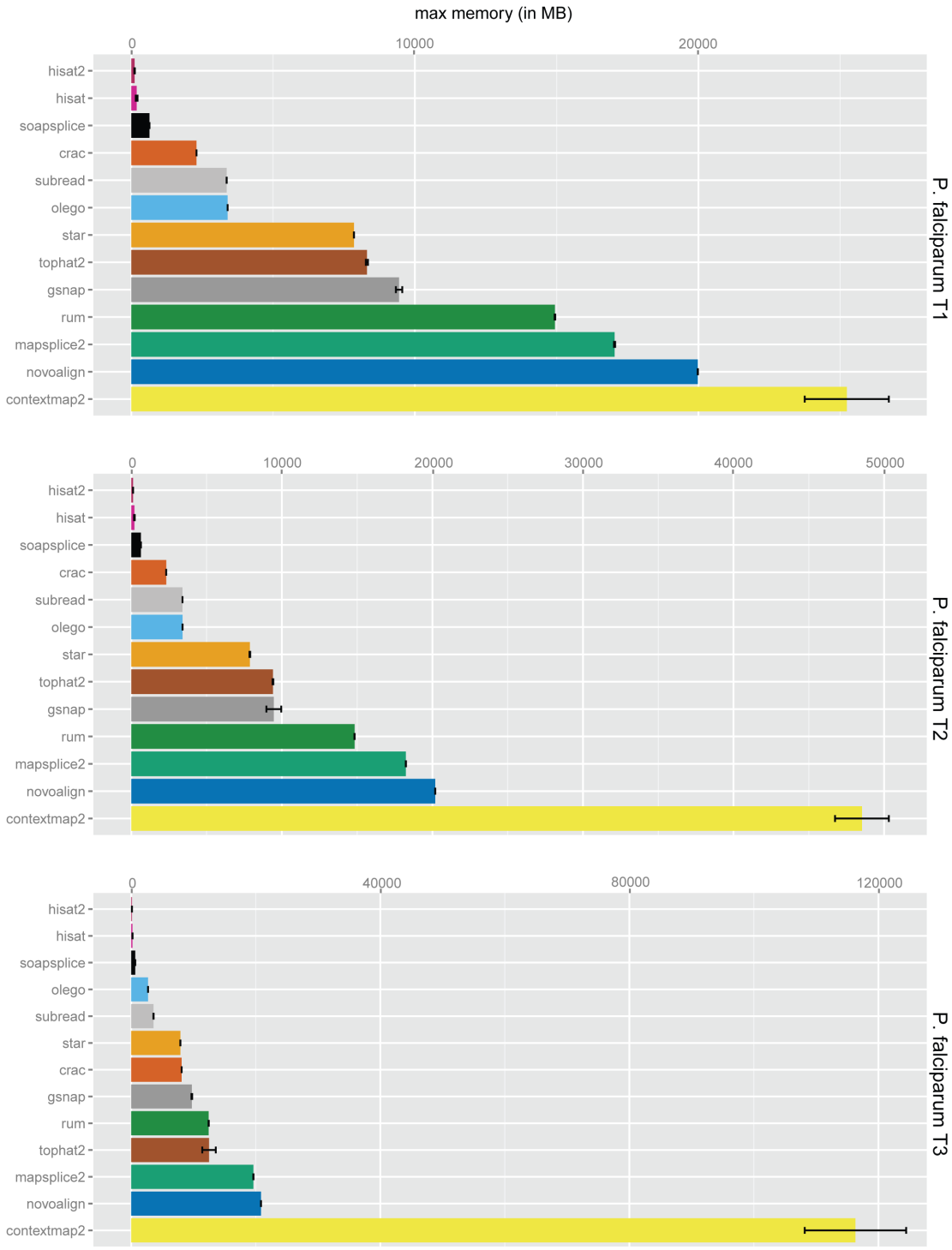

Fig. B.14 *Performance in terms of maximum memory usage for the P. falciparum datasets.* For each dataset and tool, the bars show the average maximum memory usage (in MB) computed on the three replicates. The error bars show the variability of this measure. The tools are sorted by ascending average memory usage.

Appendix C

Parameters tweaking

C.1 CLC Genomic Workbench

Alignment command

For the tweaking, this set of parameters was tested: Mapping, mismatch cost, insertion cost, deletion cost, length fraction, similarity fraction and maximum number of hits for a read.

Tested parameters

For the tweaking, the parameters in Table C.1 were tested. About 20 different parameter combinations were considered during the tweaking process.

Table C.1 CLC Genomic Workbench tweaking parameters and values

Parameters	**Tested values**	**Note**
Mapping	"Map to gene regions only (fast)"; "Also map to inter-genic regions"	Default = "Map to gene regions only (fast)"
mismatch cost	1; 2	Integer, Default = 2, Range [1,3]
insertion cost	1; 3	Integer, Default = 3, Range [1,3]
deletion cost	1; 3	Integer, Default = 3, Range [1,3]
length fraction	0.5; 0.8	Default = 0.8, Range [0,1]
similarity fraction	0.5; 0.8	Default = 0.8, Range [0,1]
maximum number of hits for a read	1; 10; 30	Integer, Default = 10, Range [1,30]

Best configurations

Table C.2 CLC Genomic Workbench best configurations (highest recall) for the base, read and junction level on the human T3 dataset

Parameters	Base level	Read level	Junction level
Mapping	"Also map to inter-genic regions"	"Also map to inter-genic regions"	"Map to gene regions only (fast)"
mismatch cost	2	1	1
insertion cost	3	1	1
deletion cost	3	1	1
length fraction	0.8	0.5	0.5
similarity fraction	0.8	0.5	0.5
maximum number of hits for a read	30	30	30

Table C.3 CLC Genomic Workbench best configurations (highest recall) for the base, read and junction level on the P. falciparum T3 dataset

Parameters	Base level	Read level	Junction level
Mapping	"Map to gene regions only (fast)"	"Map to gene regions only (fast)"	"Map to gene regions only (fast)"
mismatch cost	2	1	1
insertion cost	3	1	1
deletion cost	3	1	1
length fraction	0.8	0.8	0.8
similarity fraction	0.8	0.8	0.5
maximum number of hits for a read	30	30	10

Conclusions

The use of *"maximum number of hits for a read"* greater than or equal to the default (10) seems to improve the results. The main difference between P. falciparum and human is related to the mapping options: the option *"Also map to inter-genic regions"* makes the results slightly worse on P. falciparum while improves them on human. At the base level, the default values for the cost parameters perform significantly better than any other tested values. On the other hand, at the read and junction level the use of cost parameters lower than default results in a recall improvement. Summarizing, the default on P. falciparum is very close to the best achieved. On human, the default is good (compared with the other tools), but the tweaking could still improve the alignment results. In order to balance the performances at the base, read and junction level the default setting plus *"maximum number of hits for a*

read" set at 30 seem a good choice. On human, the option *"Also map to inter-genic regions"* could increase the quality of the results.

C.2 ContextMap2

Alignment command

```
java -Xms16000M -Xmx128000m -XX:+UseConcMarkSweepGC
-XX:NewSize=300M -XX:MaxNewSize=300M
-jar ContextMap_v2.6.0.jar mapper -reads <reads file>
--pairedend -gtf <gtf file> --noncanonicaljunctions
-aligner_name bwa -aligner_bin <bwa path>
-indexer_bin <bwa path> -indices <bwa index>
-genome <genome directory> -o <output path> -t 16
-seed <SEED> -seedmismatches <SEED_MISMATCHES>
-mismatches <MISMATCHES> -mmdiff <MMDIFF>
-maxhits <MAXHITS> -minsize <MINSIZE>
```

Tested parameters

For the tweaking, the parameters in Table C.4 were tested. About 950 different parameter combinations were considered during the tweaking process.

Table C.4 ContextMap2 tweaking parameters and values

Parameters	**Tested values**	**Note**
`-seed <SEED>`	10; 20; 30	Default = 20 (or 30 using Bowtie1)
`-seedmismatches <SEED_MISMATCHES>`	0; 1; 2	Default = 0 (or 1 using Bowtie1)
`-mismatches <MISMATCHES>`	3; 4; 5; 6; 7; 8; 9; 10; 12; 15; 17; 20; 25; 30; 35; 40; 45; 50; 55; 60	Default = 4
`-mmdiff <MMDIFF>`	0; 1; 2	Default = 0
`-maxhits <MAXHITS>`	10; 20; 50	Default = 10 (or 3 using Bowtie2)
`-minsize <MINSIZE>`	5; 10; 15	Default = 10

Best configurations

Table C.5 ContextMap2 best configurations (highest recall) for the base, read and junction level on the human T3 dataset

Parameters	Base level	Read level	Junction level
`-seed <SEED>`	20	20	20
`-seedmismatches <SEED_MISMATCHES>`	0	0	0
`-mismatches <MISMATCHES>`	35	35	35
`-mmdiff <MMDIFF>`	0	0	0
`-maxhits <MAXHITS>`	10	10	10
`-minsize <MINSIZE>`	10	10	10

Table C.6 ContextMap2 best configurations (highest recall) for the base, read and junction level on the P. falciparum T3 dataset

Parameters	Base level	Read level	Junction level
`-seed <SEED>`	20	20	20
`-seedmismatches <SEED_MISMATCHES>`	0	0	0
`-mismatches <MISMATCHES>`	35	35	60
`-mmdiff <MMDIFF>`	0	1	0
`-maxhits <MAXHITS>`	10	10	10
`-minsize <MINSIZE>`	10	10	10

Conclusions

The parameters *SEED_MISMATCHES*, *MMDIFF*, *MAXHITS_VALUES* and *MINSIZE_VALUES* do not affect significantly the results at any level. The default values seem to be a good choice for these parameters. Both organisms highlight the important role of *MISMATCHES* and *SEED*. Increasing the first one and not changing the second one is very beneficial for the majority of metrics. Determining a value that balances the junction level precision and recall seems difficult.

C.3 CRAC

Alignment command

```
crac -i <crac index> -k <K> -r <reads file 1> <reads file 2>
--sam <output sam file> --reads-length <read length>
--no-ambiguity --max-locs <MAX_LOCS>
--min-percent-single-loc <MIN_PERCENT_SINGLE_LOC>
--min-percent-multiple-loc <MIN_PERCENT_MULTIPLE_LOC>
--summary <summary file> --nb-threads 16
```

Tested parameters

For the tweaking, the parameters in Table C.7 were tested. About 440 different parameter combinations were considered during the tweaking process.

Table C.7 CRAC tweaking parameters and values

Parameters	Tested values	Note
`-k <K>`	16; 18; 19; 20; 21; 22; 23; 24; 25; 26; 27	Integer, Default = 22 (for Human)
`--no-ambiguity`	with and without this option	Default = without this option
`--max-locs <MAX_LOCS>`	300; 400; 1000	Default = 300
`--min-percent-single-loc <MIN_PERCENT_SINGLE_LOC>`	0.1; 0.15; 0.2	Default = 0.15
`--min-percent-multiple-loc <MIN_PERCENT_MULTIPLE_LOC>`	0.4; 0.5; 0.6	Default = 0.5

Best configurations

Table C.8 CRAC best configurations (highest recall) for the base, read and junction level on the human T3 dataset

Parameters	Base level	Read level	Junction level
`-k <K>`	20	19	19
`--no-ambiguity`	without this option	without this option	without this option
`--max-locs <MAX_LOCS>`	1000	1000	1000
`--min-percent-single-loc <MIN_PERCENT_SINGLE_LOC>`	0.15	0.15	0.15
`--min-percent-multiple-loc <MIN_PERCENT_MULTIPLE_LOC>`	0.5	0.5	0.5

Table C.9 CRAC best configurations (highest recall) for the base, read and junction level on the P. falciparum T3 dataset

Parameters	Base level	Read level	Junction level
`-k <K>`	18	18	16
`--no-ambiguity`	without this option	without this option	without this option
`--max-locs <MAX_LOCS>`	1000	1000	1000
`--min-percent-single-loc <MIN_PERCENT_SINGLE_LOC>`	0.15	0.15	0.15
`--min-percent-multiple-loc <MIN_PERCENT_MULTIPLE_LOC>`	0.5	0.5	0.5

Conclusions

As stated in the manual, the most important parameter is K. The right value for this parameter depends on the organism more than on the quality of the data. The value K=22 proposed for human seems to be a good choice, even if slightly better results were achieved using K=19 or K=20. On the P. falciparum dataset, the test results suggest a value around 18. The other tested parameters show negligible effects.

C.4 GSNAP

Alignment command

```
gsnap -D <index output path> -d <index name> -A sam
--max-mismatches <MAX_MISMATCHES>
--indel-penalty <INDEL_PENALITY>
--gmap-min-match-length <GMAP_MIN_MATCH_LENGTH>
--pairexpect <PAIR_EXPECT> --pairdev <PAIR_DEV>
--merge-distant-samechr --ordered --novelsplicing 1
--use-splicing <index name>.splicesites
--nthreads 16 --batch 5 --expand-offsets 1
<read file 1> <read file 2> > <output sam file>
```

Tested parameters

For the tweaking, the parameters in Table C.10 were tested. About 630 different parameter combinations were considered during the tweaking process.

Table C.10 GSNAP tweaking parameters and values

Parameters	**Tested values**	**Note**
`--max-mismatches` *<MAX_MISMATCHES>*	2; 3; 4; 5; 6; 7; 8; 9; 10; 15; 20; 25; 30; 35	Default = (readlength + index_interval-1)/kmer - 2 ; should be around 4 for both human and P. falciparum using this formula
`--indel-penalty` *<INDEL_PENALITY>*	1; 2; 3; 4; 5	Default = 2
`--gmap-min-match-length` *<GMAP_MIN_MATCH_LENGTH>*	7; 10; 12; 15; 17; 20; 25	Default = 20
`--pairexpect` *<PAIR_EXPECT>*	default; <fragment length mean>	Default = 200
`--pairdev` *<PAIR_DEV>*	default; <fragment length SD>	Default = 100

Best configurations

Table C.11 GSNAP best configurations (highest recall) for the base, read and junction level on the human T3 dataset

Parameters	**Base level**	**Read level**	**Junction level**
`--max-mismatches` *<MAX_MISMATCHES>*	15	15	2
`--indel-penalty` *<INDEL_PENALITY>*	1	1	5
`--gmap-min-match-length` *<GMAP_MIN_MATCH_LENGTH>*	10	10	15
`--pairexpect` *<PAIR_EXPECT>*	<fragment length mean>	<fragment length mean>	<fragment length mean>
`--pairdev` *<PAIR_DEV>*	<fragment length SD>	<fragment length SD>	<fragment length SD>

Table C.12 GSNAP best configurations (highest recall) for the base, read and junction level on the P. falciparum T3 dataset

Parameters	**Base level**	**Read level**	**Junction level**
`--max-mismatches` *<MAX_MISMATCHES>*	15	20	2
`--indel-penalty` *<INDEL_PENALITY>*	1	1	5
`--gmap-min-match-length` *<GMAP_MIN_MATCH_LENGTH>*	10	7	15
`--pairexpect` *<PAIR_EXPECT>*	<fragment length mean>	<fragment length mean>	<fragment length mean>
`--pairdev` *<PAIR_DEV>*	<fragment length SD>	<fragment length SD>	<fragment length SD>

Conclusions

The important role of *GMAP_MIN_MATCH_LENGTH* and *MAX_MISMATCHES* is confirmed on both P. falciparum and human. On T3 library, using *GMAP_MIN_MATCH_LENGTH* lower than or equal to default with *MAX_MISMATCHES* $\geqslant 15$ allows increasing all the metrics. On P.

falciparum for example, all the metrics have an improvement between 3% and 9% simply increasing *MAX_MISMATCHES* to 15 and decreasing *GMAP_MIN_MATCH_LENGTH* to 15. In addition, the effects of *GMAP_MIN_MATCH_LENGTH* and *MAX_MISMATCHES* seem related, so changing only one of the two parameters did not bring any improvements. The parameters related to fragment length (*PAIR_EXPECT* and *PAIR_DEV*) do not affect the results while a small *INDEL_PENALITY* (lower than or equal to default) slightly improves the base and read level recalls.

C.5 HISAT

Alignment command

```
hisat --threads 16 --time --reorder
--end-to-end -N <NUM_MISMATCH> -L <SEED_LENGTH>
-i S,1,<SEED_INTERVAL> -D <SEED_EXTENSION> -R <RE_SEED>
--pen-noncansplice <PENALITY_NONCANONICAL>
--mp <MAX_MISMATCH_PENALITY>,<MIN_MISMATCH_PENALITY>
--known-splicesite-infile <genome name>.splicesites.txt
--novel-splicesite-outfile splicesites.novel.txt
--novel-splicesite-infile splicesites.novel.txt
-f -x <index name> -1 <read file 1> -2 <read file 2>
-S <output sam file>
```

Tested parameters

For the tweaking, the parameters in Table C.13 were tested. About 820 different parameter combinations were considered during the tweaking process.

Table C.13 HISAT tweaking parameters and values

Parameters	**Tested values**	**Note**
`--end-to-end`	with and without this option	Default = without this option
`-N <NUM_MISMATCH>`	0; 1	Integer, Default = 0, Range = [0,1]
`-L <SEED_LENGTH>`	15; 18; 20; 22	Integer, Default = 22
`-i S,1,<SEED_INTERVAL>`	0.25; 0.5; 0.75; 1.15	Default = 1.15
`-D <SEED_EXTENSION>`	15; 20; 25	Default = 15
`-R <RE_SEED>`	2; 3; 5	Default = 2
`--pen-noncansplice <PENALITY_NONCANONICAL>`	0; 3; 12; 20; 30	Default = 3
`--mp <MAX_MISMATCH_PENALITY>, ...`	1; 3; 6	Default = 6
`--mp ..., <MIN_MISMATCH_PENALITY>`	0; 1; 2	Default = 2

Best configurations

Table C.14 HISAT best configurations (highest recall) for the base, read and junction level on the human T3 dataset

Parameters	**Base level**	**Read level**	**Junction level**
`--end-to-end`	without this option	without this option	without this option
`-N <NUM_MISMATCH>`	0	0	0
`-L <SEED_LENGTH>`	20	20	20
`-i S,1,<SEED_INTERVAL>`	1.15	1.15	1.15
`-D <SEED_EXTENSION>`	15	15	15
`-R <RE_SEED>`	2	2	2
`--pen-noncansplice <PENALITY_NONCANONICAL>`	3	3	20
`--mp <MAX_MISMATCH_PENALITY>, ...`	1	1	1
`--mp ..., <MIN_MISMATCH_PENALITY>`	0	0	0

Table C.15 HISAT best configurations (highest recall) for the base, read and junction level on the P. falciparum T3 dataset

Parameters	Base level	Read level	Junction level
`--end-to-end`	without this option	without this option	without this option
`-N <NUM_MISMATCH>`	0	0	0
`-L <SEED_LENGTH>`	20	20	20
`-i S,1,<SEED_INTERVAL>`	1.15	1.15	1.15
`-D <SEED_EXTENSION>`	15	15	15
`-R <RE_SEED>`	2	2	2
`--pen-noncansplice <PENALITY_NONCANONICAL>`	20	20	20
`--mp <MAX_MISMATCH_PENALITY>, ...`	1	1	1
`--mp ..., <MIN_MISMATCH_PENALITY>`	0	0	0

Conclusions

Tests suggest that *MAX_MISMATCH_PENALITY* and *MIN_MISMATCH_PENALITY* have a very important role in the quality of alignment. Both on human and P. falciparum, there is a significant improvement on all metrics setting these parameters properly: using a small value for *MAX_MISMATCH_PENALITY* and *MIN_MISMATCH_PENALITY* (1 and 0/1, respectively), the base and read level recalls increase more than 60% while the junction recall increases more than 45%. Moreover, a value of *PENALITY_NONCANONICAL* greater than the default seems to slightly improve the results. The Bowtie2-like parameters (*NUM_MISMATCH*, *SEED_LENGTH*, *SEED_INTERVAL*, *SEED_EXTENSION* and *RE_SEED)* seem have no effects on the quality of the results. The use of Bowtie2-like parameter `--end-to-end` does not change the results.

C.6 HISAT2

Alignment command

```
hisat2 --threads 16 --time --reorder
--end-to-end -N <NUM_MISMATCH> -L <SEED_LENGTH>
-i S,1,<SEED_INTERVAL> -D <SEED_EXTENSION> -R <RE_SEED>
--pen-noncansplice <PENALITY_NONCANONICAL>
--mp <MAX_MISMATCH_PENALITY>,<MIN_MISMATCH_PENALITY>
--sp <MAX_SOFTCLIPPING_PENALITY>,<MIN_SOFTCLIPPING_PENALITY>
--known-splicesite-infile <genome name>.splicesites.txt
--novel-splicesite-outfile splicesites.novel.txt
--novel-splicesite-infile splicesites.novel.txt
```

`-f -x <index name> -1 <read file 1> -2 <read file 2>`
`-S <output sam file>`

Tested parameters

For the tweaking, the parameters in Table C.16 were tested. About 830 different parameter combinations were considered during the tweaking process.

Table C.16 HISAT2 tweaking parameters and values

Parameters	**Tested values**	**Note**
`--end-to-end`	with and without this option	Default = without this option
`-N <NUM_MISMATCH>`	0; 1	Integer, Default = 0, Range = [0,1]
`-L <SEED_LENGTH>`	15; 18; 20; 22	Integer, Default = 22
`-i S,1,<SEED_INTERVAL>`	0.25; 0.5; 0.75; 1.15	Default = 1.15
`-D <SEED_EXTENSION>`	15; 20; 25	Default = 15
`-R <RE_SEED>`	2; 3; 5	Default = 2
`--pen-noncansplice <PENALITY_NONCANONICAL>`	0; 3; 12; 20	Default = 3
`--mp <MAX_MISMATCH_PENALITY>, ...`	1; 2; 3; 6	Default = 6
`--mp ..., <MIN_MISMATCH_PENALITY>`	0; 1; 2	Default = 2
`--sp <MAX_SOFTCLIPPING_PENALITY>, ...`	1; 2; 3	Default = 2
`--sp ..., <MIN_SOFTCLIPPING_PENALITY>`	0; 1; 2	Default = 1

Best configurations

Table C.17 HISAT2 best configurations (highest recall) for the base, read and junction level on the human T3 dataset

Parameters	**Base level**	**Read level**	**Junction level**
`--end-to-end`	without this option	without this option	without this option
`-N <NUM_MISMATCH>`	1	1	1
`-L <SEED_LENGTH>`	20	20	20
`-i S,1,<SEED_INTERVAL>`	0.5	0.5	0.5
`-D <SEED_EXTENSION>`	25	25	25
`-R <RE_SEED>`	5	5	5
`--pen-noncansplice <PENALITY_NONCANONICAL>`	12	12	12
`--mp <MAX_MISMATCH_PENALITY>, ...`	1	1	1
`--mp ..., <MIN_MISMATCH_PENALITY>`	0	0	0
`--sp <MAX_SOFTCLIPPING_PENALITY>, ...`	3	3	3
`--sp ..., <MIN_SOFTCLIPPING_PENALITY>`	0	0	0

Table C.18 HISAT2 best configurations (highest recall) for the base, read and junction level on the P. falciparum T3 dataset

Parameters	Base level	Read level	Junction level
`--end-to-end`	without this option	without this option	without this option
`-N <NUM_MISMATCH>`	1	1	0
`-L <SEED_LENGTH>`	20	20	22
`-i S,1,<SEED_INTERVAL>`	0.5	0.5	1.15
`-D <SEED_EXTENSION>`	25	25	15
`-R <RE_SEED>`	5	5	2
`--pen-noncansplice <PENALITY_NONCANONICAL>`	20	20	20
`--mp <MAX_MISMATCH_PENALITY>, ...`	1	1	1
`--mp ..., <MIN_MISMATCH_PENALITY>`	0	0	0
`--sp <MAX_SOFTCLIPPING_PENALITY>, ...`	3	3	2
`--sp ..., <MIN_SOFTCLIPPING_PENALITY>`	0	0	1

Conclusions

The behavior of HISAT2 parameters is very similar to HISAT: *MAX_MISMATCH_PENALITY* and *MIN_MISMATCH_PENALITY* are the most important parameters at the base and read level. Again, a value of *PENALITY_NONCANONICAL* greater than the default seems to slightly improve the results. At any level, using a *MAX_SOFTCLIPPING_PENALITY* greater than or equal to default (e.g. 2 or 3) slightly improves the results while *MIN_SOFTCLIPPING_PENALITY* seems to have no effect on the results. The Bowtie2 parameters (*NUM_MISMATCH*, *SEED_LENGTH*, *SEED_INTERVAL*, *SEED_EXTENSION* and *RE_SEED)* seem have no effects on the alignment. The use of Bowtie2 parameter `--end-to-end` does not change the results.

C.7 MapSplice2

Alignment command

```
python mapsplice.py --threads 16
--non-canonical-double-anchor
--min-map-len <MIN_MAP_LENGTH> --splice-mis <SPLICE_MISMATCHES>
--max-append-mis <APPEND_MISMATCHES> --ins <INSERTION_LENGTH>
--del <DELETION_LENGTH> --filtering <FILTER>
--output <output path> -c <genome fasta files>
-x <index name> -1 <read file 1> -2 <read file 2>
```

Tested parameters

For the tweaking, the parameters in Table C.19 were tested. About 1080 different parameter combinations were considered during the tweaking process.

Table C.19 MapSplice2 tweaking parameters and values

Parameters	**Tested values**	**Note**
`--min-map-len` *<MIN_MAP_LENGTH>*	15; 20; 25; 33; 50; 66; 75	Default = 50
`--splice-mis` *<SPLICE_MISMATCHES>*	0; 1; 2	Integer, Range = [0,2], Default = 1
`--max-append-mis` *<APPEND_MISMATCHES>*	0; 1; 2; 3	Integer, Range = [0,3], Default = 3
`---ins` *<INSERTION_LENGTH>*	6; 8; 10	Integer, Range = [0,10], Default = 6
`--del` *<DELETION_LENGTH>*	6; 8; 10	Integer, Range = [0,10], Default = 6
`--filtering` *<FILTER>*	1; 2	Integer, Range = [1,2], Default = 1

Best configurations

Table C.20 MapSplice2 best configurations (highest recall) for the base, read and junction level on the human T3 dataset

Parameters	**Base level**	**Read level**	**Junction level**
`--min-map-len` *<MIN_MAP_LENGTH>*	25	25	25
`--splice-mis` *<SPLICE_MISMATCHES>*	2	0	2
`--max-append-mis` *<APPEND_MISMATCHES>*	3	3	1
`--ins` *<INSERTION_LENGTH>*	10	6	10
`--del` *<DELETION_LENGTH>*	6	10	6
`--filtering` *<FILTER>*	1	2	1

Table C.21 MapSplice2 best configurations (highest recall) for the base, read and junction level on the P. falciparum T3 dataset

Parameters	Base level	Read level	Junction level
`--min-map-len <MIN_MAP_LENGTH>`	25	25	25
`--splice-mis <SPLICE_MISMATCHES>`	2	0	2
`--max-append-mis <APPEND_MISMATCHES>`	3	3	1
`--ins <INSERTION_LENGTH>`	6	6	10
`--del <DELETION_LENGTH>`	6	6	6
`--filtering <FILTER>`	1	1	1

Conclusions

MIN_MAP_LENGTH, *SPLICE_MISMATCHES* and *APPEND_MISMATCHES* are the most important parameters at the read and junction level. At the base level, the tweaking of the parameters does not result in any significant improvement. *INSERTION_LENGTH*, *DELETION_LENGTH* and *FILTER* seem to have negligible effects at all levels.

C.8 Novoalign

Alignment command

```
novoalign -d <output index file> -f <read file 1> <read file 2>
-F FA -o SAM -r All 10
-i PE <FRAGMENT_LENGTH_MEAN>,<FRAGMENT_LENGTH_SD>
-t <A_SCORE>,<B_SCORE> -h -1 -1
-v 0 70 70 "[>]([^:]*)" > <output sam file> 2>alignment.log
```

Tested parameters

For the tweaking, the parameters in Table C.22 were tested. About 16 different parameter combinations were considered during the tweaking process.

Table C.22 Novoalign tweaking parameters and values

Parameters	Tested values	Note
`-t <A_SCORE>,...`	default; 10; 12; 13; 20	Default = log4(N), where N is the reference genome length
`-t ...,<B_SCORE>`	2; 3; 4.5	Default = 4.5
`-h -1 -1`	with and without this option	Default = without this option
`-i PE <FRAGMENT_LENGTH_MEAN>, <FRAGMENT_LENGTH_SD>`	with and without this option	Default <FRAGMENT_LENGTH_MEAN> = 250; Default <FRAGMENT_LENGTH_SD> = 50

Best configurations

Table C.23 Novoalign best configurations (highest recall) for the base, read and junction level on the human T3 dataset

Parameters	Base level	Read level	Junction level
`-t <A_SCORE>,...`	10	10	10
`-t ...,<B_SCORE>`	4.5	4.5	4.5
`-h -1 -1`	with this option	with this option	with this option
`-i PE <FRAGMENT_LENGTH_MEAN>, <FRAGMENT_LENGTH_SD>`	without this option	without this option	without this option

Table C.24 Novoalign best configurations (highest recall) for the base, read and junction level on the P. falciparum T3 dataset

Parameters	Base level	Read level	Junction level
`-t <A_SCORE>,...`	default	default	default
`-t ...,<B_SCORE>`	default	default	default
`-h -1 -1`	without this option	without this option	without this option
`-i PE <FRAGMENT_LENGTH_MEAN>, <FRAGMENT_LENGTH_SD>`	without this option	without this option	with this option

Conclusions

The default setting achieves very good results, comparable with the best tested configuration. Decreasing *B_SCORE* from the default value results in an improvement in the junction

precision while all the other metrics result in worse performance. Given a fixed value of *B_SCORE*, decreasing *A_SCORE* seems to slightly improve the results at any levels. The other tested options show negligible effects on the results.

C.9 Olego

Alignment command

```
olego --output-file output_1.sam --num-threads 16
--regression-model <regression model> --verbose
--max-total-diff <TOTAL_DIFF> --allow-rep-anchor
--word-size <WORD_SIZE> --max-word-diff <MAX_WORD_DIFF>
--word-max-overlap <WORD_MAX_OVERLAP>
--min-anchor <MIN_ANCHOR>
--junction-file <junction file> <olego index> <read file 1>

olego --output-file output_2.sam --num-threads 16
--regression-model <regression model> --verbose
--max-total-diff <TOTAL_DIFF> --allow-rep-anchor
--word-size <WORD_SIZE> --max-word-diff <MAX_WORD_DIFF>
--word-max-overlap <WORD_MAX_OVERLAP>
--min-anchor <MIN_ANCHOR>
--junction-file <junction file> <olego index> <read file 2>

perl mergePEsam.pl -v output_1.sam output_2.sam output.sam
```

Tested parameters

For the tweaking, the parameters in Table C.25 were tested. About 1000 different parameter combinations were considered during the tweaking process.

Table C.25 Olego tweaking parameters and values

Parameters	Tested values	Note
`--max-total-diff <TOTAL_DIFF>`	4; 5; 6; 7; 8; 9; 10	Float or Integer; Default = 0.06
`--word-size <WORD_SIZE>`	13; 14; 15; 16	Default = 15
`--max-word-diff <MAX_WORD_DIFF>`	0; 1; 2	Default = 0
`--word-max-overlap <WORD_MAX_OVERLAP>`	0; 1	Default = 1
`--allow-rep-anchor`	with or without this option	Default = without this option
`--min-anchor <MIN_ANCHOR>`	5; 8; 11	Default = 8

Best configurations

Table C.26 Olego best configurations (highest recall) for the base, read and junction level on the human T3 dataset

Parameters	Base level	Read level	Junction level
`--max-total-diff <TOTAL_DIFF>`	10	10	10
`--word-size <WORD_SIZE>`	13	13	13
`--max-word-diff <MAX_WORD_DIFF>`	2	2	2
`--word-max-overlap <WORD_MAX_OVERLAP>`	1	1	1
`--allow-rep-anchor`	without this option	without this option	without this option
`--min-anchor <MIN_ANCHOR>`	11	11	11

Table C.27 Olego best configurations (highest recall) for the base, read and junction level on the P. falciparum T3 dataset

Parameters	Base level	Read level	Junction level
`--max-total-diff <TOTAL_DIFF>`	10	10	10
`--word-size <WORD_SIZE>`	14	14	13
`--max-word-diff <MAX_WORD_DIFF>`	2	2	2
`--word-max-overlap <WORD_MAX_OVERLAP>`	1	1	1
`--allow-rep-anchor`	without this option	without this option	without this option
`--min-anchor <MIN_ANCHOR>`	11	11	11

Conclusions

Both human and P. falciparum take advantage from the tuning of some parameters. Increasing *MAX_WORD_DIFF*, *TOTAL_DIFF* and *MIN_ANCHOR* results in an improvement on all metrics. At any level, the parameter tweaking on Olego seems to not bring a significant improvement: the best results achieved after the tweaking are the worse compared with the tuning of the other tools. In conclusion, Olego seems not able to manage very complex datasets.

C.10 RUM

Alignment command

```
rum_runner align --index-dir <index directory> --name <job name>
--output <output path> --chunks 16 <read file 1> <read file 2>
--verbose --preserve-names
--blat-min-identity <BLAT_MIN_IDENTITY>
--blat-rep-match <BLAT_REP_MATCH>
--blat-step-size <BLAT_STEP_SIZE>
--blat-tile-size <BLAT_TILE_SIZE>
```

Tested parameters

For the tweaking, the parameters in Table C.28 were tested. About 750 different parameter combinations were considered during the tweaking process.

Table C.28 RUM tweaking parameters and values

Parameters	Tested values	Note
`--blat-min-identity <BLAT_MIN_IDENTITY>`	75; 88; 90; 93; 95; 98	Default = 93
`--blat-rep-match <BLAT_REP_MATCH>`	256; 1024; 4096	Default = 256
`--blat-step-size <BLAT_STEP_SIZE>`	6; 8; 10; 11; 12; 13; 14	Default = 6
`--blat-tile-size <BLAT_TILE_SIZE>`	8; 10; 11; 12; 13; 14	Default = 12

Best configurations

Table C.29 RUM best configurations (highest recall) for the base, read and junction level on the human T3 dataset

Parameters	Base level	Read level	Junction level
`--blat-min-identity <BLAT_MIN_IDENTITY>`	93	93	93
`--blat-rep-match <BLAT_REP_MATCH>`	256	256	256
`--blat-step-size <BLAT_STEP_SIZE>`	6	6	6
`--blat-tile-size <BLAT_TILE_SIZE>`	12	12	12

Table C.30 RUM best configurations (highest recall) for the base, read and junction level on the P. falciparum T3 dataset

Parameters	Base level	Read level	Junction level
`--blat-min-identity <BLAT_MIN_IDENTITY>`	88	88	75
`--blat-rep-match <BLAT_REP_MATCH>`	4096	4096	4096
`--blat-step-size <BLAT_STEP_SIZE>`	10	10	6
`--blat-tile-size <BLAT_TILE_SIZE>`	8	8	8

Conclusions

The parameter *BLAT_MIN_IDENTITY* has a major role in the quality of the alignment. On P. falciparum, a value lower than default shows better results in terms of recall. The parameter values seem more related to the particular genome than the quality of the data. Indeed, applying the best P. falciparum configurations on human there is no improvement on the metrics.

C.11 SOAPsplice

Alignment command

```
soapsplice -d <index> -1 <read file 1> -2 <read file 2>
-o <output file> -p 16 -f 2 -l 0 -I <FRAGMENT_LENGTH_MEAN>
-m <MISMATCHES> -g <INDEL> -i <TAIL> -a <SHORT_LENGTH>
```

Tested parameters

For the tweaking, the parameters in Table C.31 were tested. About 460 different parameter combinations were considered during the tweaking process.

Table C.31 SOAPsplice tweaking parameters and values

Parameters	Tested values	Note
`-m <MISMATCHES>`	0; 1; 2; 3; 4; 5	Range = [0, 5]; Default = 3
`-g <INDEL>`	0; 1; 2	Range = [0, 2]; Default = 2
`-i <TAIL>`	5; 7; 10; 25; 33; 42; 50; 58; 66; 75	Default = 7
`-a <SHORT_LENGTH>`	6; 8; 10; 20; 25	Default = 8

Best configurations

Table C.32 SOAPsplice best configurations (highest recall) for the base, read and junction level on the human T3 dataset

Parameters	Base level	Read level	Junction level
`-m <MISMATCHES>`	5	5	5
`-g <INDEL>`	2	2	1
`-i <TAIL>`	33	66	5
`-a <SHORT_LENGTH>`	8	8	8

Table C.33 SOAPsplice best configurations (highest recall) for the base, read and junction level on the P. falciparum T3 dataset

Parameters	Base level	Read level	Junction level
`-m <MISMATCHES>`	5	5	5
`-g <INDEL>`	2	2	1
`-i <TAIL>`	33	66	5
`-a <SHORT_LENGTH>`	8	8	8

Conclusions

MISMATCHES seems to be the most influential parameter. Increasing the value of *MISMATCHES* results in a higher recall at the read and base level, while the junction recall seems to be mainly influenced by *SHORT_LENGTH*. Additionally, the parameter *TAIL* plays an important role, especially at the read and base level. With regards to *INDEL*, the default value shows the best results.

C.12 STAR

Alignment command

```
STAR --runThreadN 16 --genomeDir <index path>
--readFilesIn <read file 1> <read file 2>
--outFileNamePrefix <output alignment prefix>
--twopassMode Basic --outSAMunmapped Within
```

`--limitOutSJcollapsed` *<NUM_COLLAPSED_JUNCTIONS>*
`--limitSjdbInsertNsj` *<NUM_INSERTED_JUNCTIONS>*
`--outFilterMultimapNmax` *<NUM_MULTIMAPPER>*
`--outFilterMismatchNmax` *<NUM_FILTER_MISMATCHES>*
`--outFilterMismatchNoverLmax` *<RATIO_FILTER_MISMATCHES>*
`--seedSearchStartLmax` *<SEED_LENGTH>*
`--alignSJoverhangMin` *<OVERHANG>*
`--alignEndsType` *<END_ALIGNMENT_TYPE>*
`--outFilterMatchNminOverLread` *<NUM_FILTER_MATCHES>*
`--outFilterScoreMinOverLread` *<NUM_FILTER_SCORE>*
`--winAnchorMultimapNmax` *<NUM_ANCHOR>*
`--alignSJDBoverhangMin` *<OVERHANG_ANNOTATED>*
`--outFilterType` *<OUT_FILTER>*

Tested parameters

For the tweaking, the parameters in Table C.34 were tested. About 2600 different parameter combinations were considered during the tweaking process.

Table C.34 STAR tweaking parameters and values

Parameters	Tested values	Note
`--limitOutSJcollapsed` *<NUM_COLLAPSED_JUNCTIONS>*	1000000; 5000000	Default = 1000000
`--limitSjdbInsertNsj` *<NUM_INSERTED_JUNCTIONS>*	1000000; 5000000	Default = 1000000
`--outFilterMultimapNmax` *<NUM_MULTIMAPPER>*	10; 100	Default = 10
`--outFilterMismatchNmax` *<NUM_FILTER_MISMATCHES>*	3; 5; 8; 10; 20; 25; 33	Default = 10
`--outFilterMismatchNoverLmax` *<RATIO_FILTER_MISMATCHES>*	0.3; 1	Default = 0.3
`--seedSearchStartLmax` *<SEED_LENGTH>*	12; 30; 33; 50	Default = 50
`--alignSJoverhangMin` *<OVERHANG>*	3; 5; 8; 15	Default = 5
`--alignEndsType` *<END_ALIGNMENT_TYPE>*	“Local”; “EndToEnd”; “Extend5pOfRead1”; “Extend3pOfRead1”	Default = “Local”
`--outFilterMatchNminOverLread` *<NUM_FILTER_MATCHES>*	0; 0.66	Default = 0.66
`--outFilterScoreMinOverLread` *<NUM_FILTER_SCORE>*	0.3; 0.66	Default = 0.66
`--winAnchorMultimapNmax` *<NUM_ANCHOR>*	50; 200	Default = 50
`--alignSJDBoverhangMin` *<OVERHANG_ANNOTATED>*	1; 3	Default = 3
`--outFilterType` *<OUT_FILTER>*	“Normal”; “BySJout”	Default = “Normal”

Best configurations

Table C.35 STAR best configurations (highest recall) for the base, read and junction level on the human T3 dataset

Parameters	**Base level**	**Read level**	**Junction level**
`--limitOutSJcollapsed` *<NUM_COLLAPSED_JUNCTIONS>*	1000000	1000000	1000000
`--limitSjdbInsertNsj` *<NUM_INSERTED_JUNCTIONS>*	1000000	1000000	1000000
`--outFilterMultimapNmax` *<NUM_MULTIMAPPER>*	100	100	100
`--outFilterMismatchNmax` *<NUM_FILTER_MISMATCHES>*	33	33	33
`--outFilterMismatchNoverLmax` *<RATIO_FILTER_MISMATCHES>*	0.3	0.3	0.3
`--seedSearchStartLmax` *<SEED_LENGTH>*	12	12	12
`--alignSJoverhangMin` *<OVERHANG>*	15	15	15
`--alignEndsType` *<END_ALIGNMENT_TYPE>*	“Local”	"Local"	"Local"
`--outFilterMatchNminOverLread` *<NUM_FILTER_MATCHES>*	0	0	0
`--outFilterScoreMinOverLread` *<NUM_FILTER_SCORE>*	0.3	0.3	0.3
`--winAnchorMultimapNmax` *<NUM_ANCHOR>*	50	50	50
`--alignSJDBoverhangMin` *<OVERHANG_ANNOTATED>*	3	3	3
`--outFilterType` *<OUT_FILTER>*	“BySJout”	“BySJout”	“BySJout”

Table C.36 STAR best configurations (highest recall) for the base, read and junction level on the P. falciparum T3 dataset

Parameters	Base level	Read level	Junction level
`--limitOutSJcollapsed` *<NUM_COLLAPSED_JUNCTIONS>*	1000000	1000000	1000000
`--limitSjdbInsertNsj` *<NUM_INSERTED_JUNCTIONS>*	1000000	1000000	1000000
`--outFilterMultimapNmax` *<NUM_MULTIMAPPER>*	100	100	100
`--outFilterMismatchNmax` *<NUM_FILTER_MISMATCHES>*	33	33	33
`--outFilterMismatchNoverLmax` *<RATIO_FILTER_MISMATCHES>*	0.3	0.3	0.3
`--seedSearchStartLmax` *<SEED_LENGTH>*	12	30	12
`--alignSJoverhangMin` *<OVERHANG>*	15	15	15
`--alignEndsType` *<END_ALIGNMENT_TYPE>*	“Local”	"Local"	"Local"
`--outFilterMatchNminOverLread` *<NUM_FILTER_MATCHES>*	0	0	0
`--outFilterScoreMinOverLread` *<NUM_FILTER_SCORE>*	0.3	0.3	0.3
`--winAnchorMultimapNmax` *<NUM_ANCHOR>*	50	50	50
`--alignSJDBoverhangMin` *<OVERHANG_ANNOTATED>*	3	3	1
`--outFilterType` *<OUT_FILTER>*	“BySJout”	“BySJout”	“BySJout”

Conclusions

The default options of STAR achieve one of the best results on the T3 datasets. However, tweaking some parameters is possible to further improve the metrics. The important roles of *NUM_FILTER_MISMATCHES*, *END_ALIGNMENT_TYPE*, *OVERHANG*, *NUM_FILTER_SCORE*, *SEED_LENGTH* are confirmed on both human and P. falciparum. Increasing the number of allowed mismatches and leaving the default value for *END_ALIGNMENT_TYPE* improves the results. At the same time, increasing *OVERHANG* and decreasing *SEED_LENGTH* and *NUM_FILTER_SCORE* increases the recall at all levels.

C.13 Subread(Subjunc)

Alignment command

```
subjunc -i <index> -r <read file 1> -R <read file 2>
-T 16 --allJunctions --SAMoutput -o <output alignment>
-d <MIN_FRAGMENT_LENGTH> -I <INDEL> -m <NUM_HIT_SUBREADS>
-M <MISMATCHES> -n <NUM_EXTRACTED_SUBREADS>
-p <NUM_HIT_PAIR_SUBREADS> --complexIndels
```

Tested parameters

For the tweaking, the parameters in Table C.37 were tested. About 1060 different parameter combinations were considered during the tweaking process.

Table C.37 Subread tweaking parameters and values

Parameters	Tested values	Note
`-d <MIN_FRAGMENT_LENGTH>`	0; 50	Default = 50
`-I <INDEL>`	3; 5; 8; 10; 15; 20	Default = 5
`-m <NUM_HIT_SUBREADS>`	1; 3; 5	Default = 3
`-M <MISMATCHES>`	3; 5; 8; 10; 20; 30	Default = 3
`-n <NUM_EXTRACTED_SUBREADS>`	5; 10; 15	Default = 10
`-p <NUM_HIT_PAIR_SUBREADS>`	1; 3	Default = 1
`--complexIndels`	With and without this option	Default = without this option

Best configurations

Table C.38 Subread best configurations (highest recall) for the base, read and junction level on the human T3 dataset

Parameters	Base level	Read level	Junction level
`-d <MIN_FRAGMENT_LENGTH>`	0	0	0
`-I <INDEL>`	10	10	10
`-m <NUM_HIT_SUBREADS>`	1	1	1
`-M <MISMATCHES>`	20	20	20
`-n <NUM_EXTRACTED_SUBREADS>`	15	5	15
`-p <NUM_HIT_PAIR_SUBREADS>`	1	1	1
`--complexIndels`	without this option	with this option	without this option

Table C.39 Subread best configurations (highest recall) for the base, read and junction level on the P. falciparum T3 dataset

Parameters	Base level	Read level	Junction level
`-d <MIN_FRAGMENT_LENGTH>`	0	50	50
`-I <INDEL>`	10	5	5
`-m <NUM_HIT_SUBREADS>`	1	1	1
`-M <MISMATCHES>`	20	30	30
`-n <NUM_EXTRACTED_SUBREADS>`	5	5	15
`-p <NUM_HIT_PAIR_SUBREADS>`	1	1	1
`--complexIndels`	with this option	without this option	without this option

Conclusions

The tweaking of the parameters allows to considerably increase the alignment quality, both on P. falciparum and human. The most influential parameters are *MISMATCHES*, *NUM_HIT_SUBREADS* and *NUM_HIT_PAIR_SUBREADS*. Increasing *MISMATCHES* and decreasing *NUM_HIT_SUBREADS* and *NUM_HIT_PAIR_SUBREADS*, the read and base level show a recall improvement of 20-30%. With regards to *NUM_EXTRACTED_SUBREADS*, a value different from the default seems to be helpful in many cases as well as the use of `--complexIndels`. The parameters *INDEL* and *MIN_FRAGMENT_LENGTH* do not affect the results.

C.14 TopHat2

Alignment command

```
tophat2 --output-dir <output path> --num-threads 16
--mate-inner-dist <INNER_MATE_MEAN>
--mate-std-dev <INNER_MATE_SD> --b2-very-sensitive
--read-mismatches <NUM_MISMATCHES>
--read-gap-length <NUM_GAP_LENGTH>
--read-edit-dist <NUM_EDIT_DIST>
--read-realign-edit-dist <NUM_REALIGN_EDIT_DIST>
--max-insertion-length <NUM_INSERTION_LENGTH>
--max-deletion-length <NUM_DELETION_LENGTH>
--max-multihits <NUM_MULTIHITS>
--GTF <gtf file> <index> <reads file 1> <reads file 2>
```

Tested parameters

For the tweaking, the parameters in Table C.40 were tested. About 230 different parameter combinations were considered during the tweaking process.

Table C.40 TopHat2 tweaking parameters and values

Parameters	Tested values	Note
`--b2-very-sensitive`	With and without this option	Default = without this option
`--coverage-search`	With and without this option	Default = without this option
`--read-mismatches <NUM_MISMATCHES>`	3; 4; 5; 6; 7; 8; 9; 10; 11; 12; 13; 14; 15; 16; 17; 18; 19; 20; 21; 22; 23; 24; 25; 26; 27; 28; 29; 30; 31; 32; 33; 34; 35	Default = 2
`--read-gap-length <NUM_GAP_LENGTH>`	3; 4; 5; 6; 7; 8; 9; 10; 16; 25; 26; 35	Default = 2
`--read-edit-dist <NUM_EDIT_DIST>`	5; 7; 10; 16; 25; 26; 35	Default = 2
`--read-realign-edit-dist <NUM_REALIGN_EDIT_DIST>`	0; autoset-default	Default = value such that the tool will not try to realign reads already mapped in earlier steps.
`--max-insertion-length <NUM_INSERTION_LENGTH>`	4; 5; 9; 10; 16; 24	Default = 3
`--max-deletion-length <NUM_DELETION_LENGTH>`	4; 5; 9; 10; 16; 24	Default = 3
`--max-multihits <NUM_MULTIHITS>`	100	Default = 20

Best configurations

Table C.41 TopHat2 best configurations (highest recall) for the base, read and junction level on the human T3 dataset

Parameters	Base level	Read level	Junction level
`--b2-very-sensitive`	with this option	with this option	with this option
`--coverage-search`	without this option	without this option	without this option
`--read-mismatches` *<NUM_MISMATCHES>*	18	18	25
`--read-gap-length` *<NUM_GAP_LENGTH>*	25	25	25
`--read-edit-dist` *<NUM_EDIT_DIST>*	25	25	25
`--read-realign-edit-dist` *<NUM_REALIGN_EDIT_DIST>*	default	default	default
`--max-insertion-length` *<NUM_INSERTION_LENGTH>*	24	24	24
`--max-deletion-length` *<NUM_DELETION_LENGTH>*	24	24	24
`--max-multihits` *<NUM_MULTIHITS>*	100	100	100

Table C.42 TopHat2 best configurations (highest recall) for the base, read and junction level on the P. falciparum T3 dataset

Parameters	Base level	Read level	Junction level
`--b2-very-sensitive`	with this option	with this option	with this option
`--coverage-search`	without this option	without this option	without this option
`--read-mismatches` *<NUM_MISMATCHES>*	18	18	27
`--read-gap-length` *<NUM_GAP_LENGTH>*	25	25	35
`--read-edit-dist` *<NUM_EDIT_DIST>*	25	25	35
`--read-realign-edit-dist` *<NUM_REALIGN_EDIT_DIST>*	default	default	default
`--max-insertion-length` *<NUM_INSERTION_LENGTH>*	24	24	24
`--max-deletion-length` *<NUM_DELETION_LENGTH>*	24	24	24
`--max-multihits` *<NUM_MULTIHITS>*	100	100	100

Conclusions

TopHat2 shows both the worst results using the default settings and the highest improvement by the tweaking of the parameters. This behavior suggests that performing the tweaking of the parameters is very important and highly recommended for this tool. The junction level precision/recall achieved by tweaking the parameters is one of the best, compared to the other tools. The most important parameters is *NUM_MISMATCHES*. In the performed tests, the tuning of *NUM_MISMATCHES* results in a recall improvement of more than 70% at any level.

Appendix D

Alignment notes for splice unaware tools

D.1 Bowtie

Version

1.1.2

Index

```
bowtie-build -f <genome file> <index name>
```

Alignment

```
bowtie --best --sam -X 2000 -p 4 <index name>
-q -1 <read file 1> -2 <read file 2> > <output sam file>
```

D.2 Bowtie2

Version

2.2.9

Index

```
bowtie2-build -f <genome file> <index name>
```

Alignment

```
bowtie2 --very-sensitive -X 2000 --time -p 4 -x <index name>
-q -1 <read file 1> -2 <read file 2> -S <output sam file>
```

D.3 BWA

Version

0.7.15-r1140

Index

```
bwa index -p <index name> -a is <genome file>
```

Alignment

```
bwa aln -t 4 <index name> <read file 1> > <output sai file 1>
```

```
bwa aln -t 4 <index name> <read file 2> > <output sai file 2>
```

```
bwa sampe -n 2000 <index name>
<output sai file 1> <output sai file 2>
<read file 1> <read file 2> > <output sam file>
```

D.4 BWAMEM

Version

0.7.15-r1140

Index

```
bwa index -p <index name> -a is <genome file>
```

Alignment

```
bwa mem -t 4 <index name> <read file 1> <read file 2> > <output sam
file>
```

Appendix E

Publications and side projects

The full Ph.D. research activity has more widely dealt with the implementation of computational methods for the analysis of RNA-Seq data, including collateral studies not described in this thesis. Side projects not described in this thesis include: definition and implementation of an optimal analysis pipeline for Mycobacterium tuberculosis; development of software for RNA expression level quantification; development of methods for differential expression analysis on RNA-Seq time series data.

E.1 Journal papers

- G. Baruzzo, K. Hayer, E. Ji Kim, B. Di Camillo, G. Fitzgerald, G. Grant. *"Simulation-based comprehensive benchmarking of RNA-seq aligners"*. *Nature Methods*, 2016.

E.2 Abstracts and short papers

- G. Baruzzo, F. Finotello, E. Lavezzo, A. Serafini, R. Provvedi, S. Toppo, L. Barzon, R. Manganelli and B. Di Camillo *"Benchmarking RNA-Seq mapping strategies for paired-end reads "* at Network Tools and Applications in Biology (NETTAB) 2014
- G. Baruzzo, K. Hayer, E. J. Kim, B. Di Camillo, G. Grant *"Benchmark Analysis of RNA-Seq Aligners"* at Intelligent Systems for Molecular Biology (ISMB) 2016
- N. Lahens, E. Ricciotti, O. Smirnova, E. Toorens, E. Ji Kim, G. Baruzzo, K. Hayer, T. Ganguly, J. Schug, G. Grant *"A comparison of Illumina and Ion Torrent platforms in a study of differential gene expression"* at Intelligent Systems for Molecular Biology (ISMB) 2016

E.3 Commentary articles

- G. Baruzzo, K. Hayer, E. Ji Kim, B. Di Camillo, G. Fitzgerald, G. Grant. *"Comprehensive Benchmarking of RNA-Seq Aligners Indicates Large Variation in Performance"* in *"Principles of Systems Biology, No. 13". Cell Systems*, 4(1):3–6, 1 2017.

References

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

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

[3] Ugrappa Nagalakshmi, Karl Waern, and Michael Snyder. RNA-Seq: A Method for Comprehensive Transcriptome Analysis. *Curr. Protoc. Mol. Biol*, 891113(4), 2010.

[4] Marc Sultan, Marcel H. Schulz, Hugues Richard, Alon Magen, Andreas Klingenhoff, Matthias Scherf, Martin Seifert, Tatjana Borodina, Aleksey Soldatov, Dmitri Parkhomchuk, Dominic Schmidt, Sean O'Keeffe, Stefan Haas, Martin Vingron, Hans Lehrach, and Marie-Laure Yaspo. A Global View of Gene Activity and Alternative Splicing by Deep Sequencing of the Human Transcriptome. *Science*, 321(5891), 2008.

[5] Olena Morozova, Martin Hirst, and Marco A. Marra. Applications of New Sequencing Technologies for Transcriptome Analysis. *Annual Review of Genomics and Human Genetics*, 10(1):135–151, 9 2009.

[6] James H Bullard, Elizabeth Purdom, Kasper D Hansen, and Sandrine Dudoit. Evaluation of statistical methods for normalization and differential expression in mRNA-Seq experiments. *BMC bioinformatics*, 11(1):94, 2010.

[7] Qun Pan, Ofer Shai, Leo J Lee, Brendan J Frey, and Benjamin J Blencowe. Deep surveying of alternative splicing complexity in the human transcriptome by high-throughput sequencing. *Nature Genetics*, 40(12):1413–1415, 12 2008.

[8] Zhiqiang Hu, Hamish S. Scott, Guangrong Qin, Guangyong Zheng, Xixia Chu, Lu Xie, David L. Adelson, Bergithe E. Oftedal, Parvathy Venugopal, Milena Babic, Christopher N. Hahn, Bing Zhang, Xiaojing Wang, Nan Li, and Chaochun Wei. Revealing Missing Human Protein Isoforms Based on Ab Initio Prediction, RNA-seq and Proteomics. *Scientific Reports*, 5:10940, 7 2015.

[9] Zhipeng Qu and David L Adelson. Evolutionary conservation and functional roles of ncRNA. *Frontiers in genetics*, 3:205, 2012.

[10] Sarah Geisler and Jeff Coller. RNA in unexpected places: long non-coding RNA functions in diverse cellular contexts. *Nature Reviews Molecular Cell Biology*, 14(11):699–712, 10 2013.

[11] Thomas R Cech and Joan A Steitz. The noncoding RNA revolution-trashing old rules to forge new ones. *Cell*, 157(1):77–94, 3 2014.

[12] Kevin V. Morris and John S. Mattick. The rise of regulatory RNA. *Nature Reviews Genetics*, 15(6):423–437, 4 2014.

[13] James Dewey Watson and Francis Harry Compton Crick. Molecular structure of nucleic acids; a structure for deoxyribose nucleic acid. *Nature*, 171(4356):737–8, 4 1953.

[14] F. Sanger and A.R. Coulson. A rapid method for determining sequences in DNA by primed synthesis with DNA polymerase. *Journal of Molecular Biology*, 94(3):441–448, 1975.

[15] F Sanger, G M Air, B G Barrell, N L Brown, A R Coulson, C A Fiddes, C A Hutchison, P M Slocombe, and M Smith. Nucleotide sequence of bacteriophage phi X174 DNA. *Nature*, 265(5596):687–95, 2 1977.

[16] F Sanger, S Nicklen, and A R Coulson. DNA sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Sciences of the United States of America*, 74(12):5463–7, 12 1977.

[17] Martin Kircher and Janet Kelso. High-throughput DNA sequencing–concepts and limitations. *BioEssays : news and reviews in molecular, cellular and developmental biology*, 32(6):524–36, 6 2010.

[18] Daniel C Koboldt, Karyn Meltz Steinberg, David E Larson, Richard K Wilson, and Elaine R Mardis. The next-generation sequencing revolution and its impact on genomics. *Cell*, 155(1):27–38, 9 2013.

[19] The Cost of Sequencing a Human Genome - National Human Genome Research Institute (NHGRI), https://www.genome.gov/sequencingcosts/.

[20] Lin Liu, Yinhu Li, Siliang Li, Ni Hu, Yimin He, Ray Pong, Danni Lin, Lihua Lu, Maggie Law, Lin Liu, Yinhu Li, Siliang Li, Ni Hu, Yimin He, Ray Pong, Danni Lin, Lihua Lu, and Maggie Law. Comparison of Next-Generation Sequencing Systems. *Journal of Biomedicine and Biotechnology*, 2012:1–11, 2012.

[21] Jay Shendure and Hanlee Ji. Next-generation DNA sequencing. *Nature Biotechnology*, 26(10):1135–1145, 10 2008.

[22] Michael L. Metzker. Sequencing technologies — the next generation. *Nature Reviews Genetics*, 11(1):31–46, 1 2010.

[23] Sara Goodwin, John D. McPherson, and W. Richard McCombie. Coming of age: ten years of next-generation sequencing technologies. *Nature Reviews Genetics*, 17(6):333–351, 5 2016.

[24] Devin Dressman, Hai Yan, Giovanni Traverso, Kenneth W Kinzler, and Bert Vogelstein. Transforming single DNA molecules into fluorescent magnetic particles for detection and enumeration of genetic variations. *Proceedings of the National Academy of Sciences of the United States of America*, 100(15):8817–22, 7 2003.

[25] Milan Fedurco, Anthony Romieu, Scott Williams, Isabelle Lawrence, and Gerardo Turcatti. BTA, a novel reagent for DNA attachment on glass and efficient generation of solid-phase amplified DNA colonies. *Nucleic acids research*, 34(3):e22, 2 2006.

[26] Anton Valouev, Jeffrey Ichikawa, Thaisan Tonthat, Jeremy Stuart, Swati Ranade, Heather Peckham, Kathy Zeng, Joel A Malek, Gina Costa, Kevin McKernan, Arend Sidow, Andrew Fire, and Steven M Johnson. A high-resolution, nucleosome position map of C. elegans reveals a lack of universal sequence-dictated positioning. *Genome research*, 18(7):1051–63, 7 2008.

[27] Jingyue Ju, Dae Hyun Kim, Lanrong Bi, Qinglin Meng, Xiaopeng Bai, Zengmin Li, Xiaoxu Li, Mong Sano Marma, Shundi Shi, Jian Wu, John R Edwards, Aireen Romu, and Nicholas J Turro. Four-color DNA sequencing by synthesis using cleavable fluorescent nucleotide reversible terminators. *Proceedings of the National Academy of Sciences of the United States of America*, 103(52):19635–40, 12 2006.

[28] Jia Guo, Ning Xu, Zengmin Li, Shenglong Zhang, Jian Wu, Dae Hyun Kim, Mong Sano Marma, Qinglin Meng, Huanyan Cao, Xiaoxu Li, Shundi Shi, Lin Yu, Sergey Kalachikov, James J Russo, Nicholas J Turro, and Jingyue Ju. Four-color DNA sequencing with 3'-O-modified nucleotide reversible terminators and chemically cleavable fluorescent dideoxynucleotides. *Proceedings of the National Academy of Sciences of the United States of America*, 105(27):9145–50, 7 2008.

[29] Jonathan M. Rothberg, Wolfgang Hinz, and Todd M. Rearick et al. An integrated semiconductor device enabling non-optical genome sequencing. *Nature*, 475(7356):348–352, 7 2011.

[30] Marcel Margulies, Michael Egholm, and William E. Altman et al. Genome sequencing in microfabricated high-density picolitre reactors. *Nature*, 437(7057):376, 7 2005.

[31] John Eid, Adrian Fehr, and Jeremy Gray et al. Real-Time DNA Sequencing from Single Polymerase Molecules. *Science*, 323(5910), 2009.

[32] M. J. Levene, J. Korlach, S. W. Turner, M. Foquet, H. G. Craighead, and W. W. Webb. Zero-Mode Waveguides for Single-Molecule Analysis at High Concentrations. *Science*, 299(5607), 2003.

[33] James Clarke, Hai-Chen Wu, Lakmal Jayasinghe, Alpesh Patel, Stuart Reid, and Hagan Bayley. Continuous base identification for single-molecule nanopore DNA sequencing. *Nature Nanotechnology*, 4(4):265–270, 4 2009.

[34] Travic C. Glenn. Field guide to next-generation DNA sequencers. *Molecular Ecology Resources*, 11(5):759–769, 9 2011.

[35] Nora Rieber, Marc Zapatka, Bärbel Lasitschka, David Jones, Paul Northcott, Barbara Hutter, Natalie Jäger, Marcel Kool, Michael Taylor, Peter Lichter, Stefan Pfister, Stephan Wolf, Benedikt Brors, and Roland Eils. Coverage Bias and Sensitivity of Variant Calling for Four Whole-genome Sequencing Technologies. *PLoS ONE*, 8(6):e66621, 6 2013.

[36] Olivier Harismendy, Pauline C Ng, Robert L Strausberg, Xiaoyun Wang, Timothy B Stockwell, Karen Y Beeson, Nicholas J Schork, Sarah S Murray, Eric J Topol, Samuel Levy, and Kelly A Frazer. Evaluation of next generation sequencing platforms for population targeted sequencing studies. *Genome biology*, 10(3):R32, 2009.

[37] Juliane C Dohm, Claudio Lottaz, Tatiana Borodina, and Heinz Himmelbauer. Substantial biases in ultra-short read data sets from high-throughput DNA sequencing. *Nucleic acids research*, 36(16):e105, 9 2008.

[38] Kensuke Nakamura, Taku Oshima, Takuya Morimoto, Shun Ikeda, Hirofumi Yoshikawa, Yuh Shiwa, Shu Ishikawa, Margaret C Linak, Aki Hirai, Hiroki Takahashi, Md Altaf-Ul-Amin, Naotake Ogasawara, and Shigehiko Kanaya. Sequence-specific error profile of Illumina sequencers. *Nucleic acids research*, 39(13):e90, 7 2011.

[39] André E Minoche, Juliane C Dohm, and Heinz Himmelbauer. Evaluation of genomic high-throughput sequencing data generated on Illumina HiSeq and genome analyzer systems. *Genome biology*, 12(11):R112, 11 2011.

[40] Nicholas J Loman, Raju V Misra, Timothy J Dallman, Chrystala Constantinidou, Saheer E Gharbia, John Wain, and Mark J Pallen. Performance comparison of benchtop high-throughput sequencing platforms. *Nature biotechnology*, 30(5):434–9, 5 2012.

[41] Mauricio O Carneiro, Carsten Russ, Michael G Ross, Stacey B Gabriel, Chad Nusbaum, and Mark A DePristo. Pacific biosciences sequencing technology for genotyping and variation discovery in human data. *BMC Genomics*, 13(1):375, 2012.

[42] Peter A Larsen, Amy M Heilman, and Anne D Yoder. The utility of PacBio circular consensus sequencing for characterizing complex gene families in non-model organisms. *BMC genomics*, 15(1):720, 8 2014.

[43] Sara Goodwin, James Gurtowski, Scott Ethe-Sayers, Panchajanya Deshpande, Michael C Schatz, and W Richard McCombie. Oxford Nanopore sequencing, hybrid error correction, and de novo assembly of a eukaryotic genome. *Genome research*, 25(11):1750–6, 11 2015.

[44] Peng Cui, Qiang Lin, Feng Ding, Chengqi Xin, Wei Gong, Lingfang Zhang, Jianing Geng, Bing Zhang, Xiaomin Yu, Jin Yang, Songnian Hu, and Jun Yu. A comparison between ribo-minus RNA-sequencing and polyA-selected RNA-sequencing. *Genomics*, 96(5):259–265, 2010.

[45] K. D. Hansen, S. E. Brenner, and S. Dudoit. Biases in Illumina transcriptome sequencing caused by random hexamer priming. *Nucleic Acids Research*, 38(12):e131–e131, 7 2010.

[46] Yarden Katz, Eric T Wang, Edoardo M Airoldi, and Christopher B Burge. Analysis and design of RNA sequencing experiments for identifying isoform regulation. *Nature methods*, 7(12):1009–15, 12 2010.

[47] Manuel Garber, Manfred G Grabherr, Mitchell Guttman, and Cole Trapnell. Computational methods for transcriptome annotation and quantification using RNA-seq. *Nature Methods*, 8(6):469–477, 6 2011.

[48] Dmitri Parkhomchuk, Tatiana Borodina, Vyacheslav Amstislavskiy, Maria Banaru, Linda Hallen, Sylvia Krobitsch, Hans Lehrach, and Alexey Soldatov. Transcriptome analysis by strand-specific sequencing of complementary DNA. *Nucleic acids research*, 37(18):e123, 10 2009.

[49] Joshua Z Levin, Moran Yassour, Xian Adiconis, Chad Nusbaum, Dawn Anne Thompson, Nir Friedman, Andreas Gnirke, and Aviv Regev. Comprehensive comparative analysis of strand-specific RNA sequencing methods. *Nature Methods*, 7(9):709–715, 9 2010.

[50] Michael G Ross, Carsten Russ, Maura Costello, Andrew Hollinger, Niall J Lennon, Ryan Hegarty, Chad Nusbaum, and David B Jaffe. Characterizing and measuring bias in sequence data. *Genome biology*, 14(5):R51, 5 2013.

[51] FastQC, http://www.bioinformatics.babraham.ac.uk.

[52] FASTX-Toolkit, http://hannonlab.cshl.edu/.

[53] Ravi K. Patel and Mukesh Jain. NGS QC Toolkit: A Toolkit for Quality Control of Next Generation Sequencing Data. *PLoS ONE*, 7(2):e30619, 2 2012.

[54] Marcel Martin. Cutadapt removes adapter sequences from high-throughput sequencing reads. *EMBnet.journal*, 17(1):10–12, 2011.

[55] Anthony M Bolger, Marc Lohse, and Bjoern Usadel. Trimmomatic: a flexible trimmer for Illumina sequence data. *Bioinformatics (Oxford, England)*, 30(15):2114–20, 8 2014.

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

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

[58] Heng Li and Richard Durbin. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics (Oxford, England)*, 25(14):1754–60, 7 2009.

[59] Ben Langmead and Steven L Salzberg. Fast gapped-read alignment with Bowtie 2. *Nature methods*, 9(4):357–9, 3 2012.

[60] Yinlong Xie, Gengxiong Wu, Jingbo Tang, Ruibang Luo, Jordan Patterson, Shanlin Liu, Weihua Huang, Guangzhu He, Shengchang Gu, Shengkang Li, Xin Zhou, Tak-Wah Lam, Yingrui Li, Xun Xu, Gane Ka-Shu Wong, and Jun Wang. SOAPdenovo-Trans: de novo transcriptome assembly with short RNA-Seq reads. *Bioinformatics (Oxford, England)*, 30(12):1660–6, 6 2014.

[61] Gordon Robertson, Jacqueline Schein, Readman Chiu, Richard Corbett, Matthew Field, Shaun D Jackman, Karen Mungall, Sam Lee, Hisanaga Mark Okada, Jenny Q Qian, Malachi Griffith, Anthony Raymond, Nina Thiessen, Timothee Cezard, Yaron S Butterfield, Richard Newsome, Simon K Chan, Rong She, Richard Varhol, Baljit

Kamoh, Anna-Liisa Prabhu, Angela Tam, YongJun Zhao, Richard A Moore, and Martin Hirst et al. De novo assembly and analysis of RNA-Seq data. *Nature Methods*, 7(11):909–912, 11 2010.

[62] Brian J Haas, Alexie Papanicolaou, Moran Yassour, Manfred Grabherr, Philip D Blood, Joshua Bowden, Matthew Brian Couger, David Eccles, Bo Li, Matthias Lieber, Matthew D Macmanes, Michael Ott, Joshua Orvis, Nathalie Pochet, Francesco Strozzi, Nathan Weeks, Rick Westerman, Thomas William, Colin N Dewey, Robert Henschel, Richard D Leduc, Nir Friedman, and Aviv Regev. De novo transcript sequence reconstruction from RNA-seq using the Trinity platform for reference generation and analysis. *Nature protocols*, 8(8):1494–512, 8 2013.

[63] Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, Richard Durbin, and 1000 Genome Project Data Processing Subgroup. The Sequence Alignment/Map format and SAMtools. *Bioinformatics (Oxford, England)*, 25(16):2078–9, 8 2009.

[64] Picard, http://broadinstitute.github.io.

[65] Y. Liao, G. K. Smyth, and W. Shi. featureCounts: an efficient general purpose program for assigning sequence reads to genomic features. *Bioinformatics*, 30(7):923–930, 4 2014.

[66] S. Anders, P. T. Pyl, and W. Huber. HTSeq–a Python framework to work with high-throughput sequencing data. *Bioinformatics*, 31(2):166–169, 1 2015.

[67] Aaron R Quinlan and Ira M Hall. BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics (Oxford, England)*, 26(6):841–2, 3 2010.

[68] Shanrong Zhao and Baohong Zhang. A comprehensive evaluation of ensembl, RefSeq, and UCSC annotations in the context of RNA-seq read mapping and gene quantification. *BMC Genomics*, 16(1):97, 2015.

[69] Günter P. Wagner, Koryu Kin, and Vincent J. Lynch. Measurement of mRNA abundance using RNA-seq data: RPKM measure is inconsistent among samples. *Theory in Biosciences*, 131(4):281–285, 12 2012.

[70] Mark D Robinson and Alicia Oshlack. A scaling normalization method for differential expression analysis of RNA-seq data. *Genome biology*, 11(3):R25, 2010.

[71] Simon Anders and Wolfgang Huber. Differential expression analysis for sequence count data. *Genome Biology*, 11(10):R106, 2010.

[72] Cole Trapnell, Brian A Williams, Geo Pertea, Ali Mortazavi, Gordon Kwan, Marijke J van Baren, Steven L Salzberg, Barbara J Wold, and Lior Pachter. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nature Biotechnology*, 28(5):511–515, 5 2010.

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

[74] Adam Roberts and Lior Pachter. Streaming fragment assignment for real-time analysis of sequencing experiments. *Nature Methods*, 10(1):71–73, 11 2012.

[75] Mark D. Robinson, Davis J. McCarthy, and Gordon K. Smyth. edgeR: A Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics*, 26(1):139–140, 1 2009.

[76] Michael I Love, Wolfgang Huber, and Simon Anders. Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biology*, 15(12):550, 12 2014.

[77] Ning Leng, John A Dawson, James A Thomson, Victor Ruotti, Anna I Rissman, Bart M G Smits, Jill D Haag, Michael N Gould, Ron M Stewart, and Christina Kendziorski. EBSeq: an empirical Bayes hierarchical model for inference in RNA-seq experiments. *Bioinformatics (Oxford, England)*, 29(8):1035–43, 4 2013.

[78] Thomas J Hardcastle and Krystyna A Kelly. baySeq: Empirical Bayesian methods for identifying differential expression in sequence count data. *BMC Bioinformatics*, 11(1):422, 2010.

[79] Charity W Law, Yunshun Chen, Wei Shi, and Gordon K Smyth. voom: precision weights unlock linear model analysis tools for RNA-seq read counts. *Genome Biology*, 15(2):R29, 2014.

[80] Sonia Tarazona, Fernando García-Alcalde, Joaquín Dopazo, Alberto Ferrer, and Ana Conesa. Differential expression in RNA-seq: A matter of depth. *Genome Research*, 21(12):2213–2223, 12 2011.

[81] Jun Li and Robert Tibshirani. Finding consistent patterns: a nonparametric approach for identifying differential expression in RNA-Seq data. *Statistical methods in medical research*, 22(5):519–36, 10 2013.

[82] Charlotte Soneson and Mauro Delorenzi. A comparison of methods for differential expression analysis of RNA-seq data. *BMC bioinformatics*, 14(1):91, 2013.

[83] Franck Rapaport, Raya Khanin, Yupu Liang, Mono Pirun, Azra Krek, Paul Zumbo, Christopher E Mason, Nicholas D Socci, and Doron Betel. Comprehensive evaluation of differential gene expression analysis methods for RNA-seq data. *Genome Biol*, 14(9):R95, 2013.

[84] Fatemeh Seyednasrollah, Asta Laiho, and Laura L. Elo. Comparison of software packages for detecting differential expression in RNA-seq studies. *Briefings in Bioinformatics*, 16(1):59–70, 1 2013.

[85] W James Kent. BLAT–the BLAST-like alignment tool. *Genome research*, 12(4):656–64, 4 2002.

[86] Z Ning, A J Cox, and J C Mullikin. SSAHA: a fast search method for large DNA databases. *Genome research*, 11(10):1725–9, 10 2001.

[87] Anthony J. Cox. ELAND: Efficient large-scale alignment of nucleotide databases. San Diego, 2007. Illumina, (unpublished).

[88] Heng Li, Jue Ruan, and Richard Durbin. Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome research*, 18(11):1851–8, 11 2008.

[89] Ruiqiang Li, Yingrui Li, Karsten Kristiansen, and Jun Wang. SOAP: short oligonucleotide alignment program. *Bioinformatics (Oxford, England)*, 24(5):713–4, 3 2008.

[90] David Weese, Anne-Katrin Emde, Tobias Rausch, Andreas Döring, and Knut Reinert. RazerS–fast read mapping with sensitivity control. *Genome research*, 19(9):1646–54, 9 2009.

[91] Can Alkan, Jeffrey M Kidd, Tomas Marques-Bonet, Gozde Aksay, Francesca Antonacci, Fereydoun Hormozdiari, Jacob O Kitzman, Carl Baker, Maika Malig, Onur Mutlu, S Cenk Sahinalp, Richard A Gibbs, and Evan E Eichler. Personalized copy number and segmental duplication maps using next-generation sequencing. *Nature genetics*, 41(10):1061–7, 10 2009.

[92] Stephen M. Rumble, Phil Lacroute, Adrian V. Dalca, Marc Fiume, Arend Sidow, and Michael Brudno. SHRiMP: Accurate Mapping of Short Color-space Reads. *PLoS Computational Biology*, 5(5):e1000386, 5 2009.

[93] Nils Homer, Barry Merriman, and Stanley F. Nelson. BFAST: An Alignment Tool for Large Scale Genome Resequencing. *PLoS ONE*, 4(11):e7767, 11 2009.

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

[95] Fabio De Bona, Stephan Ossowski, Korbinian Schneeberger, and Gunnar Rätsch. Optimal spliced alignments of short sequence reads. *Bioinformatics (Oxford, England)*, 24(16):174–80, 8 2008.

[96] Cole Trapnell, Lior Pachter, and Steven L Salzberg. TopHat: discovering splice junctions with RNA-Seq. *Bioinformatics (Oxford, England)*, 25(9):1105–11, 5 2009.

[97] T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, 1981.

[98] Katharina E Hayer, Angel Pizarro, Nicholas F Lahens, John B Hogenesch, and Gregory R Grant. Benchmark analysis of algorithms for determining and quantifying full-length mRNA splice forms from RNA-seq data. *Bioinformatics (Oxford, England)*, 31(24):3938–45, 12 2015.

[99] Sophie Schbath, Véronique Martin, Matthias Zytnicki, Julien Fayolle, Valentin Loux, and Jean-François Gibrat. Mapping Reads on a Genomic Sequence: An Algorithmic Overview and a Practical Comparative Analysis. *Journal of Computational Biology*, 19(6):796–813, 6 2012.

[100] Peter Weiner and Peter. Linear pattern matching algorithms. In *14th Annual Symposium on Switching and Automata Theory (swat 1973)*, pages 1–11. IEEE, 10 1973.

[101] Udi Manber and Gene Myers. Suffix Arrays: A New Method for On-Line String Searches. *SIAM Journal on Computing*, 22(5):935–948, 10 1993.

[102] M Burrows and DJ Wheeler. A Block-sorting Lossless Data Compression Algorithm. *Technical Report 124, Digital Equipment Corporation*, 1994.

[103] Paolo Ferragina and Giovanni Manzini. Opportunistic data structures with applications. *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, page 670, 2000.

[104] CLC Genomics Workbench, https://www.qiagenbioinformatics.com/.

[105] Thomas Bonfert, Evelyn Kirner, Gergely Csaba, Ralf Zimmer, and Caroline C Friedel. ContextMap 2: fast and accurate context-based RNA-seq mapping. *BMC Bioinformatics*, 16(1):122, 12 2015.

[106] Nicolas Philippe, Mikaël Salson, Commes Thérèsem, and Eric Rivals. CRAC: an integrated approach to the analysis of RNA-seq reads. *Genome Biology*, 14(3):R30, 2013.

[107] Thomas D Wu and Serban Nacu. Fast and SNP-tolerant detection of complex variants and splicing in short reads. *Bioinformatics (Oxford, England)*, 26(7):873–81, 4 2010.

[108] Daehwan Kim, Ben Langmead, and Steven L Salzberg. HISAT: a fast spliced aligner with low memory requirements. *Nature Methods*, 12(4):357–360, 3 2015.

[109] Kai Wang, Darshan Singh, Zheng Zeng, Stephen J Coleman, Yan Huang, Gleb L Savich, Xiaping He, Piotr Mieczkowski, Sara A Grimm, Charles M Perou, James N MacLeod, Derek Y Chiang, Jan F Prins, and Jinze Liu. MapSplice: accurate mapping of RNA-seq reads for splice junction discovery. *Nucleic acids research*, 38(18):e178, 10 2010.

[110] Novoalign, http://www.novocraft.com/.

[111] Jie Wu, Olga Anczuków, Adrian R Krainer, Michael Q Zhang, and Chaolin Zhang. OLego: fast and sensitive mapping of spliced mRNA-Seq reads using small seeds. *Nucleic acids research*, 41(10):5149–63, 5 2013.

[112] Gregory R Grant, Michael H Farkas, Angel D Pizarro, Nicholas F Lahens, Jonathan Schug, Brian P Brunk, Christian J Stoeckert, John B Hogenesch, and Eric A Pierce. Comparative analysis of RNA-Seq alignment algorithms and the RNA-Seq unified mapper (RUM). *Bioinformatics (Oxford, England)*, 27(18):2518–28, 9 2011.

[113] Songbo Huang, Jinbo Zhang, Ruiqiang Li, Wenqian Zhang, Zengquan He, Tak-Wah Lam, Zhiyu Peng, and Siu-Ming Yiu. SOAPsplice: Genome-Wide ab initio Detection of Splice Junctions from RNA-Seq Data. *Frontiers in genetics*, 2:46, 2011.

[114] Yang Liao, Gordon K Smyth, and Wei Shi. The Subread aligner: fast, accurate and scalable read mapping by seed-and-vote. *Nucleic acids research*, 41(10):e108, 5 2013.

[115] CLC bio. CLC bio White paper - Read Mapping. 2012.

[116] K M Chao, W R Pearson, and W Miller. Aligning two sequences within a specified diagonal band. *Computer applications in the biosciences : CABIOS*, 8(5):481–7, 10 1992.

[117] Nicolas Philippe, Mikaël Salson, Thierry Lecroq, Martine Léonard, Thérèse Commes, and Eric Rivals. Querying large read collections in main memory: a versatile data structure. *BMC Bioinformatics*, 12(1):242, 2011.

[118] Nicolas Philippe, Anthony Boureux, Laurent Bréhélin, Jorma Tarhio, Thérèse Commes, and Eric Rivals. Using reads to annotate the genome: influence of length, background distribution, and sequence errors on prediction capacity. *Nucleic acids research*, 37(15):e104, 8 2009.

[119] Thomas D Wu and Colin K Watanabe. GMAP: a genomic mapping and alignment program for mRNA and EST sequences. *Bioinformatics (Oxford, England)*, 21(9):1859–75, 5 2005.

[120] Jouni Siren, Niko Valimaki, and Veli Makinen. Indexing Graphs for Path Queries with Applications in Genome Research. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 11(2):375–388, 3 2014.

[121] Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, 1970.

[122] Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.

[123] Thasso Griebel, Benedikt Zacher, Paolo Ribeca, Emanuele Raineri, Vincent Lacroix, Roderic Guigó, and Michael Sammeth. Modelling and simulating generic RNA-Seq experiments with the flux simulator. *Nucleic acids research*, 40(20):10073–83, 11 2012.

[124] Chikara Furusawa and Kunihiko Kaneko. Zipf's Law in Gene Expression. *Physical Review Letters*, 90(8):088102, 2 2003.

[125] Malcolm J. Gardner, Neil Hall, Eula Fung, Owen White, and Matthew Berriman et al. Genome sequence of the human malaria parasite Plasmodium falciparum. *Nature*, 419(6906):498–511, 10 2002.

[126] Robert Lindner and Caroline C. Friedel. A Comprehensive Evaluation of Alignment Algorithms in the Context of RNA-Seq. *PLoS ONE*, 7(12):e52403, 12 2012.

[127] Pär G Engström, Tamara Steijger, Botond Sipos, Gregory R Grant, André Kahles, The RGASP Consortium, Gunnar Rätsch, Nick Goldman, Tim J Hubbard, Jennifer Harrow, Roderic Guigó, and Paul Bertone. Systematic evaluation of spliced alignment programs for RNA-seq data. *Nature Methods*, 10(12):1185–1191, 11 2013.

[128] Giacomo Baruzzo, Katharina E Hayer, Eun Ji Kim, Barbara Di Camillo, Garret A FitzGerald, and Gregory R Grant. Simulation-based comprehensive benchmarking of RNA-seq aligners. *Nature Methods*, 2016.

[129] Nuala A O'Leary, Mathew W Wright, and J Rodney Brister et al. Reference sequence (RefSeq) database at NCBI: current status, taxonomic expansion, and functional annotation. *Nucleic acids research*, 44(D1):733–45, 1 2016.

[130] Cristina Aurrecoechea, John Brestelli, Brian P Brunk, Jennifer Dommer, Steve Fischer, Bindu Gajria, Xin Gao, Alan Gingle, Greg Grant, Omar S Harb, Mark Heiges, Frank Innamorato, John Iodice, Jessica C Kissinger, Eileen Kraemer, Wei Li, John A Miller, Vishal Nayak, Cary Pennington, Deborah F Pinney, David S Roos, Chris Ross, Christian J Stoeckert, Charles Treatman, and Haiming Wang. PlasmoDB: a functional genomic database for malaria parasites. *Nucleic acids research*, 37(Database issue):539–43, 1 2009.

[131] Sagar Chhangawala, Gabe Rudy, Christopher E Mason, and Jeffrey A Rosenfeld. The impact of read length on quantification of differentially expressed genes and splice junction detection. *Genome biology*, 16(1):131, 6 2015.

[132] James L Weber, Donna David, Jeremy Heil, Ying Fan, Chengfeng Zhao, and Gabor Marth. Human diallelic insertion/deletion polymorphisms. *American journal of human genetics*, 71(4):854–62, 10 2002.

[133] Ryan E Mills, Christopher T Luttig, Christine E Larkins, Adam Beauchamp, Circe Tsui, W Stephen Pittard, and Scott E Devine. An initial map of insertion and deletion (INDEL) variation in the human genome. *Genome research*, 16(9):1182–90, 9 2006.

[134] Haiwang Yang, Yan Zhong, Cheng Peng, Jian-Qun Chen, and Dacheng Tian. Important role of indels in somatic mutations of human cancer genes. *BMC medical genetics*, 11:128, 9 2010.

[135] Brunangelo Falini, Cristina Mecucci, Enrico Tiacci, Myriam Alcalay, Roberto Rosati, Laura Pasqualucci, Roberta La Starza, Daniela Diverio, Emanuela Colombo, Antonella Santucci, Barbara Bigerna, Roberta Pacini, Alessandra Pucciarini, Arcangelo Liso, Marco Vignetti, Paola Fazi, Natalia Meani, Valentina Pettirossi, Giuseppe Saglio, Franco Mandelli, Francesco Lo-Coco, Pier-Giuseppe Pelicci, and Massimo F. Martelli. Cytoplasmic Nucleophosmin in Acute Myelogenous Leukemia with a Normal Karyotype. *New England Journal of Medicine*, 352(3):254–266, 1 2005.

[136] ST Warren, F Zhang, GR Licameli, and JF Peters. The fragile X site in somatic cell hybrids: an approach for molecular cloning of fragile sites. *Science*, 237(4813), 1987.

[137] Ryan E Mills, W Stephen Pittard, Julienne M Mullaney, Umar Farooq, Todd H Creasy, Anup A Mahurkar, David M Kemeza, Daniel S Strassler, Chris P Ponting, Caleb Webber, and Scott E Devine. Natural genetic variation caused by small insertions and deletions in the human genome. *Genome research*, 21(6):830–9, 6 2011.

[138] Chee Seng Ku, En Yun Loy, Agus Salim, Yudi Pawitan, and Kee Seng Chia. The discovery of human genetic variations and their use as disease markers: past, present and future. *Journal of Human Genetics*, 55(7):403–415, 7 2010.

[139] Yue Jiang, Andrei L Turinsky, and Michael Brudno. The missing indels: an estimate of indel variation in a human genome and analysis of factors that impede detection. *Nucleic acids research*, 43(15):7217–28, 9 2015.

[140] Julienne M Mullaney, Ryan E Mills, W Stephen Pittard, and Scott E Devine. Small insertions and deletions (INDELs) in human genomes. *Human molecular genetics*, 19(R2):131–6, 10 2010.

[141] Xiaoqing Yu, Kishore Guda, Joseph Willis, Martina Veigl, Zhenghe Wang, Sanford Markowitz, Mark D Adams, and Shuying Sun. How do alignment programs perform on sequencing data with varying qualities and from repetitive regions? *BioData Mining*, 5(1):6, 12 2012.

[142] Ivan Borozan, Stuart N. Watt, and Vincent Ferretti. Evaluation of Alignment Algorithms for Discovery and Identification of Pathogens Using RNA-Seq. *PLoS ONE*, 8(10):e76935, 10 2013.

[143] Jing Shang, Fei Zhu, Wanwipa Vongsangnak, Yifei Tang, Wenyu Zhang, Bairong Shen, Jing Shang, Fei Zhu, Wanwipa Vongsangnak, Yifei Tang, Wenyu Zhang, and Bairong Shen. Evaluation and comparison of multiple aligners for next-generation sequencing data analysis. *BioMed research international*, 2014:309650, 2014.

[144] Ayat Hatem, Doruk Bozdağ, Amanda E Toland, and Ümit V Çatalyürek. Benchmarking short sequence mapping tools. *BMC Bioinformatics*, 14(1):184, 2013.

[145] Matthew Ruffalo, Thomas LaFramboise, and Mehmet Koyutürk. Comparative analysis of algorithms for next-generation sequencing read alignment. *Bioinformatics (Oxford, England)*, 27(20):2790–6, 10 2011.

[146] H. Li and N. Homer. A survey of sequence alignment algorithms for next-generation sequencing. *Briefings in Bioinformatics*, 11(5):473–483, 9 2010.

[147] C A Heid, J Stevens, K J Livak, and P M Williams. Real time quantitative PCR. *Genome research*, 6(10):986–94, 10 1996.

[148] Marisa L Wong and Juan F Medrano. Real-time PCR for mRNA quantitation. *BioTechniques*, 39(1):75–85, 7 2005.

[149] Tania Nolan, Rebecca E Hands, and Stephen A Bustin. Quantification of mRNA using real-time RT-PCR. *Nature Protocols*, 1(3):1559–1582, 11 2006.

[150] Stefan Cikos, Alexandra Bukovská, and Juraj Koppel. Relative quantification of mRNA: comparison of methods currently used for real-time PCR data analysis. *BMC molecular biology*, 8:113, 12 2007.

[151] Lichun Jiang, Felix Schlesinger, Carrie A Davis, Yu Zhang, Renhua Li, Marc Salit, Thomas R Gingeras, and Brian Oliver. Synthetic spike-in standards for RNA-seq experiments. *Genome research*, 21(9):1543–51, 9 2011.

[152] Thermo Fisher Scientific.

[153] Nuno A. Fonseca, John Marioni, and Alvis Brazma. RNA-Seq Gene Profiling - A Systematic Empirical Comparison. *PLoS ONE*, 9(9):e107026, 9 2014.

[154] Alexander Kanitz, Foivos Gypas, Andreas J Gruber, Andreas R Gruber, Georges Martin, and Mihaela Zavolan. Comparative assessment of methods for the computational inference of transcript isoform abundance from RNA-seq data. *Genome biology*, 16(1):150, 2015.

[155] Praveen Kumar Raj Kumar, Thanh V. Hoang, Michael L. Robinson, Panagiotis A. Tsonis, and Chun Liang. CADBURE: A generic tool to evaluate the performance of spliced aligners on RNA-Seq data. *Scientific Reports*, 5:13443, 8 2015.

[156] Merly Escalona, Sara Rocha, and David Posada. A comparison of tools for the simulation of genomic next-generation sequencing data. *Nature Reviews Genetics*, 17(8):459–469, 6 2016.

[157] Nicolas L Bray, Harold Pimentel, Páll Melsted, and Lior Pachter. Near-optimal probabilistic RNA-seq quantification. *Nature Biotechnology*, 34(5):525–527, 4 2016.

[158] Rob Patro, Geet Duggal, Michael I Love, Rafael A Irizarry, and Carl Kingsford. Salmon provides accurate, fast, and bias-aware transcript expression estimates using dual-phase inference. *bioRxiv*, 2016.

[159] Rob Patro, Stephen M Mount, and Carl Kingsford. Sailfish enables alignment-free isoform quantification from RNA-seq reads using lightweight algorithms. *Nature Biotechnology*, 32(5):462–464, 4 2014.