CZECH TECHNICAL UNIVERSITY IN PRAGUE

FACULTY OF BIOMEDICAL ENGINEERING

Department of Biomedical Technology

A tool for RNAseq data processing in patients with leukemia

Master thesis

Study programme:	Biomedical and Clinical Informatics
Specialization:	Software Technologies
Supervisor of the master thesis:	Ing. Ondřej Klempíř, Ph.D.
Consultant of the master thesis:	Mgr. Pavla Suchánková

Bc. Michaela Součková

Kladno 2023

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení:	**Součková**	Jméno:	**Michaela**	Osobní číslo:	**483028**
Fakulta:	**Fakulta biomedicínského inženýrství**				
Garantující katedra:	**Katedra biomedicínské informatiky**				
Studijní program:	**Biomedicínská a klinická informatika**				

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Vytvoření nástroje pro zpracování dat z RNAseq analýz pacientů s leukemickým onemocněním

Název diplomové práce anglicky:

A tool for RNAseq data processing in patients with leukemia

Pokyny pro vypracování:

Transcriptome sequencing (RNAseq) is a modern method to study the cell transcriptome, which can be used to identify known and previously undescribed fusion genes, determine the expression of individual genes and their variants, and the presence of expressed mutations and small insertions or deletions. Detection of genetic aberrations and fusion transcripts in leukemia patients can serve as a tool for the correct classification of the disease and the selection of appropriate treatment. RNAseq is a source of large amounts of data that must be processed quickly, efficiently, and accurately. Therefore, it is necessary to develop functional processing of these data. The analysis of these data is not a trivial task and requires a sequence of steps that have different inputs and outputs that may or may not be communicated by each step. For effective use of computational and human resources, it is appropriate and necessary to automate individual analysis steps. The sequence of steps forms a bioinformatics pipeline, and the steps can be sequenced using a Bash script. This brings with it a large number of problems, e.g. simple non-portability, difficulty to add additional steps, difficulty to optimise resources and performance; also documentation of pipelines written in this way is problematic and demanding and requires a high degree of technical knowledge not only of Linux systems. Current trends in bioinformatic pipeline creation use NextFlow technology (Di Tommaso et al., 2017). The aim of this work is to create a bioinformatics pipeline that will process data from RNAseq analyses of leukemia patients. The input to this pipeline will be the raw sequencer data and the output will be the analysis of fusion genes in individual patients preprepared for expert analysis. The sub-objectives of this work are: Describe the capabilities of the NextFlow tool for the execution of bioinformatic pipelines, including deployment. Create a functional bioinformatic pipeline for processing fusion gene analysis of patient samples. Creation of a summary output adapted for expert analysis. Creation of documentation and user manuals for the installation and use of the pipeline.

Seznam doporučené literatury:

[1] Di Tommaso et al., Nextflow enables reproducible computational workflows, Nature Biotechnology, ročník 35, číslo 4, 2017, doi:10.1038/nbt.3820
[2] ANTAO Tiago, Bioinformatics with Python Cookbook, ed. ed. Second, Packt Publishing, 2018, ISBN 9781789344691
[3] PŘISTOUPILOVÁ Anna, Využití nových metod analýzy genomu ve studiu molekulární podstaty vzácných geneticky podmíněných onemocnění, 2020

Jméno a příjmení vedoucí(ho) diplomové práce:

Ing. Ondřej Klempíř, Ph.D.

Jméno a příjmení konzultanta(ky) diplomové práce:

Mgr. Pavla Suchánková

Datum zadání diplomové práce: **14.02.2023**
Platnost zadání diplomové práce: **20.09.2024**

doc. Ing. Zoltán Szabó Ph.D.
vedoucí katedry

prof. MUDr. Jozef Rosina, Ph.D., MBA
děkan

DECLARATION

I hereby declare that I have completed this thesis with the topic “A tool for RNAseq data processing in patients with leukemia” independently, and that I have attached an exhaustive list of citations of the employed sources.

I do not have a compelling reason against the use of the thesis within the meaning of Section 60 of the Act No. 121/2000 Sb., on copyright, rights related to copyright and amending some laws (Copyright Act).

In Kladno 18.5.2023 Bc. Michaela Součková

ACKNOWLEDGEMENTS

This endeavour would not have been possible without my supervisor, Ing. Ondřej Klempíř, Ph.D., and my consultant, Mgr. Pavla Suchánková, to whom I am grateful for their patience and feedback.

Special thanks to The Institute of Hematology and Blood Transfusion for an invaluable internship opportunity.

Computational resources were provided by the e-INFRA CZ project (ID:90254), supported by the Ministry of Education, Youth and Sports of the Czech Republic. Thanks should also go to their inestimable user support.

ABSTRACT

A tool for RNAseq data processing in patients with leukemia:

The aim is to develop a bioinformatic Nextflow pipeline that would analyse RNAseq data of leukemic patients with the emphasis on fusion gene detection. Since gene fusions are believed to be associated with tumour phenotype, they have been of significant importance for clinical purposes, as well as for understanding tumorigenesis. With mapping current trends in RNAseq data processing and fusion detection, we provide a modular workflow consisting of processes that leverage suitable bioinformatic tools and manage fusion gene detection along with pre-processing and validation. The detected fusion candidates are preprepared as a formatted summary table for subsequent expert analysis.

Keywords

RNAseq, fusion gene, Nextflow

Table of Contents

List of symbols and abbreviations

List of abbreviations

Abbreviation	Meaning
bp	Base pairs
cDNA	Complementary DNA
CML	Chronic myeloid leukemia
CPU	Central Processing Unit
DNA	Deoxy nucleic acid
FAR	Fusion allelic ratio
FFPM	Fusion fragment per million
HTML	Hypertext Markup Language
ITD	Internal tandem duplication
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
MR	Molecular response
mRNA	Messenger ribonucleic acid
NGS	Next-Generation Sequencing
OS	Operating system
PCR	Polymerase chain reaction
RAM	Random access memory
RNA	Ribonucleic acid
RNAseq	RNA Sequencing

1 Introduction

Transcriptome sequencing (RNAseq) is a method that rapidly emerged with the advent of Next-Generation Sequencing (NGS). By the virtue of these quickly developing technologies, we could witness an extensive research on human genomic aberrations that are believed to be cause factors of variety of illnesses. Among these variations, gene fusions have been of great interest due to their associations with cancer [1].

Gene fusions are hybrid genes formed from two previously separate genes and can occur as a result of a translocation, deletion, or chromosomal inversion [2]. Their analysis from RNAseq data is now almost a routine task in cancer research and oncological clinical practice, since they are ideal for diagnostic purposes, enable the subclassification of disease entities and affect sensitivity to relevant drugs [1]. A canonical example is BCR::ABL1 that is believed to be necessary for chronic myeloid leukemia initiation and maintenance and is found in ~95% of patients [3].

The analysis of RNAseq data, such as gene fusion detection, is not straightforward, as it requires a series of distinct steps that are often computationally demanding [4] and each has its own set of inputs and outputs, which may or may not be shared between them. These sequential steps collectively form a bioinformatics pipeline, which can be orchestrated using a Bash script. However, this approach comes with various downsides, including issues with portability, difficulty in incorporating additional steps, and optimizing resources and performance. Furthermore, documenting pipelines written in this manner is demanding and problematic, requiring extensive technical knowledge. In response to these challenges, contemporary trends in bioinformatics pipeline development use Nextflow [5], an open-source framework for bioinformatics workflow management, which utilises parallelisation and distributed computing.

The aim of this thesis is to create a fusion gene detection pipeline leveraging the abilities of Nextflow along with Docker technologies that will process data from RNA sequencing of leukemia patients and provide a summary of fusion gene candidates preprepared for expert analysis.

2 Biological background and clinical motivation

2.1 Fusion Mechanism

2.1.1 DNA and RNA

Genetic instructions, which are required for the development, function, and reproduction of all known living organisms, are carried in nucleic acids - deoxy nucleic acid (DNA) and ribonucleic acid (RNA) [6]. "*These macromolecules contain the information for determining the amino acid sequence, and hence the structure and function of all the proteins of a cell, are part of the cellular structures that select and align amino acids in the correct order as a polypeptide chain is being synthesised and catalyse a number of fundamental chemical reactions in cells, including formation of peptide bonds between amino acids during protein synthesis.*" [7]

The nucleic acids are chemically similar. The linear structures of both are linear polymers composed of monomers called **nucleotides,** all of which have a common structure. A phosphate group is linked by a phosphoester bond to pentose (ribose for RNA and deoxyribose for DNA) that in turn is linked to a nitrogen- and carbon-containing ring structure commonly referred to as a **base**. [2] There are five bases: purines, adenine and guanine, and pyrimidines, cytosine, thymine, and uracil. The bases are often referred to by their first letter, A, G, C, T, and U. Sometimes, these single-letter abbreviations are also commonly used to denote the entire nucleotides. In DNA we can find only A, G, C and T, and in RNA thymine is replaced by uracil [6].

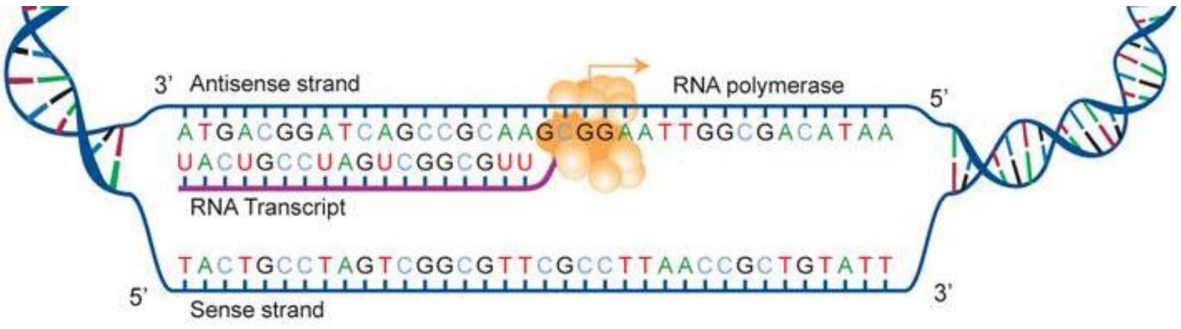


Figure 2.1: DNA and RNA [8]

As seen in Figure 2.1, DNA is made of two strands of polynucleotide chains coiled around each other in a double-helix structure and can be as long as several hundred million nucleotides [7]. The sugar and phosphate groups lie outside the molecule, and the purines and pyrimidines lie inside the molecule. A always pairs with T, and G always pairs with C, creating the so-called **base pairs** (bp) [9]. On the other hand, RNA is a single-stranded molecule of less than 100 to many thousands of nucleotides [7].

2.1.2 End-to-End directionality

A nucleic acid strand has an end-to-end chemical orientation: the 5' end has a hydroxyl or phosphate group on the 5' carbon of its terminal sugar; the 3' end usually has a hydroxyl group on the 3' carbon of its terminal sugar. [7] The two chains of DNA are said to be antiparallel because they lie in the opposite orientation with respect to one another, with the 3'-hydroxyl terminus of one strand opposite the 5'-phosphate terminus of the second strand. [9]

Few conventional terms are used to refer to each strand and its directionality.

- DNA is double-stranded. By convention, for a reference chromosome, one whole strand is called the **forward strand (+)** and the other the **reverse strand (-)**.
- Sequences are conventionally written and read in the 5'->3' direction, since the synthesis proceeds in that direction.
- The mRNA sequence of a gene corresponds to the DNA sequence as read from the gene's coding strand. Therefore, the mRNA sequence always corresponds to the 5-3 coding sequence of a gene.
- If a sequence is considered in **reverse**, then it is formed by reversing the order of the letters. If a sequence is considered as a **complement**, then it is formed by swapping each base for the other one in its base pair. [9]

2.1.3 Genome and genes

A **genome** is all of an organism’s DNA sequence [6]. Humans have 46 DNA molecules in each somatic cell, each of which forms one chromosome. We inherit 23 chromosomes from each parent, where each set of 23 chromosomes encodes a complete copy of our genome and is made up of 6×10^9 nucleotides [9].

The genome of cellular organisms has regions that contain the instructions for making proteins, so-called **coding regions**, but it may also have regions used to produce molecules other than proteins or regions that regulate the rates by which other processes take place.

A **gene** is considered the fundamental unit of inheritance. It is a segment of DNA that specifies the structure of proteins that are responsible for the phenotype (observable traits of an organism) associated with a particular gene. The estimated number of genes in human DNA ranges from 30 000 to 120 000 genes. [10]

All genes are in almost every cell, however, only some of the genes are used in any particular cell at any given time. When and in what tissue a gene is expressed is controlled by a specific region of a gene, called promoter. The structure of a gene's protein is specified by the gene's coding region. [10]

These coding regions of most genes are not continuous, as seen in Figure 2.2. The areas that are transcribed into mRNA are called **exons**. However, exons can be interrupted by areas that do not appear in mature mRNA, called **introns**. The purpose of these parts is not yet fully known. [10]

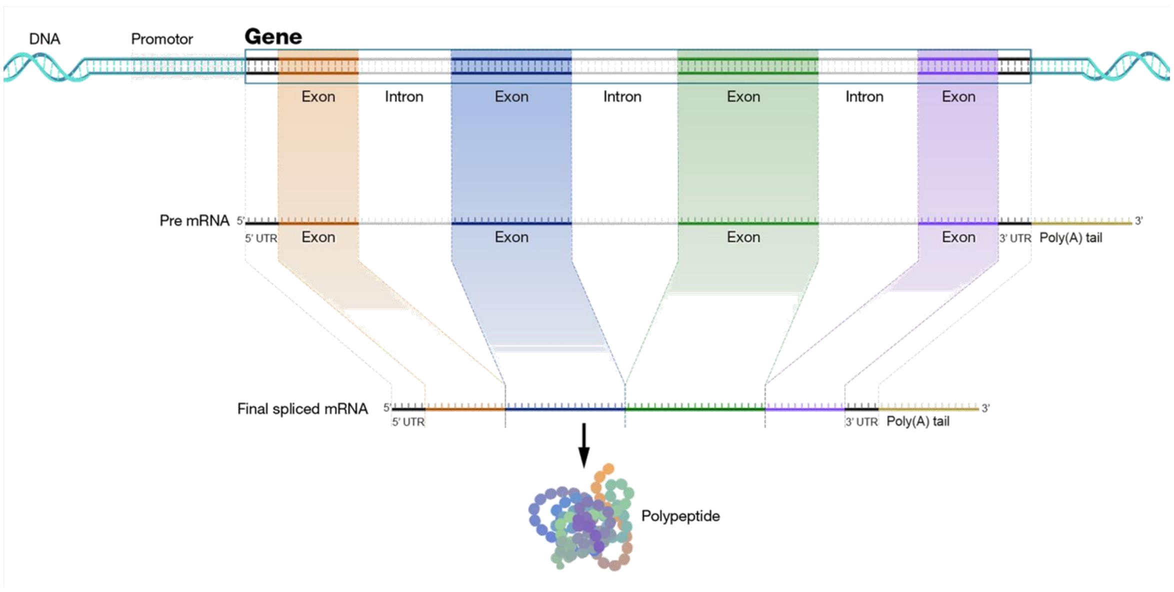


Figure 2.2: The process of transcription from DNA to mRNA [8]

2.1.4 Gene expression and transcriptome

The process of going from DNA to a functional product (e.g., protein) is known as **gene expression**. The 'central dogma' of molecular biology states that DNA-encoded genetic information is transcribed to mRNA and then translated to protein [11]. The transcription process involves creating an RNA copy of the gene using the DNA of the gene as a template. The translation of mRNA into protein is a process of decoding the structural message in mRNA and synthesis of a given protein. [10]

The identity of each gene expressed in a particular cell at a given time and its level of expression is defined as the **transcriptome** [10]. The transcriptome is the complete set of coding and noncoding RNAs that are transcribed at a specific developmental stage and/or are present under various physiological conditions within a cell type or tissue [12].

2.1.5 Gene fusions

Gene fusion refers to the formation of a new chimeric transcript or gene structure as a result of joining complete or partial sequences of two or more exons of different genes that have the potential to encode new proteins with new functions [4]. Gene fusions are formed by chromosomal rearrangement, including translocation, inversion, deletion, or tandem duplication [2]. These formations can be observed in all domains of life and are a constant source of new genes, which can be beneficial, but may also lead to abnormal cell proliferation and cancer [4].

Taking into account their prevalence and common characteristics in diverse types of human cancer, gene fusions are always regarded a distinct class of mutations [13], and have been used successfully as diagnostic tools [2].

For the terminology clarification, **gene fusion** refers to DNA-level fusion events, and **chimeric RNA** refers to any transcript composed of exons from different parental genes, including gene fusion transcripts. Chimeric RNA can be a product of gene fusion, but can also be generated by trans-splicing of two separate precursor mRNAs and alternative splicing of a readthrough transcript (in these cases it cannot be detected by DNA-based assays, as they are produced in the absence of chromosomal rearrangement, and therefore RNA-based analysis should be used). [2]

Both gene fusions and chimeric RNAs have strong associations with cancer and have major impacts on cancer diagnosis and treatment; however, detection of either is not necessarily indicative of cancer. [2] An example of a gene fusion is BCR::ABL1, which is significant for chronic myeloid leukemia, as described below.

As shown in the Figure 2.3.a, "*gene fusions may originate through balanced and unbalanced chromosome rearrangements. Balanced changes comprise translocations (the transfer of chromosome segments between chromosomes), insertions (a chromosome segment in a new interstitial position in the same or another chromosome) and inversions (a rotation of a chromosome segment by 180 degrees).*" [1] Both balanced and unbalanced aberrations may lead to the deregulation of either gene A or gene B in one of the breakpoints by the juxtaposition of the coding sequences with the regulatory sequences of the gene in the other breakpoint, or by the creation of a chimeric gene through the fusion of parts of the two genes, one in each breakpoint [1], as seen in the Figure 2.3.b.

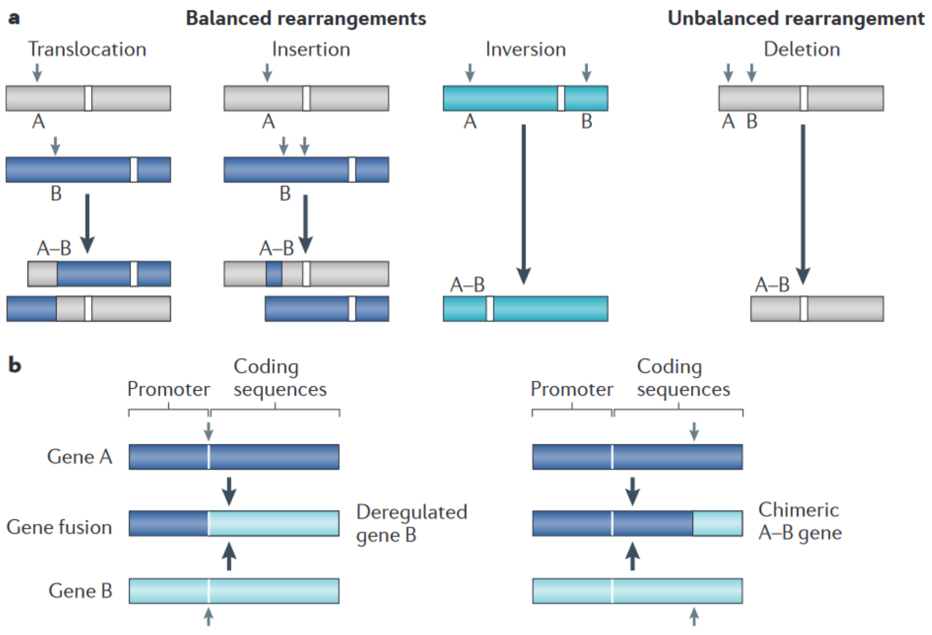


Figure 2.3: The chromosomal basis of gene fusions. a: Balanced and unbalanced rearrangements. Small grey arrows indicate breakpoints, and the large arrows indicate the resulting rearranged chromosomes. A and B signify affected genes. **b**: Possible outcomes of rearrangements. Small grey arrows indicate breakpoints. [1]

2.2 Clinical practice

Gene fusions are a prototypical example of a pathognomonic mutation, in a sense they are characteristic for a particular disease, and the detection and characterisation of gene fusions have been of great importance for clinical purposes, as well as for understanding tumorigenesis [1]. Thousands of fusion genes have been identified in cancer patients, but the functional consequences and therapeutic implications of most of these remain largely unknown. [14, 15]

Gene fusions have now been identified in several common carcinomas, including those of the prostate, lung, breast, head and neck, brain, skin, gastrointestinal tract, and kidney, which, along with widely documented gene fusions in thyroid and salivary gland tumours, support the notion that gene fusions are integral to the genomic landscape of most cancers. [16]

The close association between the type of gene fusion and the tumour phenotype makes gene fusions ideal for diagnostic purposes, enabling the subclassification of otherwise seemingly identical disease entities [1]. Some fusions were identified to markedly affect sensitivity to relevant drugs [14]. In addition, many gene fusions add important information for risk stratification, and increasing numbers of chimeric proteins encoded by the gene fusions serves as specific targets for treatment, resulting in dramatically improved patient outcomes. [1, 17]

For example, gene fusions that involve oncogenes such as *ERG*, *ETV1*, *TFE3*, *NUT*, *POU5F1*, *NFIB*, *PLAG1* and *PAX8* are diagnostically useful. Tumours with fusions involving therapeutically targetable genes such as *ALK*, *RET*, *BRAF*, *RAF1*, *FGFR1–4*, and *NOTCH1–3* have immediate implications for precision medicine across tissue types [16].

Gene fusions are also frequently seen in leukemia and several of the recurrent gene fusions are required for subgrouping of leukemia and prognostication [18]. Characterisation of oncogenic fusion *BCR::ABL1* at t(9,22) translocation loci in chronic myeloid leukemia, the first gene fusion described in cancer, culminated in the development of a molecularly targeted therapy, provides a compelling paradigm of 'bench-to-bedside' for cancers [16, 19].

2.2.1 Chronic myeloid leukemia

Chronic myeloid leukemia (CML) is a myeloproliferative disorder. It is characterised by a biphasic or triphasic clinical course in which a terminal blastic phase follows a chronic phase of variable duration. [20] CML is a rare disease with an incidence of 1 or 2 cases per 100 000 people every year, and is most common in older people, with a median age at diagnosis of around 65 years, where men are affected more frequently than are women [21].

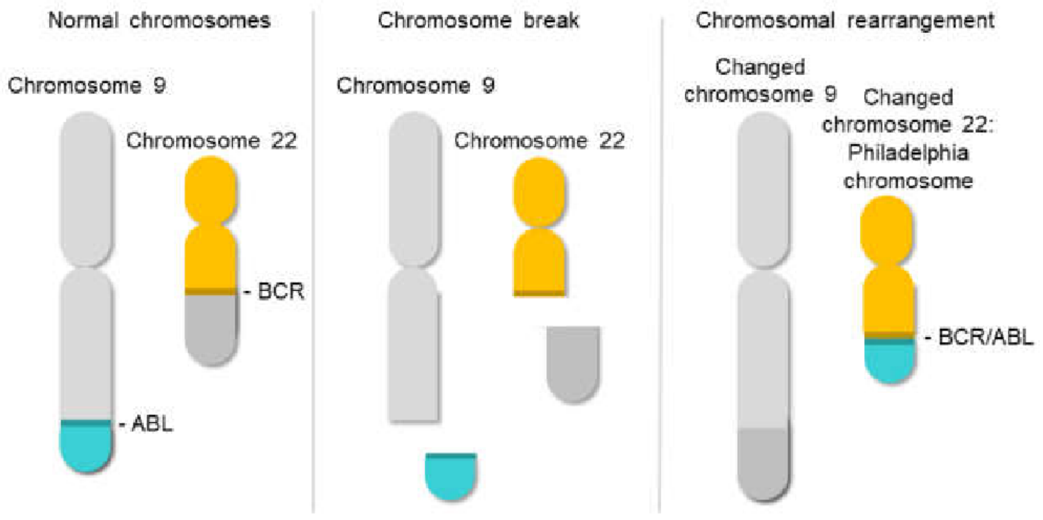


Figure 2.4: **Structure of the BCR::ABL1 oncogene.** Schematic representation of the t(9;22)(q34;q11) translocation triggering the Philadelphia chromosome. [22]

CML was the first neoplastic disease for which knowledge of the genotype led to rationally designed therapy [21]. On a molecular level, most patients demonstrate **BCR::ABL1** fusion genes, which are the result of a translocation between chromosomes 9 and 22, see Figure 2.4. This translocation leads to a shortened chromosome 22, called the **Philadelphia chromosome** [2, 18, 20, 21], and its chimeric transcript encodes a fusion protein that is an altered constitutively active ABL1 kinase [2]. Only in about 5% of cases the Philadelphia chromosome cannot be detected [3], and confirmation of diagnosis is dependent on finding the BCR::ABL1 transcript [21].

3 Overview of the current state of the art

3.1 Next-Generation Sequencing

3.1.1 Sequencing

Sequencing is a process to determine the order of bases in DNA/RNA [6]. In this thesis, the focus is mainly on **RNA sequencing** (RNAseq or transcriptome sequencing), a tool used to study the transcriptome, the total RNA molecules present in one or a collection of cells, providing the knowledge of gene regulation and protein content information. RNAseq provides insight into differential expression of genes, differently spliced transcripts, gene alleles, noncoding and small RNAs, alternative splicing, or also gene fusions. RNA sequencing has also been used to discover novel gene sequences in transcriptomes. [4, 23]

3.1.2 Generations

The term "generation" refers to the chemistry and technology used by the sequencing process. First-generation sequencing denotes Sanger sequencing. Frederic Sanger stood at the very beginning of DNA sequencing in 1972. He came with a method for "DNA sequencing with chain-terminating inhibitors", which became popular and led to many biological successes including the first sequence of human genome in 2001. However, the method did not observe rapid changes for the next three decades. [23]

Around 2005, **Next-Generation Sequencing (NGS)** was introduced [3] and is termed second generation sequencing. This method has been able to parallelise the sequencing reactions in a massive manner, and therefore generate a huge amount of data very rapidly at a modest cost. [23]

And that is the main difference between Sanger (traditional) sequencing and NGS. "*Sequencers based on Sanger sequencing produce a read length (the length of DNA fragment that can be sequenced at a time) of 800-1000 bp. Because only one read can be sequenced in one capillary of the sequencer at a time, the total output of the run is equal to the read length. However, sequencers with multiple capillaries allow us to sequence multiple samples at a time, for example 8, 16, 48, or 96. On the other hand, next-generation sequencers work on the principle of sequencing millions of DNA fragments simultaneously in a massively parallel mode and produce sequence data in megabases, gigabases, and now terrabases. The whole genome or transcriptome of an organism is fragmented into millions of small pieces and sequenced independently in parallel*." [23]

The third-generation sequencing (also next-NGS) refers to technologies which were developed to make sequencing cheaper than second-generation sequencing. This method interrogates molecules of DNA without amplifying them through PCR. [23, 24]

The invention of NGS enabled researchers and clinicians to study biological systems at a level and resolution never before possible. The enormous information produced by NGS helps to understand genomic variations, disease mechanisms, and resistance, helping to develop better diagnostics, therapies, and better breeds. [23]

3.1.3 Terminology

The terms important for the understanding of NGS are described below:

- **The template** is a DNA/RNA sequence part of which is sequenced on a sequencing machine or assembled from raw sequences [25].
- In NGS, a **library** is defined as a collection of RNA fragments that represents either the entire transcriptome or a target region. Each NGS platform has its specificities, but, in simple terms, the preparation of an NGS library starts with the fragmentation of the starting material; then sequence adaptors are connected to fragments to allow the enrichment of those fragments. [24]. In other words, library is a set of nucleic acid fragments which has undergone all processing steps and is ready for actual sequencing.
- **A read** is a raw sequence that comes from a sequencing machine. The read may consist of multiple segments. For sequencing data, reads are indexed by the order in which they are sequenced [25].
- **Single-end sequencing** refers to reading a fragment where the fragment is read from one end only during sequencing. On the other hand, **paired-end sequencing** allows to sequence both ends of a fragment, producing twice the number of reads for the same time and effort in library preparation and providing more accurate read alignment and the ability to detect insertion-deletion variants, which is not possible with single-read data [26]. An additional advantage of paired-end RNAseq is the opportunity to detect chimeric transcripts resulting from gene fusion events when sequencing cancer transcriptomes [4].

3.1.4 NGS workflow

NGS platforms, depending on the technology used, differ from each other in terms of read length, data produced, and data quality. NGS can also be used in a number of ways depending on the application, which can be classified into whole genome sequencing, whole exome sequencing, whole transcriptome sequencing and can also be applied to a subset of genes (targeted sequencing) [27]. However, in every type of sequencing, the major changes occur in the sample processing and library preparation steps. Once the library is prepared, a particular sequencing platform can use the same chemistry to sequence the fragments. [23]

The next-generation RNA sequencing workflow itself is divided into four main steps.

1. First, before preparing an RNA library itself, some preparations may need to be made first. For RNA sequencing, total RNA is isolated from a sample of interest, which, depending on the type of RNA to be profiled, may be purified to enrich for mRNAs, microRNAs, lincRNAs, etc.
2. Second, **library preparation**, which may involve steps such as reverse transcription to cDNA (complementary DNA), PCR amplification, and may or may not preserve strandedness information [4].
3. The third step generates the actual sequence via the chemistries for each technology [28]. **Sequencing** can produce one read in a single-end sequencing reaction, or two ends separated by an unsequenced fragment in paired-end reactions [4].
4. And last but not least, when the signal is converted to data, the data need to be converted to interpretable information, and the information into actionable knowledge, all as part of **downstream data processing** [23], on which we focus more in Section 3.2 NGS data processing.

When NGS was introduced, the main challenges started to shift from sequencing itself to downstream bioinformatics. Technological advances make it continually faster and cheaper to produce genomic sequence data than to store, manage, and analyse them [29], leading to a "data deluge" problem. For example *"the compressed single-end sequencing data from one flow cell of an Illumina HiSeq 2500 might be 20 GB and twice as large once uncompressed to allow for processing and manipulation."* [4]

3.2 NGS data processing

Downstream data analysis of RNA sequencing data consists of quality control, trimming of sequencing adapters, and removal of reads with poor quality scores, followed by mapping reads, differential expression analysis, identification of novel transcripts, and pathway analysis [4].

In general, the NGS data processing workflow can be broken down into three main components: primary, secondary, and tertiary analyses [24].

3.2.1 Primary analysis

Primary analysis consists of the detection and analysis of raw data (signal analysis), targeting the generation of legible sequencing reads (base calling) and scoring base quality [24], resulting in, for example, FASTQ files. However, this whole process takes place primarily on board the NGS instrument [23]. Primary analysis also includes the pre-processing of NGS reads to ensure that only high-quality reads of the optimal length are used for downstream analysis.

Pre-processing prepares sequences for read alignment and can include read filtering, demultiplexing, and trimming [4]:

- **Filtering** is a process in which reads are filtered out of the data based on base call quality (Phred score) and the length of the read. Poor confidence-base calls can lead to the detection of false-positive variants, so they need to be removed. Reads that are too short are likely to align with multiple regions in the genome and cause poor mapping metrics.
- Multiplexing in NGS refers to multiple samples being sequenced simultaneously on the same instrument. Thus, **demultiplexing** is needed and refers to the separation of sequencing reads into separate files according to the barcode index used for each sample.
- **Trimming** removes adaptor sequences ligated to the ends of libraries during the library preparation process, as they can interfere with mapping and assembly. The reads are also trimmed to remove poor quality bases from the ends of the reads. [30]

3.2.2 Secondary analysis

Making sense of RNAseq data is very dependent on the question of interest. However, the main steps are common for most applications: read mapping, quantifying the expression levels of genes, transcripts, and exons, and then differential analysis of gene expression. [31]

After direct sequencing of the cDNA fragments, it is not known which reads came from which transcripts. Therefore, transcripts need to be reconstructed by **mapping** short RNAseq reads [32].

There are two main strategies (see Figure 3.1):

- The de novo assembly approach is used when working on an organism without a reference genome [33]. Reads are first assembled into longer contigs, and these contigs can then be considered as the expressed transcriptome to which reads are remapped for quantification [34].

- The second approach is aligning the reads to a reference genome or reference transcripts [33]. This approach allows the discovery of new, unannotated transcripts [34].

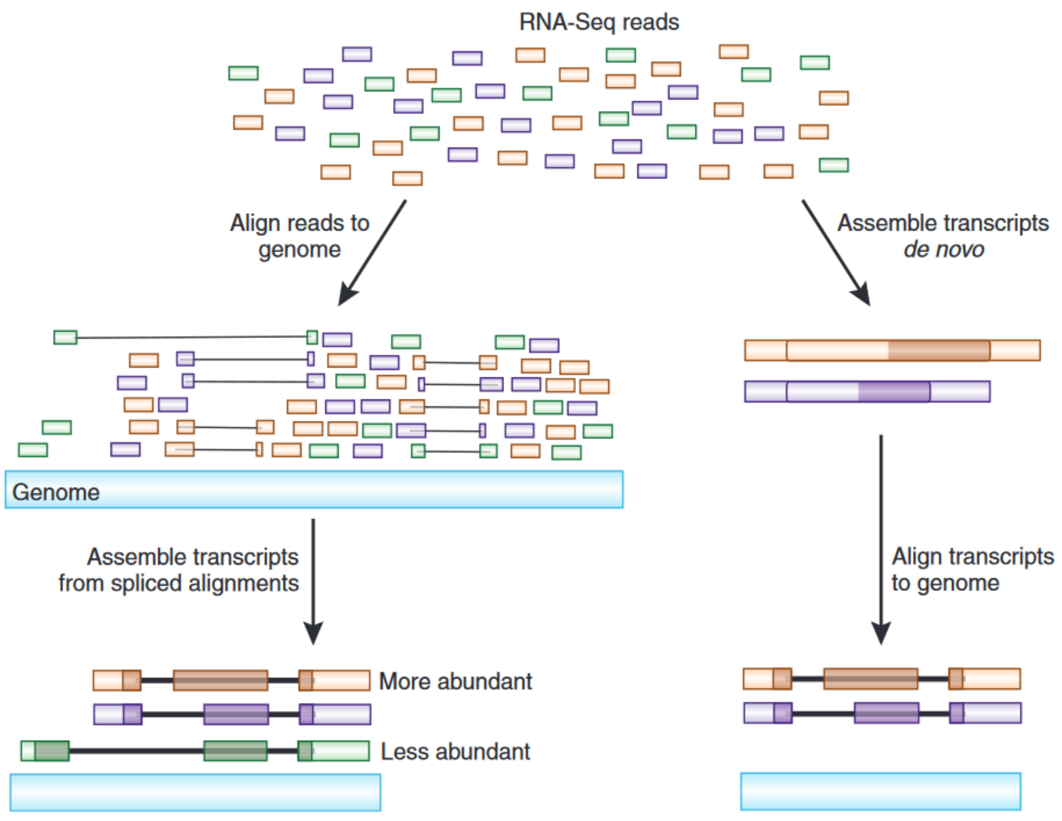

RNA-Seq reads
Align reads to genome
Assemble transcripts *de novo*
Genome
Assemble transcripts from spliced alignments
Align transcripts to genome
More abundant
Less abundant

Figure 3.1: Strategies for reconstructing transcripts from RNAseq reads [32]

After read mapping, the following step is **read counting and quantification** of expression levels of genes, transcripts, and exons [31], where gene expression is measured by the number of reads mapped to a gene [35]. Obtaining the transcriptome expression profile requires genomic elements (i.e. genes, transcripts, or exons) to be defined in the context of the genome. There are many aspects that affect gene quantification, from the choice of genome annotation [36] to taking into account alternative RNA splicing [37] and isoforms [36]. Gene quantification algorithms can be divided into two categories: transcript-based approaches and union-exon-based approaches [31].

The next important step is **normalisation**. The aim is to remove systematic technical effects that occur in the data [38]. The easiest way to normalise the difference in the sizes of the sequencing library is to rescale the total read counts, but this approach is too simple, because the RNA sequencing counts inherently represent the relative abundances of genes in a sample. The number of reads mapped to a gene is not only dependent on the expression level and length of the gene, but also on the composition of the RNA population that is being sampled [31].

The main aim of most RNA-seq studies is **differential analysis** to identify differentially expressed genes between distinct sample groups [31]. "*Differential expression analysis means taking the normalised read count data and performing statistical analysis to discover quantitative changes in expression levels between experimental groups. For example, we use statistical testing to decide whether, for a given gene, an observed difference in read counts is significant, that is, whether it is greater than what would be expected just due to natural random variation*." [34]

Methods for differential gene expression analysis can be grouped into two subsets [39]:

1. **Parametric methods** capture all information about the data within the parameters. Each expression value for a given gene is mapped into a particular distribution, such as the Poisson or negative binomial.
2. **Non-parametric methods** can capture more details about the data distribution, i.e., not imposing a rigid model to be fitted. It is possible because non-parametric models take into consideration that data distribution cannot be defined from a finite set of parameters, thus the amount of information about the data can increase with its volume. [39]

There is no consensus about which methodology is most appropriate or which approach is better in terms of robustness, accuracy, and reproducibility. The results of differential gene expression analysis are influenced by many factors at almost every step of the RNAseq analysis, from library preparation and structure of the experiment [39], to normalisation [31].

Gene fusion detection, which is part of the secondary analysis, is described in more detail in Section 3.3 Fusion gene detection.

3.2.3 Tertiary analysis

Last but not least, the relevance of the data produced is evaluated from a biological context [39] as the final step of the entire bioinformatic analysis pipeline. To cite Sean Scott from QIAGEN in Tanya Samazan's article [40] "*The core of tertiary analysis is what we refer to as 'interpretation.' Interpretation involves the biological classification of observed variants, determination of the clinical relevance of these variants, the deemed action-ability of these variants in terms of treatment options and extends to the ordering physician in terms of how clinically helpful the results or recommendations are.*"

And he also continues "*When a cancer patient has progressed to a late-stage cancer, they've moved through the initial steps of standard care practices and they're looking for alternative treatment and/or investigational drug options. Interpretation can involve not just the molecular and genomic profiling of a patient, but the assessment of how the diagnostic, theranostic, resistance or prognostic data enables a medical oncologist to identify and select the right targeted therapies or combination of therapies for their patient, based upon the evidence indicating that the treatment(s) may be efficacious for a patient-specific cancer type and genomic molecular profile and likely improve the patient's outcome.*"

3.3 Fusion gene detection

A number of new software tools have quickly emerged to identify structural variants, as well as gene fusions resulting from these variants [27]. Implementations of the various prediction methods vary in the read alignment tools employed, the genome database and gene set resources used, and criteria for reporting candidate fusion transcripts and for filtering out likely false positives [3]. Available fusion predictors vary in prediction accuracy [19], installation complexity, execution time, robustness, and hardware requirements [3].

Fusion genes in cancer samples can be detected by finding novel transcripts in RNAseq data [41]. In RNAseq datasets (as well as in DNAseq), gene fusion detection is based on unmapped and discordant read fragments. In RNAseq, fusion detection is further complicated by intron–exon boundaries [19]. Based on computational strategies for fusion gene detection, the methods can be grouped into two categories [27, 42] (see Figure 3.2):

1. **Mapping-first approaches** that align RNAseq reads to genes and genomes to identify discordantly mapping reads that are suggestive of rearrangements [3]. The mapping-first approach is faster and more commonly used, than the assembly-first approach [27] and therefore discussed in more detail below.
2. **Assembly-first approaches** that directly assemble reads into longer transcript sequences followed by identification of chimeric transcripts consistent with chromosomal rearrangements. [3]. For an assembly algorithm, if it assembles short reads directly without mapping them to the references, then it is called de novo assembly. The exclusive advantage of de novo assembly is that it does not need a reference genome/transcriptome for fusion detection, however, it can be too time-consuming and too error prone. [27]

The fusion gene detection methods overall follow the same three steps: mapping and filtering, fusion junction detection, and assembly and selection of the fusion gene. [27]

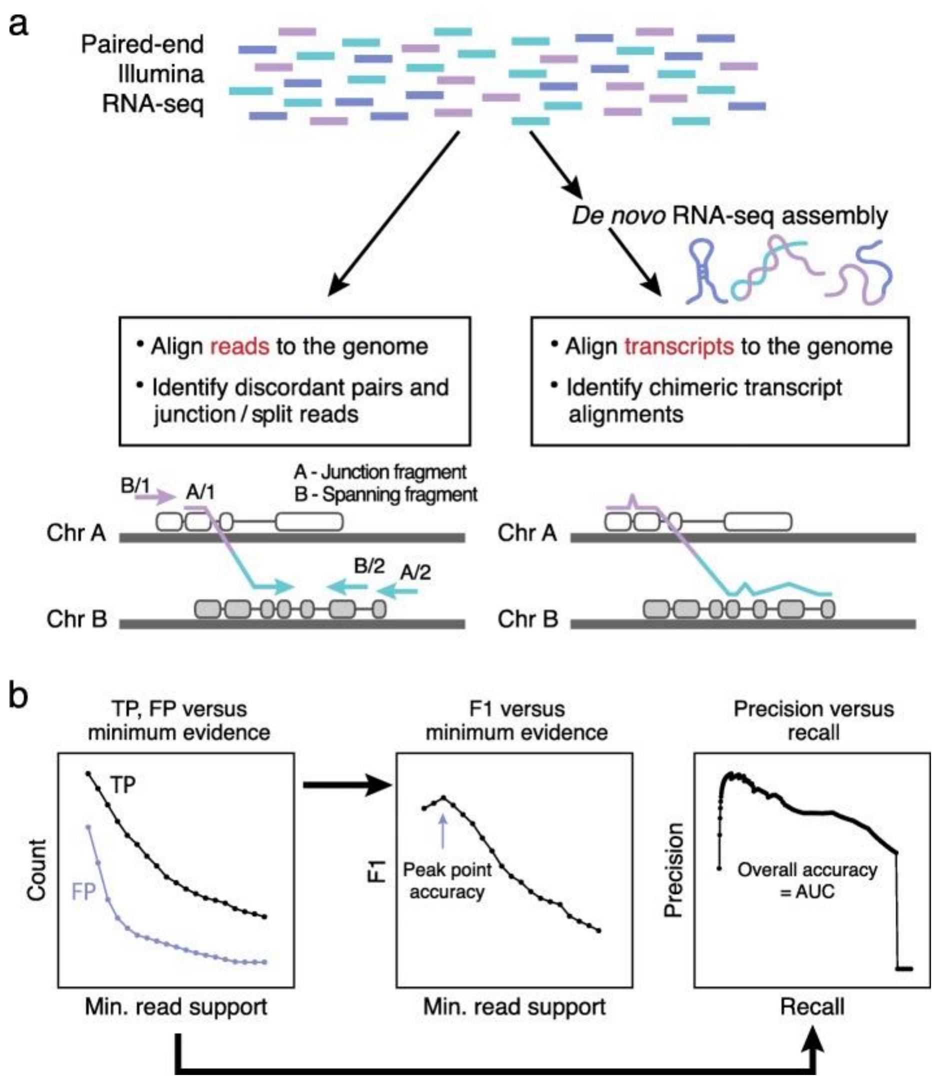

a
Paired-end Illumina RNA-seq
De novo RNA-seq assembly
• Align reads to the genome
• Identify discordant pairs and junction / split reads
• Align transcripts to the genome
• Identify chimeric transcript alignments
A - Junction fragment
B - Spanning fragment
B/1
A/1
B/2
A/2
Chr A
Chr B
Chr A
Chr B
b
TP, FP versus minimum evidence
TP
FP
Count
Min. read support
F1 versus minimum evidence
F1
Peak point accuracy
Min. read support
Precision versus recall
Precision
Overall accuracy = AUC
Recall

Figure 3.2: Methods for fusion transcript prediction and accuracy evaluation. [3]

Evidence supporting predicted fusions is typically measured by the number of RNAseq fragments found as chimeric (split or junction) reads that directly overlap the fusion transcript chimeric junction, or as discordant read pairs (bridging read pairs or fusion spanning reads) where each pair of reads maps to opposite sides of the chimeric junction without directly overlapping the chimeric junction itself. [3]

3.3.1 Mapping-first approach

As illstrated in Figure 3.3, the initial step is **mapping** (among the known and widly used mapping tools are for example Bowtie, STAR or BLAT) followed by evaluation of each aligned read (pair) [27, 42]. The reads unrelated to fusions are removed from further consideration. The methods that are based primarily on ‘split reads’ filter out all mapped reads. However, for methods that exploit ‘spanning reads’, all discordantly mapped pairs are preserved. In addition to discordantly mapped reads, the unmapped reads (potentially ‘split reads’) are also kept, in order to assist in the selection of fusion candidates [27], [43].

To further discard reads that are less likely to harbor fusions, most fusion gene detection tools developed an additional **filtering** techniques [27, 44, 45]. An example of a commonly used filter relates to intra-chromosomal fusion. A fusion candidate is discarded if the distance between its fusion partner genes is smaller than D (a defined threshold), suggesting that it is a read-through transcription and not an intra-chromosomal fusion. [27, 44]

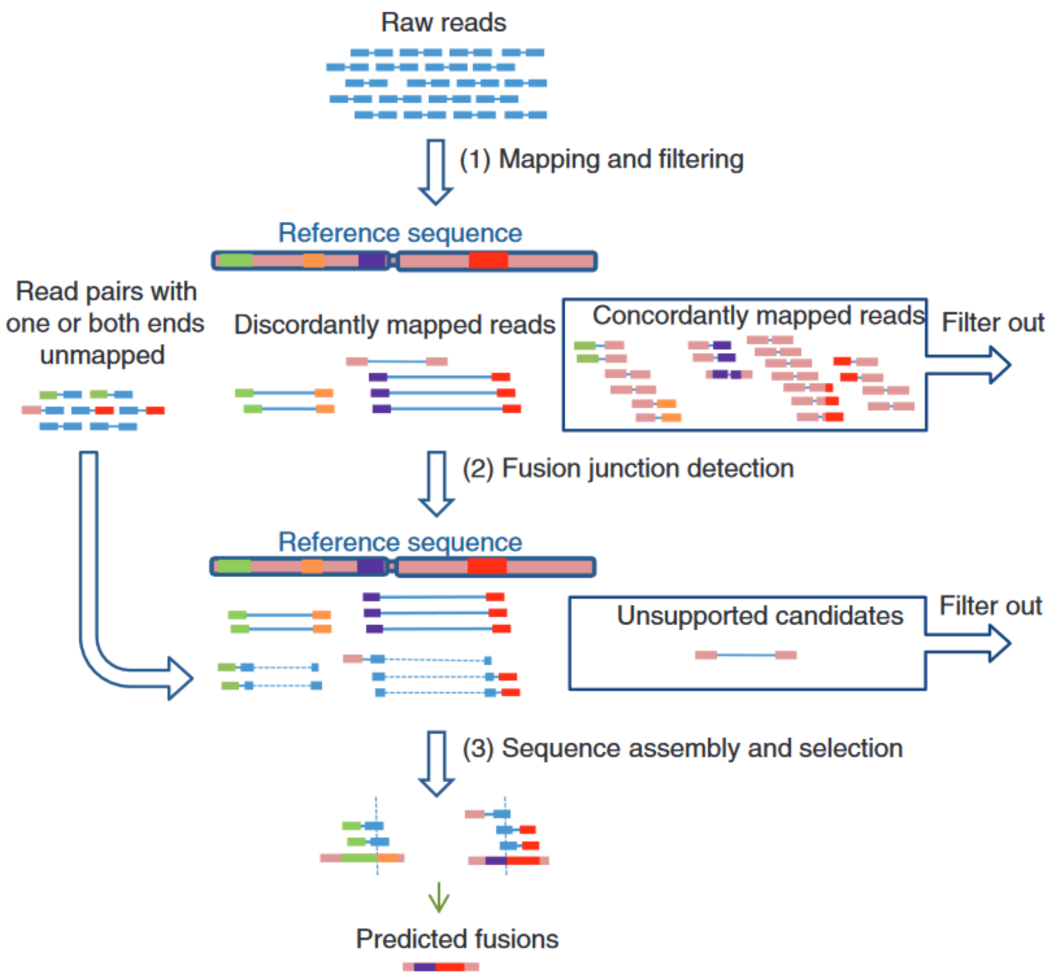

Raw reads
(1) Mapping and filtering
Reference sequence
Read pairs with one or both ends unmapped
Discordantly mapped reads
Concordantly mapped reads
Filter out
(2) Fusion junction detection
Reference sequence
Unsupported candidates
Filter out
(3) Sequence assembly and selection
Predicted fusions

Figure 3.3: A procedure to detect gene fusions through the mapping-first approach [27]

Next step is the **detection of fusion junctions** through either ‘split read’ or ‘spanning read’ mapping [27, 45]. Spanning reads contain one read located in different genes, while split reads indicates a single read overlapping on two different genes [45].

1. When leveraging the **‘split reads’**, the unmapped reads are cut into multiple pieces. The first and last segments of each ‘split read’ are then mapped against the reference sequences independently. If the two end segments of a ‘split read’ are mapped to two different chromosomes or genes, then the read is potentially from a fusion gene. Once this alignment pattern is detected, the precise location of the fusion junction can then be found by adjusting the boundaries of the original fragments and performing realignment. [27]
2. A different strategy to detect fusion junctions is to infer fusion breakpoints from **‘spanning reads’** and then select those predictions that are likely to be real using ‘split reads’ [27, 42]. Discordant alignments are first grouped into clusters, each consisting of a maximal set of reads that share the same pair of breakpoints. Then, the boundary region of each candidate fusion junction is identified from its cluster. Next, fusion junction loci are inferred and putative fusion transcripts are predicted. Finally, unmapped reads are aligned to the predicted fusion transcripts. The predictions, to which the highest number of unmapped reads is aligned, are nominated as candidate fusion genes. [27]

After identifying fusion junctions, the sequence of each candidate fusion gene can be derived by joining the two partner genes together. Previously unmapped reads are then aligned to the candidate fusions. Reads mapped in this step (called ‘supporting reads’) provide an additional layer of confidence to the fusion candidates [42].

Many existing methods require the presence of supporting reads as a prerequisite to nominate a fusion. The requirement for more supporting reads removes more inauthentic candidates, however, by risking discarding true fusion genes of low transcription level or coverage. [27]

To help distinguish true fusions from candidates expressed at low transcription levels, it is common to include scoring functions to rank fusion candidates [42]. The candidates with the maximum likelihood to be real fusions are selected as final outputs. These scoring functions are mostly based on features including mapping quality [27, 43], number of supporting reads and read depths. [27]

3.3.2 Fusion expression levels and fusion allelic ratios

After predicting the fusion candidates, it is a good practise to provide a reasoning about the quality of evidence supporting the fusion genes, since they can be biologically relevant or derived from an experimental or bioinformatic artifact. Here we present two characteristics of relevance, fusion fragment per million and fusion allelic ratios, as we include them in the output file. These values are used in a so called COSMIC like fusion prediction. [46]

Fusion fragment per million (FFPM) is a normalized measure of the quantity of RNAseq fragments supporting the fusion event, in a sense it represents the fusion fragments per million total reads and can be used as an approximation for fusion expression level [46, 47].

Fusion allelic ratios (FAR) is computed as a pragmatic way of estimating the relative expression of the fusion transcript as compared to the alternative non-fused allele of the gene. FAR "*is computed separately for each of the fused genes and reflects the proportion of evidence supporting the fused-allele as compared to the number of reads supporting the non-fused allele and overlapping the fusion transcript breakpoint. While this is not a highly accurate measurement of the ratio of fused vs. non-fused transcripts, it's a useful metric to serve as a proxy for this information*." [46]

4 Aims of thesis

The aim of this thesis is to develop a bioinformatic pipeline that will process data from RNAseq analyses of leukemia patients with the objective of finding the fusion gene candidates. The input to this pipeline will be the raw sequencer data in FASTQ format and the output will be the analysis of fusion genes in individual patients preprepared for expert analysis.

The sub-objectives are:

- Describe the capabilities of the NextFlow tool for the execution of bioinformatic pipelines, including deployment.
- Create a functional bioinformatic pipeline for processing fusion gene analysis of patient samples.
- Creation of a summary output adapted for expert analysis.
- Creation of documentation and user manuals for the installation and use of the pipeline.

5 Methods

The aim of this thesis was to build a pipeline that would effectively detect fusion genes in the given samples. For the individual tasks, including data pre-processing, fusion genes detection, results validation, or visualisations, we chose open-source tools with available maintained repositories based on certain criteria. To keep reproducibility and maintenance simple, each tool was “dockerized” and the images were made available through the Docker hub.

To connect these steps together and to create the pipeline itself, we leveraged Nextflow, a bioinformatics framework which not only makes integrating the processes very easy, but also simplifies the deployment on cloud or cluster and offers the resources management.

Since the amount of data processed in such analysis is large as well as the computational resources required by the tools, it is not always possible to run the pipeline on commonly used personal devices. Therefore, we developed the pipeline leveraging the grid and cloud services offered by MetaCentrum, a virtual organisation which not only gives access to immense computational resources, but also supports Kubernetes deployment and Nextflow module.

5.1 Nextflow

NextFlow is free open-source software for bioinformatics workflow managing, which responds to the increasing need of analytical tools for big data, which are often inadequate or require knowledge of complex low-level tools. The framework is, in other words, pipeline orchestrator and a programming domain specific language, which utilises parallelisation and distributed computing to facilitate the writing of data-intensive computational pipelines.

Among NextFlow features are enabling workflows portability and reproducibility, simplifying parallelization and large-scale deployment, and easing integration of existing tools, systems, and industry standards.

NextFlow is JVM (Java Virtual Machine) application and can be used on any POSIX compatible system (e.g., Linux, OS X, Solaris), requiring BASH and Java 8 or higher to be installed.

The NextFlow pipeline script is made by joining together different processes, where each process can be written in any scripting language that can be executed by the Linux platform (Bash, Perl, Ruby, Python, etc.) and is independent and isolated from another. NextFlow also makes it very easy to change the target system, where processes are executed without any other modifications, thus providing an abstraction between the functional logic of the pipeline and the underlying execution system. [48]

5.2 MetaCentrum

The data analysis requires certain amount of computing and storage capacity – executing locally is very useful for development and testing, but for real world computational pipelines high performance computing or cloud is often required. This need is met by MetaCentrum VO, CESNET's regional virtual organisation of the Czech National Grid Organisation MetaCentrum (NGI), which offers software and hardware infrastructure for academic and research purposes. Registered users get access to grid services including consistent and dependable computational resources (which exceed possibilities of individual supercomputing centres), along with various application software and qualified user support. In addition, they provide cloud services and an open-source framework for distributed storage and processing of large amounts of data [49]. All that is relevant in terms of RNAseq data analysis, since manipulating RNAseq data is computationally intensive and typically requires access to a powerful resource [4].

MetaCentrum is also used by many great Czech academic or research teams from a wide range of specialisations, from biochemistry, nuclear or medical research to bibliography or social research.

Grid services

Grid computing is a distributed computing model that provides a network of widely distributed computer resources working together to perform a task. Compared to other approaches, rather than achieving high performance computational needs by having large clusters of similar computing resources or a single high performance system, such as a supercomputer, grid computing attempts to harness the computational resources of a large number of dissimilar devices [50]. In addition, another difference between grids and cluster computing is that the resources are not limited to physical computing resources but may include application-specific resources such as files and databases [51].

For the end users or applications of the grid, the grid looks like a virtual machine with powerful capabilities with a single point of access for performing tasks [52, 53]. The essence of grid computing is to manage heterogeneous and loosely coupled resources in an efficient way in this distributed system and to coordinate these resources through a task scheduler so they can complete specific cooperative computing tasks. [52]

Grid computing typically leverages the spare CPU cycles of devices that are not currently needed for a system's own needs, and then focusses them on the particular goal of the grid computing resources [50].

Cloud services

Cloud computing is a method of sharing resources via the Internet. The main features of cloud computing are:

- Virtualisation, where cloud computing platforms and applications are built based mostly on resource virtualisation technology, which plays an important role in improving resource efficiency and increasing service reliability and security.
- Flexibility, where cloud resource platforms can dynamically expand or reduce in size depending on user needs, which reduces the investment risk for the user and meets the needs of different users.
- Cloud computing offers on-demand service, meaning that services can be provided according to the actual needs of users.
- Cloud computing platforms use dynamic network management systems to monitor the status and efficiency of each resource node, to dynamically migrate nodes that have low efficiency or failure, and to ensure that overall system performance is not affected, resulting in high reliability.
- Cloud rental resources must be highly customisable. Infrastructure as a service allows users to deploy specialised and virtual appliances. [52]

MetaCentrum Cloud is the Infrastructure as a Service cloud on top of open-source OpenStack project. Their most important cloud resources are virtual machines, virtual networking, private and/or public IP addresses, storage, and cloud load balancers. [54]

5.3 Docker

Docker is an open-source platform that runs applications and makes the process easier to develop and distribute. It allows developers to package their applications and all their dependencies into a standard called **containers**, where the applications are virtualized and executed. Containers provide a lightweight, isolated environment that ensures consistent behaviour across different operating systems and infrastructure, meaning that they provide an extra layer of abstraction. [55]

Every Docker container is built on what is called an **image**, which includes all the necessary libraries, dependencies, and configuration files needed to run the application [56]. Images can be easily shared and deployed on any machine that has Docker installed. The foundation of every image is a base image, which can be an operating system image (e.g. Ubuntu) or any other accessible one. [55]

One of multiple methods to build an image, which we use in this thesis, is to create a **docker file**. A Dockerfile is a text document in which the user defines the build steps for the application. The Dockerfile is written in a domain-specific language called the Dockerfile syntax. When the command 'docker build' is run from the bash terminal, it follows all the instructions given in the docker file and builds an image. [55]

One of Docker's features is the ability to find, download, and start container images that were created by other developers quickly. The place where images are stored is called a **registry** [57] (e.g. Docker Hub).

5.4 Kubernetes

Kubernetes is an open-source container orchestration system. It controls containerized applications across multiple hosts and handles deploying, monitoring, and scaling containers [58], reducing repetitive manual processes involved in container deployment and management [59].

Kubernetes has at its core a shared persistent store, with components monitoring changes to relevant objects [60]. The smallest deployable unit in Kubernetes is a **pod** which consists of one or more containers. For performance isolation, Kubernetes offers the resource management technique for users to define the computing resources (e.g., CPU and memory) for the pods. [58]

5.5 Data processing tools

In the pipeline, we include multiple open-source tools, where each covers a different step of the analysis. For pre-processing we used Fastp, for alignment and indexing STAR and Samtools, for fusion genes detection then JAFFA, FusionCatcher, Arriba, STAR-Fusion, and Cicero and for validation FusionInspector. The following tables Table 5.1.a and Table 5.1.b introduce each of them, their key role in this thesis, parameters, and requirements.

We chose various fusion genes detection tools, since each can report a different set of fusions based on their algorithm. Therefore, our analysis can offer a wider range of results and a higher probability of finding the relevant gene fusions. For the selection of the tools, we used the following criteria:

- The ability to detect gene fusions relevant for the leukemia diagnosis
- Active maintenance and availability, preferably not more than 2 years from the last update
- Support of the hg38/GRC38 reference
- Preferably available Dockerfile

Table 5.1 a: Overview of the utilized bioinformatic tools

Tool	Role in the pipeline	GitHub repository	Source	Minimal RAM (GB)
Fastp	Data pre-processing	https://github.com/OpenGene/fastp	[61]	4
STAR	Transcripts alignment to a reference	https://github.com/alexdobin/STAR	[62]	16 – 32
Samtools	BAM file sorting and indexing	https://github.com/samtools/samtools	[63]	Unknown
JAFFA (Direct mode)	Fusion genes detection	https://github.com/Oshlack/JAFFA	[64]	10 – 75
FusionCatcher	Fusion genes detection	https://github.com/ndaniel/fusioncatcher	[65]	24
Arriba	Fusion genes detection	https://arriba.readthedocs.io/en/latest/	[17]	10
STAR-Fusion	Fusion genes detection	https://github.com/STAR-Fusion/STAR-Fusion/wiki	[3]	< 16
Cicero	Fusion genes detection	https://github.com/stjude/CICERO	[66]	24
FusionInspector	Results validation	https://github.com/FusionInspector/FusionInspector	[46]	> 40

Table 5.1 b: Overview of the utilized fusion genes detection tools

Tool	Supported reference	Source of reference files*	Input files	Approach	Aligner
JAFFA (Direct mode)	hg38, hg19, mm10	Provided	Reads as FASTQ files	Mapping-first	Bowtie2, BLAST, BLAT
FusionCatcher	hg38, hg19	Provided	Reads as FASTQ files,	Mapping-first	Bowtie, BLAT, STAR, Bowtie2
Arriba	hg19, hg38, mm10	Provided	Aligned reads as BAM file or reads as FASTQ files	Mapping-first	STAR
STAR-Fusion	hg19, hg38	Provided	JUNCTION file from STAR	Mapping-first	STAR
Cicero	hg19, hg38	Provided	Reads as FASTQ files or aligned reads as BAM file with BAI index	Assembly-first	BLAT

* The tools require additional files, which can, depending on the tool, include reference files, indexes, files built from Ensembl database, blacklists, etc.

The fusion callers that we did not include due to their unavailability are Comrad, FusionAnalyser, ShortFuse, and FusionQ. FusionMap officially announced the end of maintenance. We also did not include Bellerophones, BreakFusion, Chimerascan, nFuse, FusionHunter, FusionMap, MapSplice, TopHat Fusion, InFusion, FusionSeq, FusionScan, Ericscript, Gfusion and Pizzly due to their inactivity in maintenance, and FuSeq and SOAPfuse due to their lack of support of containerisation.

5.5.1 Fastp

Fastp is an open-source tool developed in C/C++ providing all the necessary pre-processing operations for FASTQ files. It can perform quality profiling, adapter trimming, read filtering, deduplication, and base correction with a single scan of the data. It supports both single-end and paired-end short read data and also provides basic support for long-read data. One of the advantages is the multi-threading support, which is useful for our pipeline. In addition to the pre-processed data, a report in both HTML and JSON format can be generated, which allows for direct comparison of quality statistics altered by pre-processing. [61]

Method

Among the steps computed by Fastp are:

1. Trimming – Fastp includes two methods. First there is adapter trimming, which is based on computing k-mer and on a tree-based algorithm. The second method is polyG and polyX tail trimming, which focusses on fixing issues observed in Illumina NextSeq and NovaSeq series.
2. Base correction – Fastp corrects mismatches in an overlap of pair reads, since if the reads are of high quality, they are usually completely reverse-complemented. However, the correction is performed only if the base pairs have an imbalanced quality score and the total mismatch is below a given threshold.
3. Sliding window quality pruning – To improve read quality, the method marks the bases in the window as discarded if the average quality is lower than a given threshold. [61]
4. Deduplication – Fastp removes duplicate reads, which reduces potential false positive results and bias in results affected by PCR duplicates in the subsequent analysis and reduces the size of the files.

5.5.2 STAR

STAR (Spliced Transcripts Alignment to a Reference) is a C++ tool for aligning RNA sequencing reads to a reference genome and detection of novel splice junctions [62].

Method

STAR was designed to align the non-contiguous sequences directly to the reference genome where the algorithm consists of two major steps: seed searching step and clustering/stitching/scoring step.

1. For every read that STAR aligns, STAR will search for the longest sequence that exactly matches one or more locations on the reference genome. These longest matching sequences are called the Maximal Mappable Prefixes. The different parts of the read that are mapped separately are called 'seeds'. STAR will then search again for only the unmapped portion of the read to find the next longest sequence that exactly matches the reference genome. This sequential searching of only the unmapped portions of the reads underlies the efficiency of the STAR algorithm. STAR uses an uncompressed suffix array to efficiently search for the Maximal Mappable Prefixes, this allows for quick searching against even the largest reference genomes.

   If STAR does not find an exact matching sequence for each part of the read due to mismatches or indels, the previous Maximal Mappable Prefixes will be extended. If extension does not give a good alignment, then the poor-quality adapter sequence will be soft-clipped.

2. Separate seeds are stitched together to create a complete read by first clustering the seeds together based on proximity to a set of ‘anchor’ seeds, or seeds that are not multi-mapping. Then the seeds are stitched together based on the best alignment for the read (scoring based on mismatches, indels, gaps, etc.). [62]

5.5.3 Samtools

SAMtools is a C or Java package providing utilities for post-processing alignments in the SAM/BAM format, including converting formats, indexing, sorting, or merging alignments. SAMtools was developed by the creators of the SAM format, and both together offer a generic and modular approach that separates the alignment step from downstream analyses. [63, 67]

5.5.4 JAFFA

JAFFA is a multi-step pipeline for gene fusions detection built using the Bpipe platform. They introduce a new method that can be applied to any read length (reads from 100 bp up to full-length transcripts), single- or paired-end.

JAFFA can be run in three modes, depending on the reads length:

- Direct mode is recommended for 100bp reads or longer and uses the mapping-first approach.
- Long mode is recommended for high-error long reads.
- The hybrid mode is recommended for low error rate sequencing of 70-95bp and uses both mapping and assembly approach.
- The assembly mode is recommended for low error rate short reads of <70bp and uses the assembly-first approach.

Method

Unlike the usual approach used by the majority of other tools, JAFFA compares a tumour transcriptome with the reference transcriptome instead of the reference genome. Such approach brings multiple advantages including the avoidance of error-prone splice site alignment and therefore simplified identification of fusion transcripts. Another advantage is that the reference transcriptome consists of less sequence than the reference genome, allowing slower, but more accurate alignment algorithms to be used. That allows JAFFA to analyse even longer reads.

The pipeline consists of 6 steps.

1. RNAseq reads are first filtered to remove intronic and intergenic reads. 50 bp reads would then be assembled into contigs using Oases. For longer reads, this step is not necessary.
2. The resulting tumour sequences are then aligned to the reference transcriptome and those that align to multiple genes are selected. These contigs make up a set of initial candidate fusions.
3. Next, the pipeline counts the number of reads and read pairs that span the breakpoint.
4. The candidates are then aligned to the human genome. Genomic coordinates of the breakpoint are determined.
5. Further selection and candidate classification is carried out using quantities such as genomic gap size, supporting reads, and alignment of breakpoints to exon-exon boundaries.
6. A final list of candidates is reported along with their sequence. [64]

5.5.5 FusionCatcher

FusionCatcher is a software tool for finding fusion genes in paired-end RNAseq data.

Method

FusionCatcher includes both aligning the sequencing reads on transcript and mapping on genome. The method consists of multiple steps. First, preprocessing and filtering of RNA sequencing data are performed, including quality control. Second, FusionCatcher uses an ensemble approach consisting of four different methods and four different aligners for identifying the fusion junctions. Each method corresponds to one aligner where the aligners are Bowtie, BLAT, STAR, and Bowtie2.

The Bowtie method uses information regarding the exon/intron positions (i.e. genome annotation). It serves as a filter: the unmapped reads, which are the reads which passed the quality filtering and do not map on the transcriptome and the genome, are kept for further analyses. Therefore, the number of unmapped reads given as input to the next three methods (which are more computationally demanding) is reduced.

Reads mapping on the transcriptome are used further to build a preliminary list of candidate fusion genes by searching for pairs of genes. The pairs in the list are then filtered and removed, using known and novel criteria that make biological sense.

The unmapped reads, which still remain unmapped after aligning during the Bowtie method, together with the reads, which support the candidate fusion genes, are further aligned using the BLAT aligner, the STAR aligner, and the Bowtie2 aligner. Only candidates who pass multiple criteria will make it to the final list of fusion genes. [65]

5.5.6 Arriba

Arriba is the winner of the DREAM SMC-RNA Challenge, an international competition organised by ICGC, TCGA, IBM and Sage Bionetworks in 2018. Their method builds on the output of STAR, which is by default included in the Arriba workflow but can be run separately.

Arriba also provides an R script for visualisation of detected fusion genes, which is capable of taking output files from either Arriba or STAR-fusion/FusionInspector.

Method

Conceptually, Arriba is nothing more than a collection of filters. The generation of fusion candidates is entirely handled by STAR, which collects all evidence about potential gene fusions in the chimeric alignment file. Most of the candidates in these files are alignment artifacts, in vitro-generated artifacts, or transcript variants that are also observed in healthy tissue. Arriba applies a set of filters which try to detect artifacts based on various features that are characteristic for artifacts. The filters are either read-level, which assess candidates based on information contained in a single read (pair), or event-level, which integrate information from multiple reads. The complete list with descriptions can be found in the Arriba documentation. [17]

5.5.7 STAR-Fusion

STAR-Fusion is a component of the Trinity Cancer Transcriptome Analysis Toolkit (CTAT) implemented in Perl. As with Arriba, STAR-Fusion analyses the output of STAR.

Method

STAR-Fusion uses chimeric read alignments reported by STAR in its Chimeric.out.junction file to identify candidate fusions. It maps the reads to exons of reference gene structure annotations based on coordinate overlaps. STAR-Fusion primarily focuses on filtering the alignment evidence and preliminary fusion predictions to remove likely artifacts and likely false-positive predictions.

The method involves multiple steps:

1. Read alignments between pairs of genes that are localized to sequence similar regions between those genes are excluded.
2. A database of all-vs-all blastn matches between all reference cDNA sequences is queried to identify regions of sequence similarity between candidate fusion genes.
3. If chimeric read alignment evidence overlaps sequence similar regions, the alignment is discarded. Duplicate paired-end read alignments are removed, and the remaining alignments are assigned to preliminary fusion gene pair candidates.
4. STAR-Fusion selects those candidate gene pairs for which the fusion-supporting evidence indicates a sense-sense orientation between the fusion pairs and scores them according to the number of split reads supporting the fusion breakpoint and the number of paired-end fragments that span the breakpoint.
5. These preliminary fusion gene candidates are filtered in two stages: a basic filtering stage that requires mini- mum fusion evidence support and an advanced filtering stage that examines characteristics of the genes involved in the candidate fused gene pairs. [3]

5.5.8 Cicero

Cicero is a local assembly-based algorithm for fusion gene detection written in Perl. It aims to overcome the limitations of existing RNAseq analysis methods, which compare gene fusions detected by RNAseq with structural variations discovered by whole-genome sequencing. The local assembly takes advantage of the longer RNAseq read length (≥ 75 bp) generated by NGS and therefore allowing the detection of non-canonical fusions and ITDs.

Method

Cicero integrates RNAseq read support with extensive annotation for candidate ranking. The algorithm is implemented through the three key steps.

1. Fusion detection by de novo local assembly at candidate breakpoints (which consists of identification of candidate fusion breakpoints marked by soft-clipped reads, assembly of the fusion contig, and mapping of the fusion contig for discovery of the partner locus breakpoint) and analysis of splice junction reads (for fusion transcripts lacking soft-clipped reads).
2. Fusion annotation including a reading frame check for the fusion partner genes.
3. Ranking of candidate fusions based on the supporting evidence in RNAseq and matches to known fusions (the ranking is based on fusion allele frequency, matching length, repetitive mapping, and frame-check results with a quality status determined by matches to known fusion events or artifacts). [66]

5.5.9 FusionInspector

FusionInspector, which serves as a level of validation in our pipeline, is a component of the Trinity Cancer Transcriptome Analysis Toolkit (CTAT) and assists in fusion transcript discovery by performing a supervised analysis of fusion predictions, attempting to recover and re-score evidence for such predictions. Through reports, interactive visualisations, and classification, FusionInspector assists researchers in reasoning about the quantity and quality of the evidence supporting predicted fusions, to differentiate likely artifacts from fusions with characteristics similar to biologically relevant fusions known to occur in tumours and normal tissues. [46]

FusionInspector uses the set of genomic resources in the genome library identical to that used with STAR-Fusion, including the human reference genome, annotations to the gene structure, and the STAR genome index.

Method

As input, FusionInspector takes a list of candidate fusion genes found by fusion transcript prediction tools. Then "*extracts the genomic regions for the fusion partners and constructs mini-fusion-contigs containing the pairs of genes in their proposed fused orientation. The original reads are aligned to these candidate fusion contigs; fusion-supporting reads that would normally align as discordant pairs or split reads should align as concordant 'normal' reads in this fusion-gene context. Those reads supporting each fusion (spanning fragments and fusion-breakpoint-containing reads) are identified, reported, and scored accordingly*." [68] An illustration of the method can be seen in Figure 5.1.

The evidence for fusions as evaluated by FusionInspector is easily viewed and navigated via html-based fusion reports included as output.

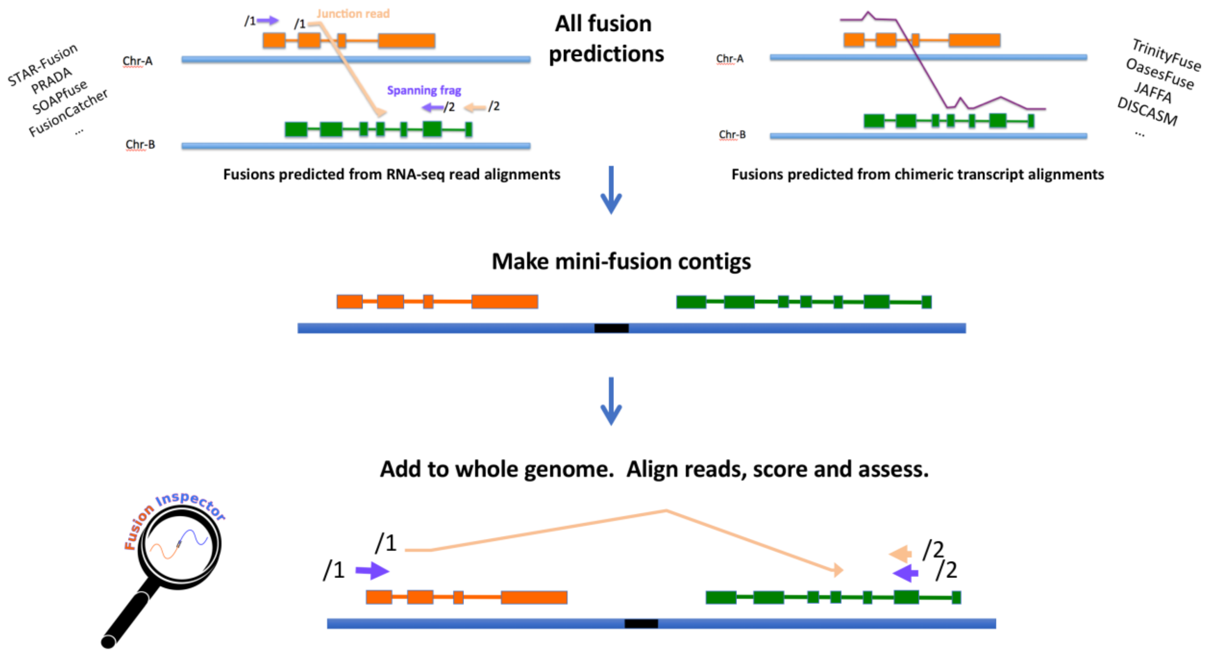


Figure 5.1: An overview of the FusionInspector process [68]

6 Design and implementation

6.1 Workflow

The main aim of the pipeline was to find gene fusions candidates in raw FASTQ data and return them in a table together with a visualisation of the results. As mentioned above, such an analysis can be complex, but there are four main steps (see Figure 6.1):

1. **Data pre-processing** takes the input data in compressed FASTQ format and processes them as described in Section 3.2.1 Primary analysis. Some of the tools used in the subsequent steps also perform their own pre-processing, however, we aimed to ensure that each of them receives the data pre-processed in the same manner.
2. **Fusion genes detection** takes the pre-processed data and returns a list of gene fusion candidates with their parameters. In this step we include multiple tools, where each comes with their own algorithm; nevertheless, they all perform alignment to the reference genome or transcriptome at some stage and these files need to be provided.
3. **Result validation** provides a level of validation for the evidence of predictions. It takes a list of gene fusions predicted in the previous step and aims to discover them in the reference genome.
4. **Results post-processing** collects the results and unites them into one table provided in the output. In addition, we utilise multiple tools that include functionality for visualisation.

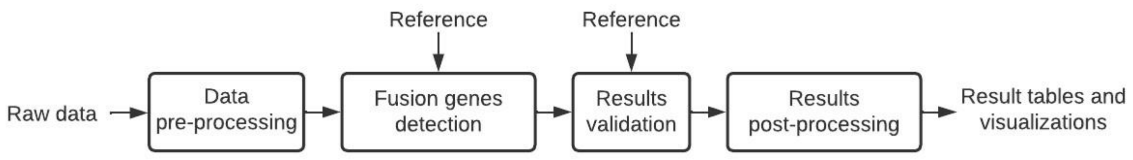


Figure 6.1: Simple workflow of the proposed solution

Below, we illustrate the integration of each tool in the workflow (see Figure 6.2):

1. The pre-processing step includes Fastp, which takes the input compressed FASTQ data and outputs them pre-processed to the subsequent processes in the same format. Furthermore, it returns an HTML report with visualisation of the result.
2. Fusion genes detection includes multiple tools: STAR-Fusion, Arriba, Cicero, FusionCatcher, and JAFFA, where each needs to be provided with its own

reference files. Some of them use the same aligner (STAR in this case) and also provide the option to skip the alignment stage and take already aligned data as input. We take advantage of that option and run STAR only once, instead of multiple times. Therefore:

- STAR-Fusion and Arriba take output from STAR, skipping their in-built aligning phase,
- Cicero offers to take already aligned data; however, it is required to be sorted and indexed, which is managed by Samtools,
- FusionCatcher and JAFFA do not provide the option to separate the alignment, thus they take the pre-processed data directly from Fastp.

Each fusion genes detection tool outputs a file in either TSV, CSV, or TXT format, containing a list of fusion genes candidates and their attributes. These files are collected and taken by both the validation and post-processing step.

3. The validation stage incorporates FusionInspector, which requires predicted fusion gene names that are to be analysed, along with the FASTQ data (from Fastp) and its reference. Apart from a HTML report containing a visual representation of the result, it provides a TSV file with assessed fusion genes candidates in a similar format as previous fusion gene callers. This file is added to the collection from the previous step and passed to the post-processing step.
4. The post-processing stage includes two steps. First, the outputs of all the tools from the fusion genes detection and the result of FusionInspector are collected and combined into one comprehensive table. Second, the result of the validation step is passed to Arriba, which provides a useful script for visualising the predicted fusion genes and is customised to take a file formed by FusionInspector.

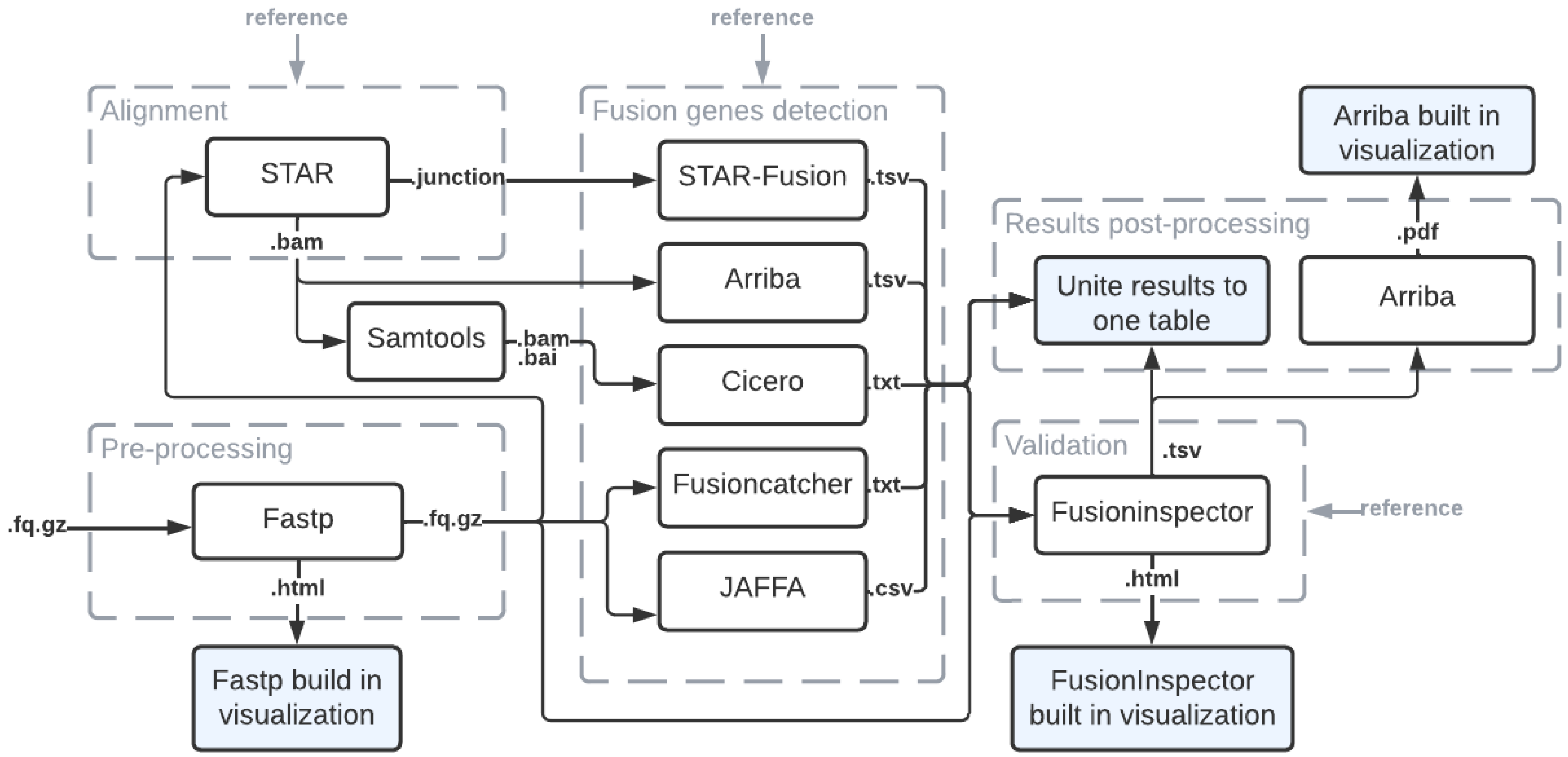


Figure 6.2: Bioinformatic tools and their integration in the workflow. The dashed boxes represent the steps from previous simple workflow, white boxes are tools, the functionality of which is utilised for the given processes (all the included tools were described in detail in Section 5.5 Data processing tools), and blue boxes show the output generating processes.

6.2 Dockerization

We created Docker images for all the tools mentioned above, including Fastp, STAR, Samtools, JAFFA, FusionCatcher, Arriba, STAR-Fusion, Cicero and FusionInspector. All the images were built based on the original Dockerfiles or Images provided by the developers, with few following exceptions:

- The Dockerfile for Arriba was adjusted to replace the run_arriba.sh script with our modified one, which skips the STAR step and takes a BAM file on the input, instead of FASTQ files. The modified script can be found in the project under 'docker/arriba/run_arriba.sh'.
- Fastp does not offer a Dockerfile, hence we created it ourselves.
- The Cicero image was built from their official image with overwritten `ENTRYPOINT` as the only change.

For Docker to be set to `enable` in the Nextflow configuration file, a Docker image is expected for every process of the Nextflow pipeline. Therefore, we also created other images for the rest of the processes which do not include mentioned tools - e.g. an image with Python and required packages to run a process containing only a Python script.

Each of the used Dockerfiles is available in the project in the `docker` directory. The list of images made available through Docker Hub is shown in Table 6.1.

Table 6.1: A list of Docker images available through the Docker Hub

Docker image	Content	Nextflow process
souckmi2/fastp:1.0	Fastp 0.23.2	fastp
souckmi2/star:1.0	STAR 2.7.10b	star
souckmi2/samtools:1.0	Samtools 1.17	samtools
souckmi2/jaffa:1.0	JAFFA 2.3	jaffa
souckmi2/fusioncatcher:1.0	FusionCatcher v1.30	fusioncatcher
souckmi2/arriba:1.0	Arriba v2.4.0	arriba, draw_fusions
souckmi2/starfusion:1.0	STAR-Fusion 1.11.1	starfusion
souckmi2/fusioninspector:1.0	FusionInspector 2.8.0	fusioninspector
souckmi2/cicero:1.0	Cicero v1.9.5	cicero
souckmi2/py-numpy-pandas:1.0	Python 3.9 with numpy, pandas and xlsxwriter	merge_input_files, fusion_tables, final_result_table

6.3 Input files

To run the pipeline, a user needs to provide multiple files listed below:

1. **Reads files**, containing the paired end sequencing data to be analysed, where the only supported extension is .fastq.gz. File naming is required to match one of the following patterns:
   - If there are only two files (read 1 and read 2) per sample, they need to match the pattern '*_R1,_R2.fastq.gz', for example, 'p1_R1.fastq.gz'.
   - In case there are multiple files from different lanes per one sample, the files are expected to have the typical Illumina naming convention matching the regular expression '^(.+_S[0-9]+)+(_.+)*_R([1-2])_', for example 'p4_S4_L001_R1_001.fastq.gz'

   It is possible to provide reads for multiple samples and the analysis will be conducted on each of them parallelly, exporting results to separate directories for each sample.
2. **Reference files** for each of the tools that request them. We provide an already prepared directory containing all the required files (available at https://owncloud.cesnet.cz/index.php/s/XVovHksT8m1hIMa), however, it is possible to provide a custom directory as long as it follows the same directory structure.

6.4 Nextflow implementation

In practise, a Nextflow pipeline script is made by joining together different **processes**. Each process can be written in any scripting language that can be executed by the Linux platform. Processes communicate with each other via queues called **channels**, where any process can define one or more channels as **input** and **output**.

We developed the pipeline using DSL2, which provides a syntax extension that allows the definition of **module** libraries and **sub-workflows**. Module files are scripts that can be included and shared across workflow pipelines.

The **configuration file** (nextflow.config) contains the settings that are read when the pipeline is launched. The file provides the ability to separate the workflow implementation from the configuration setting required by the underlying execution platform. This enables portable deployment without the need to modify the application code.

6.4.1 Input parameters

There are several parameters, which the user is required to provide when running the pipeline:

- **readsdir** (required) – A path to the directory containing the FASTQ files to be analysed, which need to file certain naming conventions as mentioned in Section 6.3 Input files.
- **mergeInputFiles** (default False) – If the reads files are from different lines and need to be merged (as described in Section 6.2 Input files), this parameter must be set to True.
- **reference** (required) – A path to the directory containing the reference files requested by the tools used in the pipeline.
- **outdir** (required) – A path where the result output of the analysis should be stored.
- **runId** (required) – A run id which will be added to the name of output files.

If the 'metacentrum' profile is used (see Section 6.4.4 Configuration), additional configuration parameters need to be provided (for more details, see GitLab README):

- **k8s_namespace** (required) – Kubernetes namespace.
- **k8s_storageclaimname** (required) – Storage claim name.
- **k8s_storagemountpath** (required) – Storage mount path.
- **k8s_launchdir** (required) – Path to a launch directory.
- **k8s_workdir** (required) – Path to a work directory.

6.4.2 Nextflow workflows and modules

We leverage the ability of Nextflow to create sub-workflows and modules, which makes the implementation of complex pipelines easier and clearer.

There are two modules added to the implementation. The **Fusions Module** contains the processes and a sub-workflow specific to fusion gene detection. The **Helpers Module** then includes processes that are to be reused by any part of the pipeline in any stage (see Table 6.2). The main workflow in the main.nf script is then able to include the modules, and we can construct the complete workflow of the whole analysis.

With this modular approach, we leave space for an eventual extension of the pipeline with other distinct steps of the analysis. For example, if the detection of fusion genes was not the only goal, but we wanted to add RNAseq differential expression analysis conducted on the same data, we could easily add another module, keeping the structure of the project intact.

Table 6.2: Nextflow modules

Module	Processes	Workflow
Fusions Module	fastp, star, samtools, jaffa, fusioncatcher, arriba, starfusion, cicero, fusion_tables, fusioninspector, draw_fusions, final_result_table	fusion_detection
Helpers Module	merge_input_files	-

6.4.3 Processes and channels

Each step of the analysis is represented by a process, a basic processing primitive to execute a script. Each process includes a definition block for **input**, **output**, and a **process script**. The process script can be written in any language that can be executed on the Linux platform. We included scripts in Bash to run the bioinformatic tools and Python for managing the files content if needed.

Figure 6.3 visualises a direct acyclic graph (DAG) of the out pipeline generated by Nextflow. The vertices of the graph represent the pipeline processes and operators, while the edges represent the data connections (i.e. channels) between them. Arrows pointing to small dots picture the files produced by the process that are copied to the output directory.

There are 13 processes defined. Each of the 10 bioinformatic tools mentioned above has one process that handles only the task for which the tool is included in the analysis. In addition, there is a second process with Arriba, draw_fusions, which calls the R script provided by Arriba for fusion gene visualisation. The remaining three processes, described below, include Python scripts.

Merge_input_files process

Given a list of FASTQ files, it merges, for each sample, all files from different lanes, so that on the output there is only one R1 and R2 file per sample. The Python script is based on a script from the SciLifeLab GitHub repository [69]. This process is executed only if the '--mergeInputFiles' parameter is set to true and the provided FASTQ files follow the naming convention described in Section 6.3 Input files.

Fusions_table process

FusionInspector expects the input to be a text file (or files) with a list of fusion candidates, with each formatted as geneA--geneB. Therefore, this process takes the results from the fusion callers and transforms them to suit the requirements.

Final_result_table process

This process handles the generation of the final result table. It collects the results of the fusion gene detection tools along with the output of FusionInspector and unites them into one comprehensive table, which is exported as a CSV file and a formatted XLXS file. The table columns and formatting as well as the relations between the final table and callers results are described in Section 6.5.3 Final_result_table.xlsx.

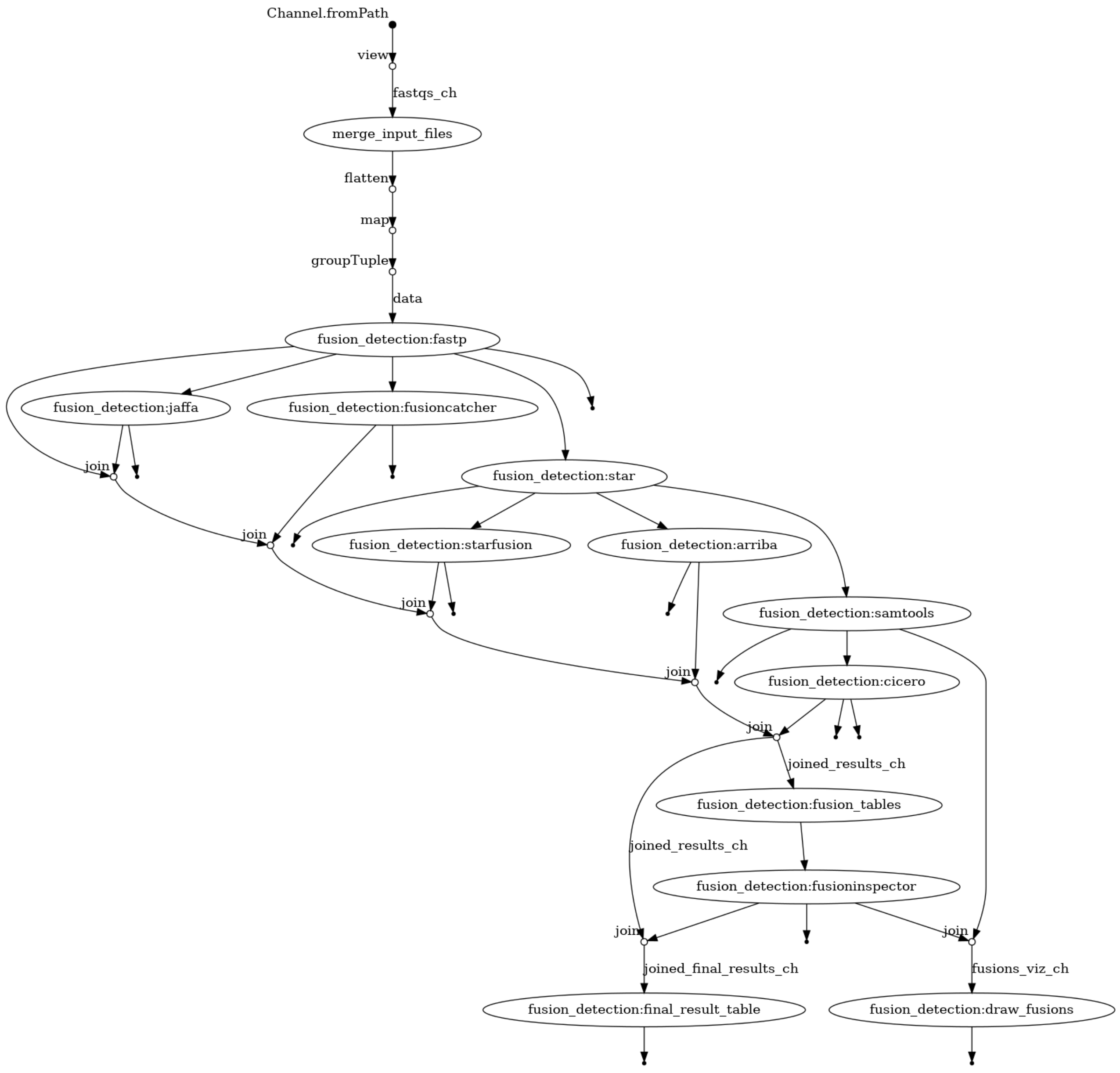


Figure 6.3: DAG visualisation of the pipeline generated by Nextflow

6.4.4 Configuration

The configuration file allows to manage the settings of various scopes, from the pipeline input parameters to docker configurations or deployment settings.

Profiles

One of the Nextflow features is the config **profile**. A profile is a set of configuration attributes that can be selected during pipeline execution by using the '-profile' command line option, providing an option to switch between executors and change deployment parameters without the need to edit any files. We provide two profiles (separated into two files in the config_profiles directory):

1. The '**standard**' profile is a default profile that is selected if no profile is defined by the user. The 'executor' is set to 'local', that is, it runs the pipeline processes on the computer where Nextflow is launched.
2. The '**metacentrum**' profile is designed to configure pipeline execution on a Metacentrum Kubernetes server. Unlike the standard profile, the 'executor' is set to 'k8s', and the configuration includes the k8s scope. If this profile is selected, the user needs to provide additional parameters (see Section 6.4.1 Input parameters).

Resources Management

Nextflow provides an option to set the resources (CPU and memory) to be reserved for each pipeline process. We set slightly different values for each profile, see Table 6.3. If the 'metacentrum' profile is selected, at least 20 CPU and 100 GB of RAM are expected, ideally around 40 CPU and 180 GB of RAM to allow processes to run parallel. For the 'standard' profile, 1 CPU and 45 GB of RAM are required, ideally then at least 65 GB of RAM.

Table 6.3: Resources allocated for each process

	Standard profile		Metacentrum profile	
Process	CPU	RAM (GB)	CPU	RAM (GB)
fastp	1	16	10	20
star	1	45	10	80
samtools	1	2	5	10
jaffa	1	8	10	32
fusioncatcher	1	10	20	100
arriba	1	8	10	20
starfusion	1	8	5	16
cicero	1	8	20	70
fusioninspector	1	45	20	80

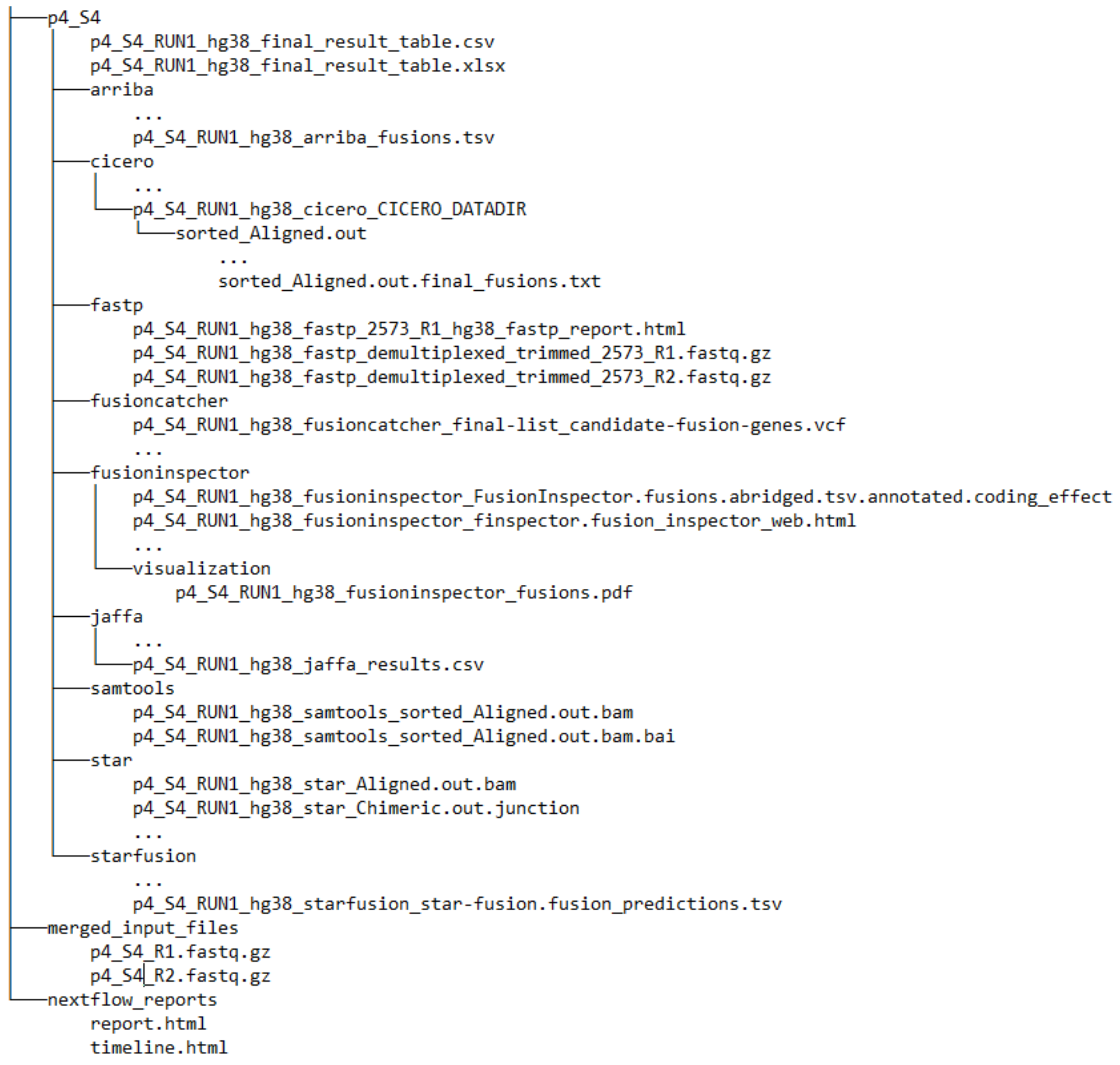
6.5 Output files

The main aim of our project was to build a pipeline which outputs a file with a table of detected fusion genes candidates, along with visualisation of the results. However, we also include most of the files produced by the bioinformatic tools.

6.5.1 Naming convention

Every file produced by a process is renamed so that it follows the pattern '<sample_name>_<run_id>_<reference_version>_<caller>_<original_file_name>'.

Therefore, for example, if the input FASTQ file carries the name 'p1_S4_L001_R1_001.fastq.gz' and the run id provided by the user is 'RUN1', a file produced by Arriba originally named 'fusions.tsv' would be named 'p1_S4_RUN1_hg38_arriba_fusions.tsv'.



```
├───p4_S4
│   │   p4_S4_RUN1_hg38_final_result_table.csv
│   │   p4_S4_RUN1_hg38_final_result_table.xlsx
│   ├───arriba
│   │       ...
│   │       p4_S4_RUN1_hg38_arriba_fusions.tsv
│   ├───cicero
│   │   │   ...
│   │   └───p4_S4_RUN1_hg38_cicero_CICERO_DATADIR
│   │       └───sorted_Aligned.out
│   │               ...
│   │               sorted_Aligned.out.final_fusions.txt
│   ├───fastp
│   │       p4_S4_RUN1_hg38_fastp_2573_R1_hg38_fastp_report.html
│   │       p4_S4_RUN1_hg38_fastp_demultiplexed_trimmed_2573_R1.fastq.gz
│   │       p4_S4_RUN1_hg38_fastp_demultiplexed_trimmed_2573_R2.fastq.gz
│   ├───fusioncatcher
│   │       p4_S4_RUN1_hg38_fusioncatcher_final-list_candidate-fusion-genes.vcf
│   │       ...
│   ├───fusioninspector
│   │   │   p4_S4_RUN1_hg38_fusioninspector_FusionInspector.fusions.abridged.tsv.annotated.coding_effect
│   │   │   p4_S4_RUN1_hg38_fusioninspector_finspector.fusion_inspector_web.html
│   │   │   ...
│   │   └───visualization
│   │           p4_S4_RUN1_hg38_fusioninspector_fusions.pdf
│   ├───jaffa
│   │   │   ...
│   │   └───p4_S4_RUN1_hg38_jaffa_results.csv
│   ├───samtools
│   │       p4_S4_RUN1_hg38_samtools_sorted_Aligned.out.bam
│   │       p4_S4_RUN1_hg38_samtools_sorted_Aligned.out.bam.bai
│   ├───star
│   │       p4_S4_RUN1_hg38_star_Aligned.out.bam
│   │       p4_S4_RUN1_hg38_star_Chimeric.out.junction
│   │       ...
│   └───starfusion
│           ...
│           p4_S4_RUN1_hg38_starfusion_star-fusion.fusion_predictions.tsv
├───merged_input_files
│       p4_S4_R1.fastq.gz
│       p4_S4_R2.fastq.gz
└───nextflow_reports
        report.html
        timeline.html
```

Figure 6.4: Output directory tree with key result files

6.5.2 Directory tree

The output directory with the path defined by the user has the following directory tree (Figure 6.4), showing the files that are the key result of the pipeline: final result table, visualisations, result tables for fusion callers, and processed data files for Fastp and STAR. There are three folders:

- **Sample** folder – The folder is named after the name of the sample according to the input FASTQ files. There are two files, in CSV and XLSX format, that provide the final result of the analysis together with one directory for each bioinformatic tool. These directories contain all the files produced by the tools.
- **Merged_input_files** folder – If the '--mergeInputFiles' parameter is set to true, this folder contains the merged FASTQ files, otherwise it is not created.
- **Nextflow_reports** folder – The folder contains the Nextflow generated reports describing the pipeline execution.

6.5.3 Final_result_table.xlsx

This file is the main result of the analysis. It contains a list of detected fusion genes candidates and is intended to be forwarded to a clinical expert. The excel table unites the results of all the fusion genes callers and FusionInspector, as shown in Table 6.7. The columns of the result table are briefly described below in Table 6.4 (we grouped the values that are comparable, but the exact definitions can differ depending on the caller, therefore, for more information, see the documentation of the callers output files).

The cells are formatted according to the rules in Table 6.5, with the relevance to cancer specified in Table 6.6.

Table 6.4: Description of the columns of the final result table

Column	Description
caller	The bioinformatic tool whose results are shown in the row
gene1	Gene symbol of the 5' end
gene2	Gene symbol of the 3' end
chrom1/position1	Chromosomal position of the breakpoint in gene1
chrom2/position2	Chromosomal position of the breakpoint in gene2
spanning_reads	The number of reads which cover the breakpoint
spanning_pairs	The number of read-pairs, where each read in the pair aligns entirely on either side of the breakpoint
coverage1/coverage2	The coverage near the breakpoint in gene1 and gene2
annotation	A simplified annotation for fusion transcript, depending on the caller. It mostly contains database(s) in which the fusion was found
confidence	Suggests the credibility of predictions
prediction_effect	Provides information on the location of the breakpoints. It indicates whether the proposed breakpoint occurs at reference exon junctions
anchor	Indicates whether there are split reads that provide 'long' alignments on both sides of the putative breakpoint
FAR_left	Left fusion allelic ratio (only for FusionInspector)
FAR_right	Right fusion allelic ratio (only for FusionInspector)
FFPM	Normalised measure of the quantity of RNAseq fragments supporting the fusion event as: fusion fragments per total million RNAseq fragments (only for FusionInspector)

Table 6.5: Formatting rules of the final result table

Rule	Format
The spanning reads value is less than 5	Red cell
The coverage1 is greater than 50	Red cell
The coverage2 is greater than 50	Red cell
The value in column chrom1 and chrom2 are equal	Both cells yellow
The confidence column suggests low confidence (i.e. contains ‘LowConfidence’, ‘LQ’ or ‘low’)	Red row
The prediction effect contains ‘intragenic’, ‘intron’, ‘out-of-frame’ or ‘INCL_NON_REF_SPLICE’	Red row
The annotation column includes one of the sources that suggest that the fusion is relevant to cancer biology*	Green row
The annotation column includes one of the sources that suggest that the fusion pair may not be relevant to cancer, and be potential false positive*	Red row

* The sources are named in Table 6.6

Table 6.6: Resources in annotation column of the final result table and their relevance. They were selected based on CTAT_HumanFusionLib [70].

Relevance to cancer	Sources
Relevant to cancer biology	Mitelman, chimerdb_omim, chimerdb_pubmed, ChimerKB, ChimerPub, ChimerSeq, Cosmic, YOSHIHARA_TCGA, Klijn_CellLines, Larsson_TCGA, CCLE, HaasMedCancer, GUO2018CR_TCGA, TumorFusionsNAR2018, TCGA_StarF2019, CCLE_StarF2019, DEEPEST2019, ‘Yes’ (represents Mitelman database, according to Jaffa)
May not be relevant to cancer, and be potential false positive	GTEx_recurrent_StarF2019, BodyMap, DGD_PARALOGS, HGNC_GENEFAM, Greger_Normal, Babiceanu_Normal, ConjoinG

Table 6.7: The relation between the columns of the new final result table and the columns of callers output files

Result table	Arriba	STAR-Fusion	Jaffa	FusionCatcher	Cicero	FusionInspector
gene1	gene1	FusionName	fusion genes	Gene_1_symbol (5end_fusion_partner)	geneA	FusionName
gene2	gene2			Gene_2_symbol (3end_fusion_partner)	geneB	
chrom1	breakpoint1	LeftBreakpoint	chrom1	Fusion_point_for_gene_1 (5end_fusion_partner)	chrA	LeftBreakpoint
position1			base1		posA	
chrom2	breakpoint2	RightBreakpoint	chrom2	Fusion_point_for_gene_2 (3end_fusion_partner)	chrB	RightBreakpoint
position2			base2		posB	
spanning_reads	split_reads1, split_reads2	SpanningFragCount	spanning reads	-	readsA, readsB	SpanningFragCount
spanning_pairs	discordant_mates	JunctionReadCount	spanning pairs	Spanning_unique_reads	matchA, matchB	JunctionReadCount
coverage1	coverage1	-	-	-	coverageA	-
coverage2	coverage2	-	-	-	coverageB	-
annotation	tags	annots	known	Fusion_description	-	annots
confidence	confidence	-	classification	-	rating	-
prediction_effect	site1, site2	SpliceType	Inframe, aligns	Predicted_effect	featureA, featureB	SpliceType
anchor	-	LargeAnchorSupport	-	-	-	LargeAnchorSupport
FAR_left	-	-	-	-	-	FAR_left
FAR_right	-	-	-	-	-	FAR_right
FFPM	-	-	-	-	-	FFPM

6.5.4 Visualisation

In the output directory there are four files providing html or pdf-based reports – one summarising the pre-processing phase, two for fusion genes detection results and a Nextflow workflow execution report.

Pre-processing report

Fastp offers to visualise quality control and filtering results on a single HTML page. The file can be found in the output directory with path '<output path>/<sample name>/fastp/* fastp_report.html'.

The report includes a summary table, the sequence distribution of trimmed adapters, insert size distribution plot, and plots displaying quality/base content/KMER counting before and after filtering for both reads separately.

Fusion Genes Visualisation

We offer two reports visualizing the results of FusionInspector. FusionInspector offers to generate an HTML summary file, which can be found at '<ouput path>/<sample name>/fusioninspector/*fusion_inspector_web.html'. In the initial view, it provides a table of fusions and attributes. After selecting a fusion, it creates a tab that shows the evidence for the visualisation in a web-based igv view.

The second report is created by Arriba, which provides a script that renders visualisations of the transcripts involved in predicted fusions and is capable of taking the FusionInspector result file as input. It generates a PDF file with one page for each predicted fusion. Each page depicts the fusion partners, their orientation, the retained exons in the fusion transcript, and statistics about the number of supporting reads. The file is found at '<output path>/<sample name>/fusioninspector/visualization/*fusions.pdf'.

Nextflow execution report

Nextflow can create an HTML execution report, a single document that includes many metrics about a workflow execution. The report is organised into three main sections: summary, resources, and tasks.

The summary section reports the execution status, the launch command, overall execution time, and other workflow metadata. The resources section plots the distribution of resource usage for each workflow process, including CPU, memory, job duration, and disk I/O. The Tasks section lists all executed tasks, reporting for each of them the status, the actual command script, and other metrics.

7 Results

The Nextflow pipeline is available at GitLab as a public repository at https://gitlab.com/souckmi/gene-fusion-pipeline, thus it can be run on any device with the internet connection and Nextflow installed, as briefly described below or as explained in the GitLab repository README.

To run the analysis, the reference files for each caller must be downloaded from https://owncloud.cesnet.cz/index.php/s/XVovHksT8m1hIMa. For testing purposes, we also included a small dataset provided by FusionCatcher, which is available in the GitLab repository in the test/data directory. In the next sections, we represent a real data running example.

7.1 Testing data

Our solution was tested with two anonymised datasets, which were provided by The Institute of Hematology and Blood Transfusion with the patients' written informed consent for research. Additionally, we include a smaller dataset provided by FusionCatcher, that can be found in GitLab repository. The dataset are described in Table 7.1.

Table 7.1: Real testing data

Sample name	Multiple lines	Size	Description
p1_S4	Yes	1,49 GB	A male patient diagnosed at 25 years of age, diagnostical sample, ph+ ALL, 0,1% BCR::ABL1 MR
p2_S10	Yes	3,91 GB	A male patient diagnosed with CML at 74 years of age, sample taken 7 months after diagnoses, 4% BCR::ABL1 in IS
test_data	No	1,50 MB	Minimalistic testing samples provided by FusionCatcher

7.2 Real data running example

As mentioned above, the pipeline can be run either locally or with Metacentrum. Here, we provide an example of running the analysis with the ‘metacentrum’ profile, i.e. with Kubernetes executor (more detailed documentation is available for both profiles in GitLab README).

With Nextflow installed and reference files prepared, the pipeline is executed with an example command in Code7.1. It is possible to replace the double dashed parameters with a parameter "-params-file" and path to a JSON or YAML file that contains them.

```
nextflow kuberun https://gitlab.com/souckmi/gene-fusion-pipeline \
         -r <revision> \
         -profile metacentrum
         -pod-image 'cerit.io/nextflow/nextflow:22.06.1'\
         -v <storage claim name>:/mnt\
         --readsdir /path/to/data/directory \
         --runId <run id>
         --mergeInputFiles true \
         --reference /path/to/reference/directory \
         --outdir /path/to/output/directory \
         --k8s_namespace <k8s name space> \
         --k8s_storageclaimname <storage claim name> \
         --k8s_storagemountpath /mnt \
         --k8s_launchdir /mnt/path/to/launchdir \
         --k8s_workdir /mnt/path/to/workDir \
```

Code 7.1: Example command line command to run the pipeline with MetaCentrum

7.2.1 Execution time and resource management

The duration of the execution is dependent on the size and content of the input data and the computational resources provided – if only the minimal required resources are available, the processes cannot run parallelly, and the analysis runs longer.

Table 7.2: Time duration of execution for different input files

Sample	p1_S4	p2_S10	p3_S4 + p2_S10
Duration	2h 2m 52s	3h 39m 52s	4h 24m 53s

We tested the pipeline with 252 GB of RAM and 48 CPUs available. Below we show the percentage of usage of allocated resources for samples p2_S10, see Figure 7.2. The Figure 7.1 illustrates the timeline and time duration for executing the pipeline for samples p1_S4 and p2_S10 separately, Figure 7.3 then with both samples p1_S4 and p2_S10 on the input. The comparison of execution duration for samples of varied sizes can be seen in Table 7.2.

a:

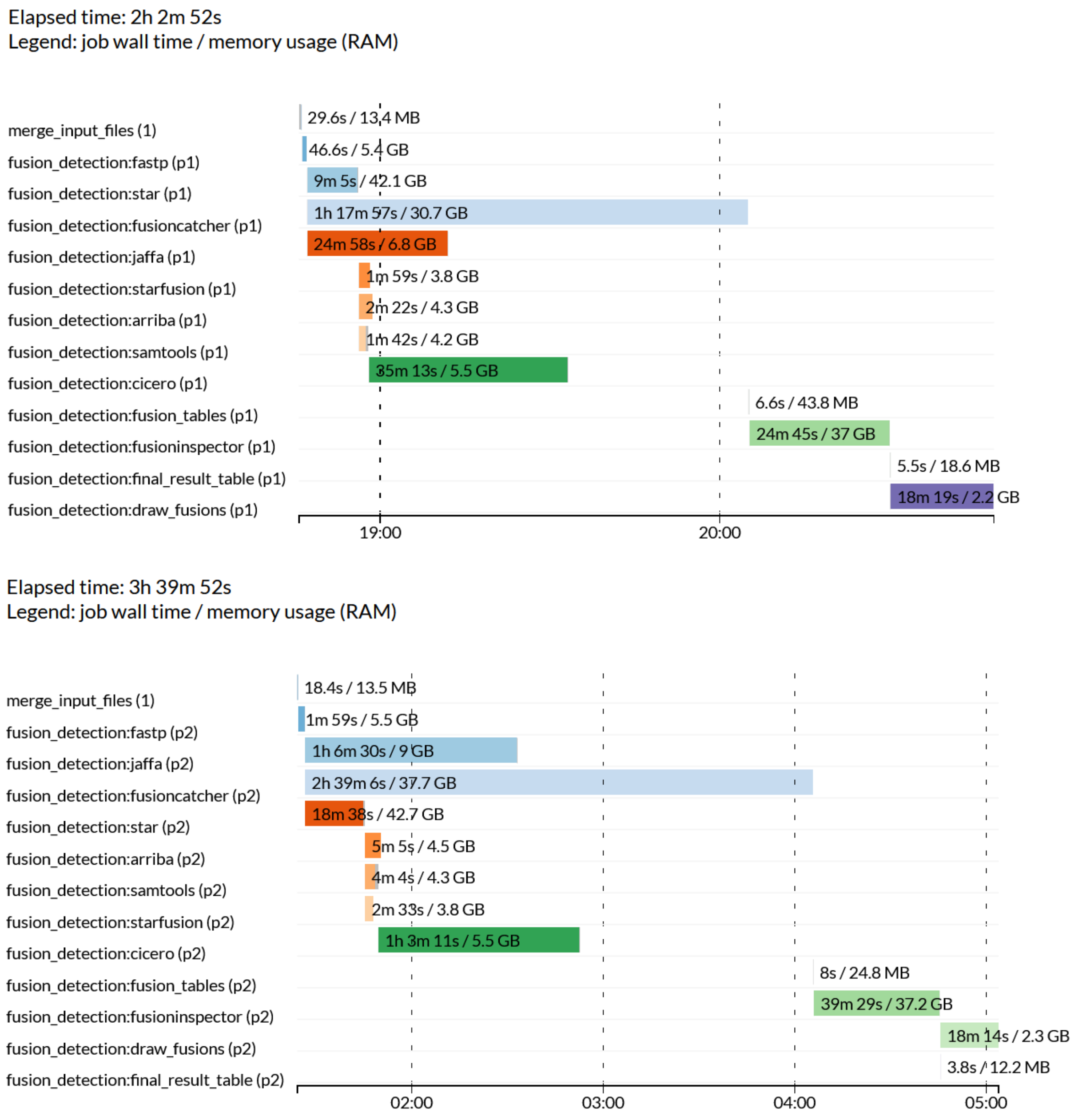
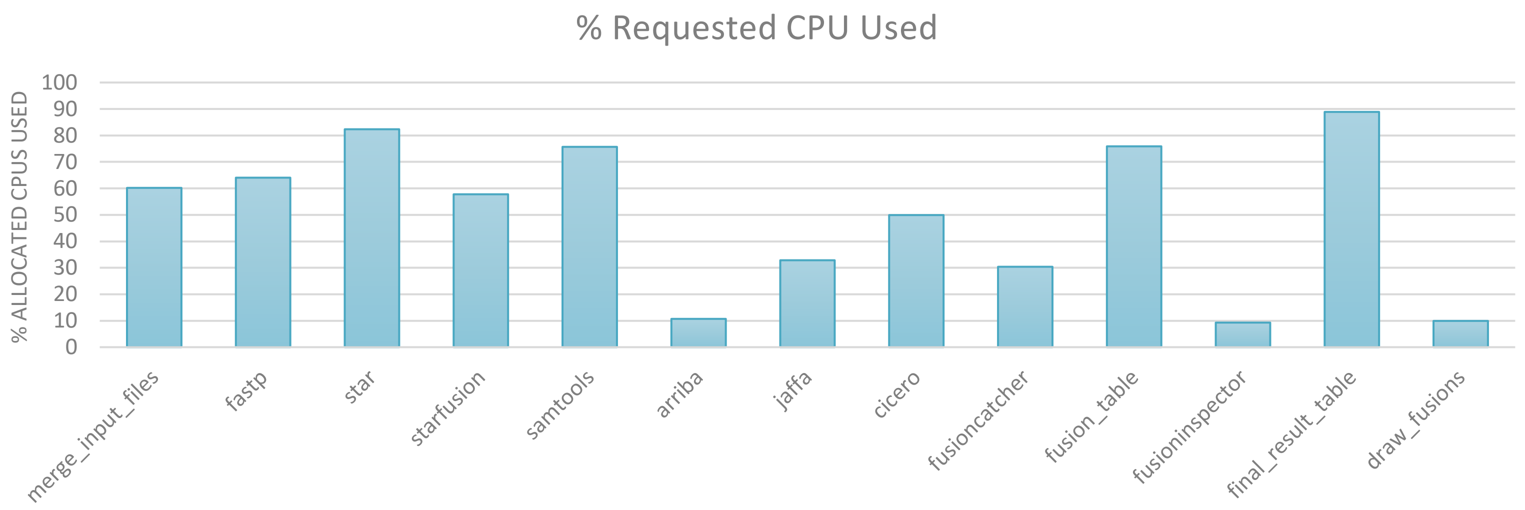


Figure 7.1: Execution timeline with wall time and memory usage for each process. a: Sample p1_S4. **b:** Sample p2_S10.



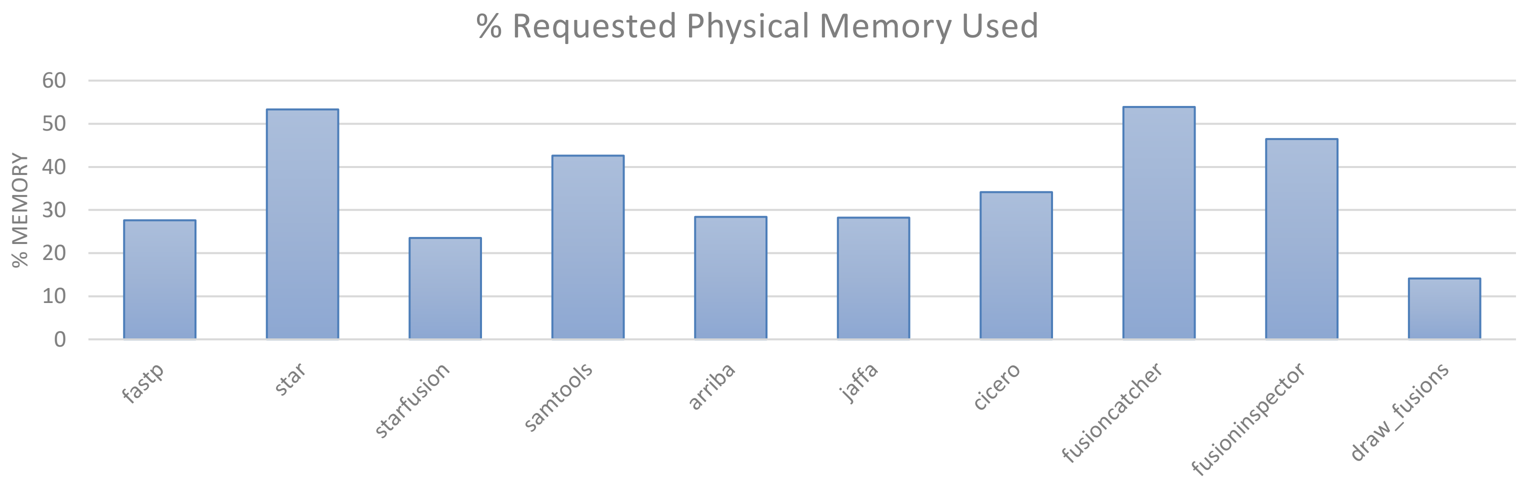


Figure 7.2: Percentage of requested RAM and CPU compared to the allocated values

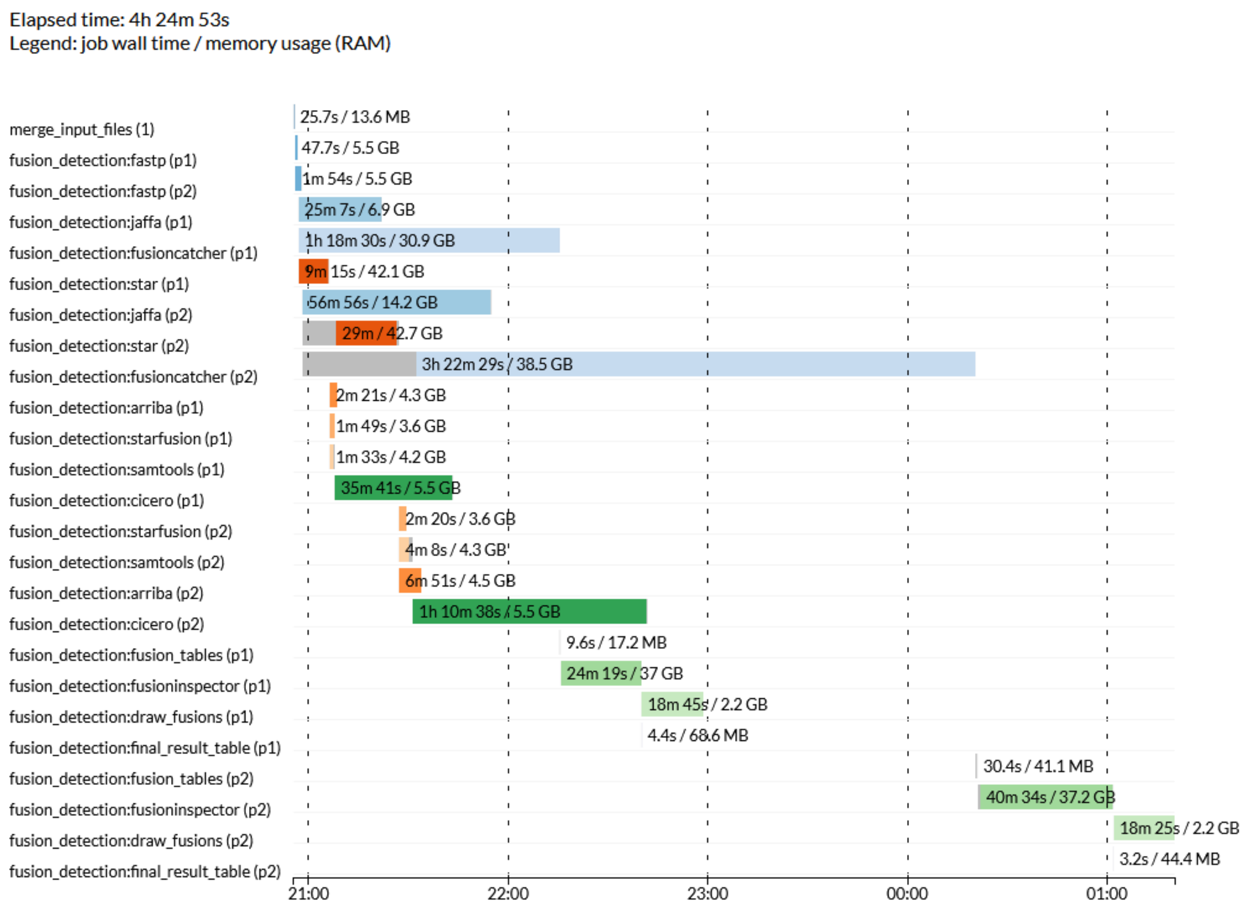
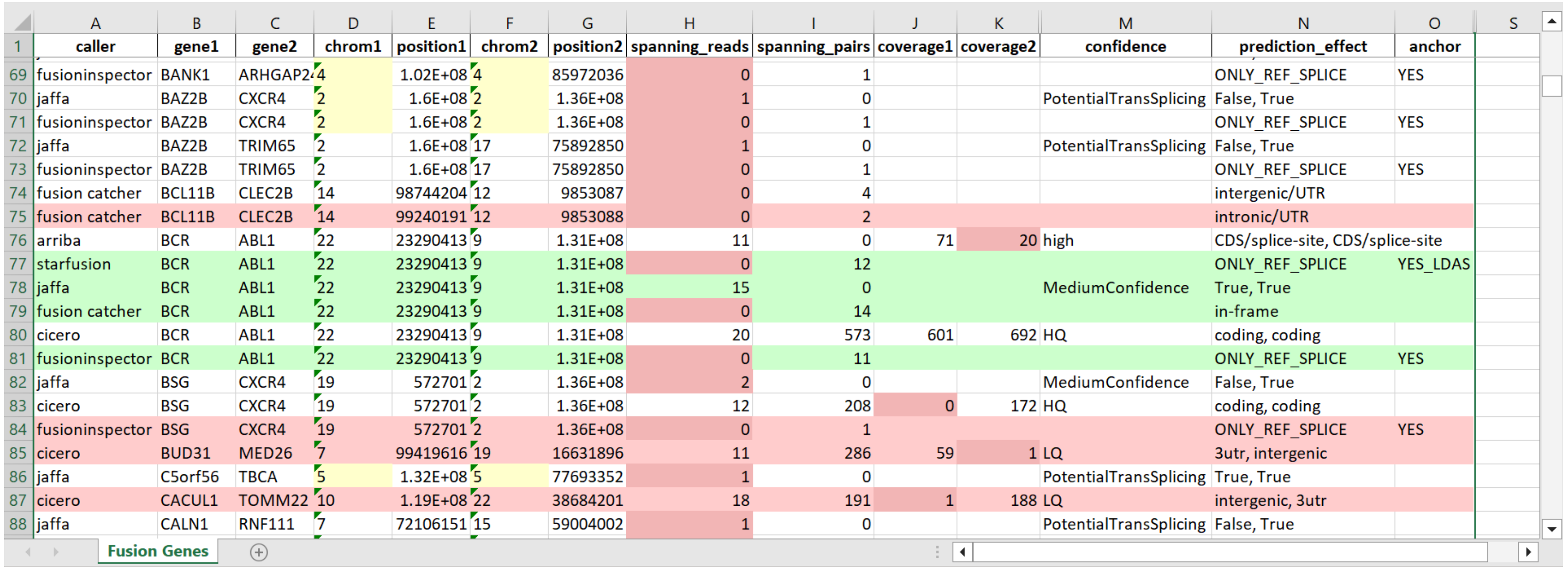


Figure 7.3: Execution timeline for analysis with both testing samples on the input. A grey segment of a bar represents submitted state, colourful then running state of a process.

7.2.2 Output files

Below we present the four main output files, where all the following figures are results of the p1_S4 sample analysis. As mentioned, the pipeline generates an XLSX file containing a formatted table of fusion gene candidates for each caller; an example can be seen in Figure 7.4, where the columns ‘anot’, ‘FAR_right’, ‘FAR_left’, and ‘FFPM’ are omitted.

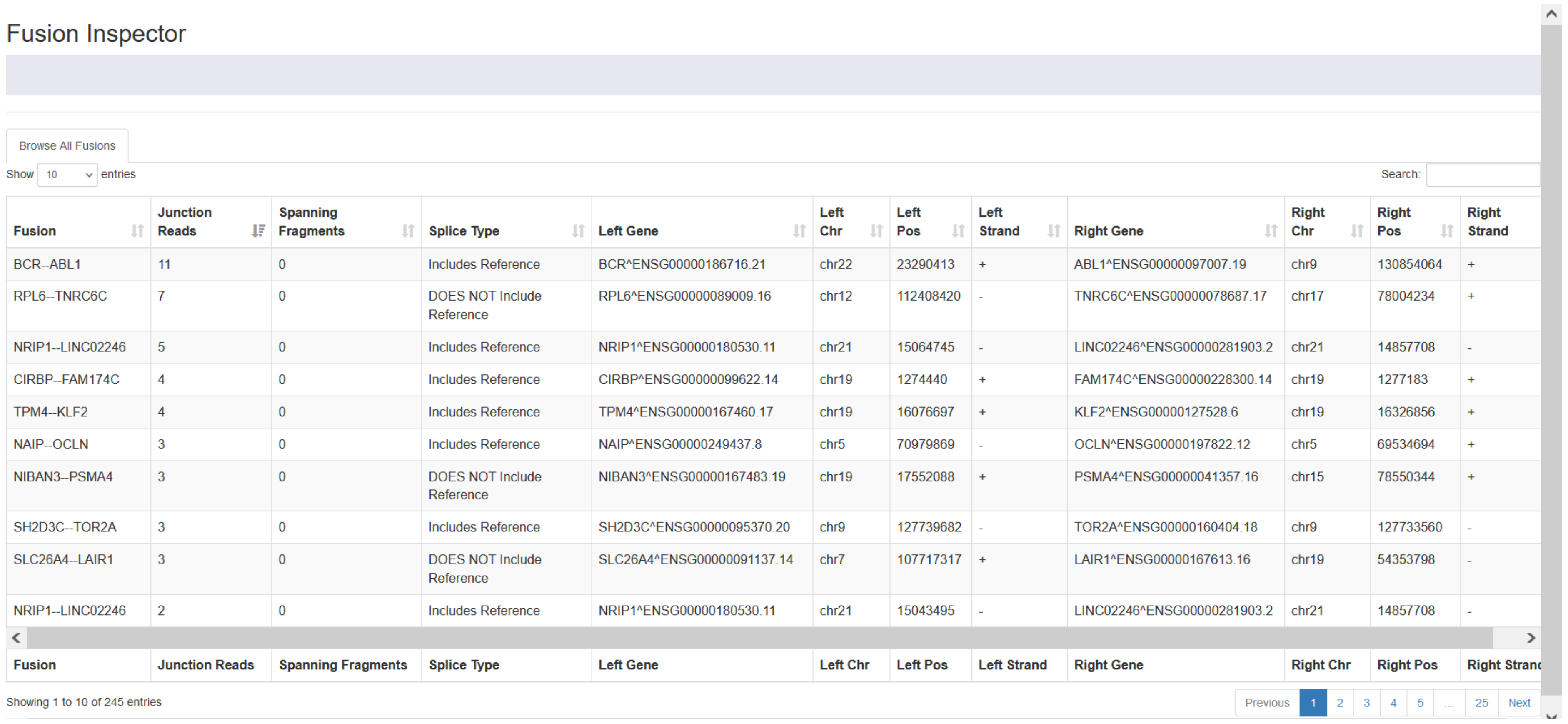
Figures 7.5 and 7.6 show the FusionInspector visualisation, fusion_inspector_web.htm, and Figure 7.7 shows an example of a FusionInspector gene fusion candidate visualised by Arriba in fusions.pdf. Two plots from the summary report generated by Fastp can be seen in Figure 7.8.



	A	B	C	D	E	F	G	H	I	J	K	M	N	O	S
1	caller	gene1	gene2	chrom1	position1	chrom2	position2	spanning_reads	spanning_pairs	coverage1	coverage2	confidence	prediction_effect	anchor	
69	fusioninspector	BANK1	ARHGAP2	4	1.02E+08	4	85972036	0	1				ONLY_REF_SPLICE	YES	
70	jaffa	BAZ2B	CXCR4	2	1.6E+08	2	1.36E+08	1	0			PotentialTransSplicing	False, True		
71	fusioninspector	BAZ2B	CXCR4	2	1.6E+08	2	1.36E+08	0	1				ONLY_REF_SPLICE	YES	
72	jaffa	BAZ2B	TRIM65	2	1.6E+08	17	75892850	1	0			PotentialTransSplicing	False, True		
73	fusioninspector	BAZ2B	TRIM65	2	1.6E+08	17	75892850	0	1				ONLY_REF_SPLICE	YES	
74	fusion catcher	BCL11B	CLEC2B	14	98744204	12	9853087	0	4				intergenic/UTR		
75	fusion catcher	BCL11B	CLEC2B	14	99240191	12	9853088	0	2				intronic/UTR		
76	arriba	BCR	ABL1	22	23290413	9	1.31E+08	11	0	71	20	high	CDS/splice-site, CDS/splice-site		
77	starfusion	BCR	ABL1	22	23290413	9	1.31E+08	0	12				ONLY_REF_SPLICE	YES_LDAS	
78	jaffa	BCR	ABL1	22	23290413	9	1.31E+08	15	0			MediumConfidence	True, True		
79	fusion catcher	BCR	ABL1	22	23290413	9	1.31E+08	0	14				in-frame		
80	cicero	BCR	ABL1	22	23290413	9	1.31E+08	20	573	601	692	HQ	coding, coding		
81	fusioninspector	BCR	ABL1	22	23290413	9	1.31E+08	0	11				ONLY_REF_SPLICE	YES	
82	jaffa	BSG	CXCR4	19	572701	2	1.36E+08	2	0			MediumConfidence	False, True		
83	cicero	BSG	CXCR4	19	572701	2	1.36E+08	12	208	0	172	HQ	coding, coding		
84	fusioninspector	BSG	CXCR4	19	572701	2	1.36E+08	0	1				ONLY_REF_SPLICE	YES	
85	cicero	BUD31	MED26	7	99419616	19	16631896	11	286	59	1	LQ	3utr, intergenic		
86	jaffa	C5orf56	TBCA	5	1.32E+08	5	77693352	1	0			PotentialTransSplicing	True, True		
87	cicero	CACUL1	TOMM22	10	1.19E+08	22	38684201	18	191	1	188	LQ	intergenic, 3utr		
88	jaffa	CALN1	RNF111	7	72106151	15	59004002	1	0			PotentialTransSplicing	False, True		

Fusion Genes

Figure 7.4: The XLSX final result table with the detected BCR::ABL1 fusion gene (sample p1_S4)


Fusion Inspector

Browse All Fusions

Show 10 entries

Search:

Fusion	Junction Reads	Spanning Fragments	Splice Type	Left Gene	Left Chr	Left Pos	Left Strand	Right Gene	Right Chr	Right Pos	Right Strand
BCR--ABL1	11	0	Includes Reference	BCR^ENSG00000186716.21	chr22	23290413	+	ABL1^ENSG00000097007.19	chr9	130854064	+
RPL6--TNRC6C	7	0	DOES NOT Include Reference	RPL6^ENSG00000089009.16	chr12	112408420	-	TNRC6C^ENSG00000078687.17	chr17	78004234	+
NRIP1--LINC02246	5	0	Includes Reference	NRIP1^ENSG00000180530.11	chr21	15064745	-	LINC02246^ENSG00000281903.2	chr21	14857708	-
CIRBP--FAM174C	4	0	Includes Reference	CIRBP^ENSG00000099622.14	chr19	1274440	+	FAM174C^ENSG00000228300.14	chr19	1277183	+
TPM4--KLF2	4	0	Includes Reference	TPM4^ENSG00000167460.17	chr19	16076697	+	KLF2^ENSG00000127528.6	chr19	16326856	+
NAIP--OCLN	3	0	Includes Reference	NAIP^ENSG00000249437.8	chr5	70979869	-	OCLN^ENSG00000197822.12	chr5	69534694	+
NIBAN3--PSMA4	3	0	DOES NOT Include Reference	NIBAN3^ENSG00000167483.19	chr19	17552088	+	PSMA4^ENSG00000041357.16	chr15	78550344	+
SH2D3C--TOR2A	3	0	Includes Reference	SH2D3C^ENSG00000095370.20	chr9	127739682	-	TOR2A^ENSG00000160404.18	chr9	127733560	-
SLC26A4--LAIR1	3	0	DOES NOT Include Reference	SLC26A4^ENSG00000091137.14	chr7	107717317	+	LAIR1^ENSG00000167613.16	chr19	54353798	-
NRIP1--LINC02246	2	0	Includes Reference	NRIP1^ENSG00000180530.11	chr21	15043495	-	LINC02246^ENSG00000281903.2	chr21	14857708	-
Fusion	Junction Reads	Spanning Fragments	Splice Type	Left Gene	Left Chr	Left Pos	Left Strand	Right Gene	Right Chr	Right Pos	Right Stranc

Showing 1 to 10 of 245 entries

Previous 1 2 3 4 5 … 25 Next

Figure 7.5: The FusionInspector HTML summary visualisation (sample p1_S4). The initial view.

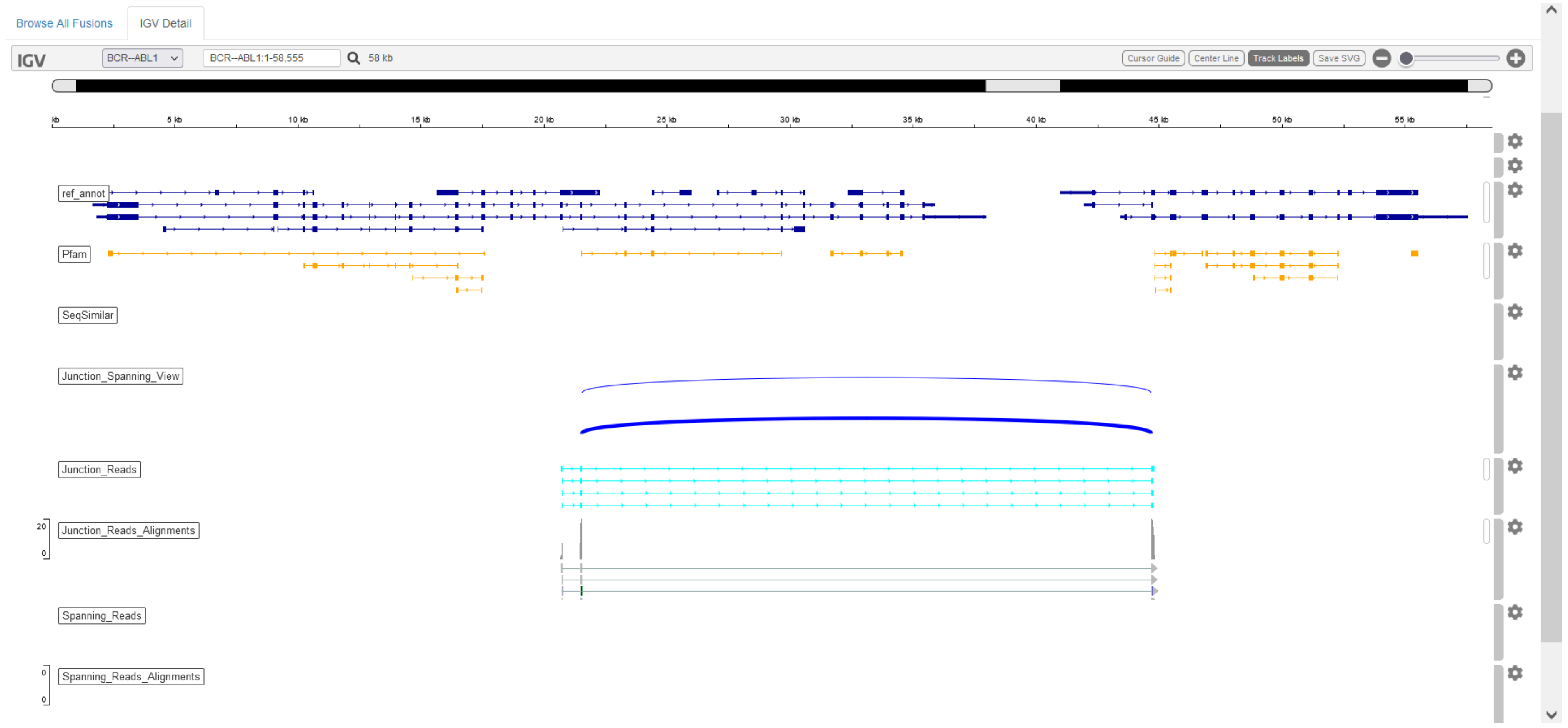


Figure 7.6: The FusionInspector HTML summary visualisation (sample p1_S4). IGV view of the BCR::ABL1 fusion gene.

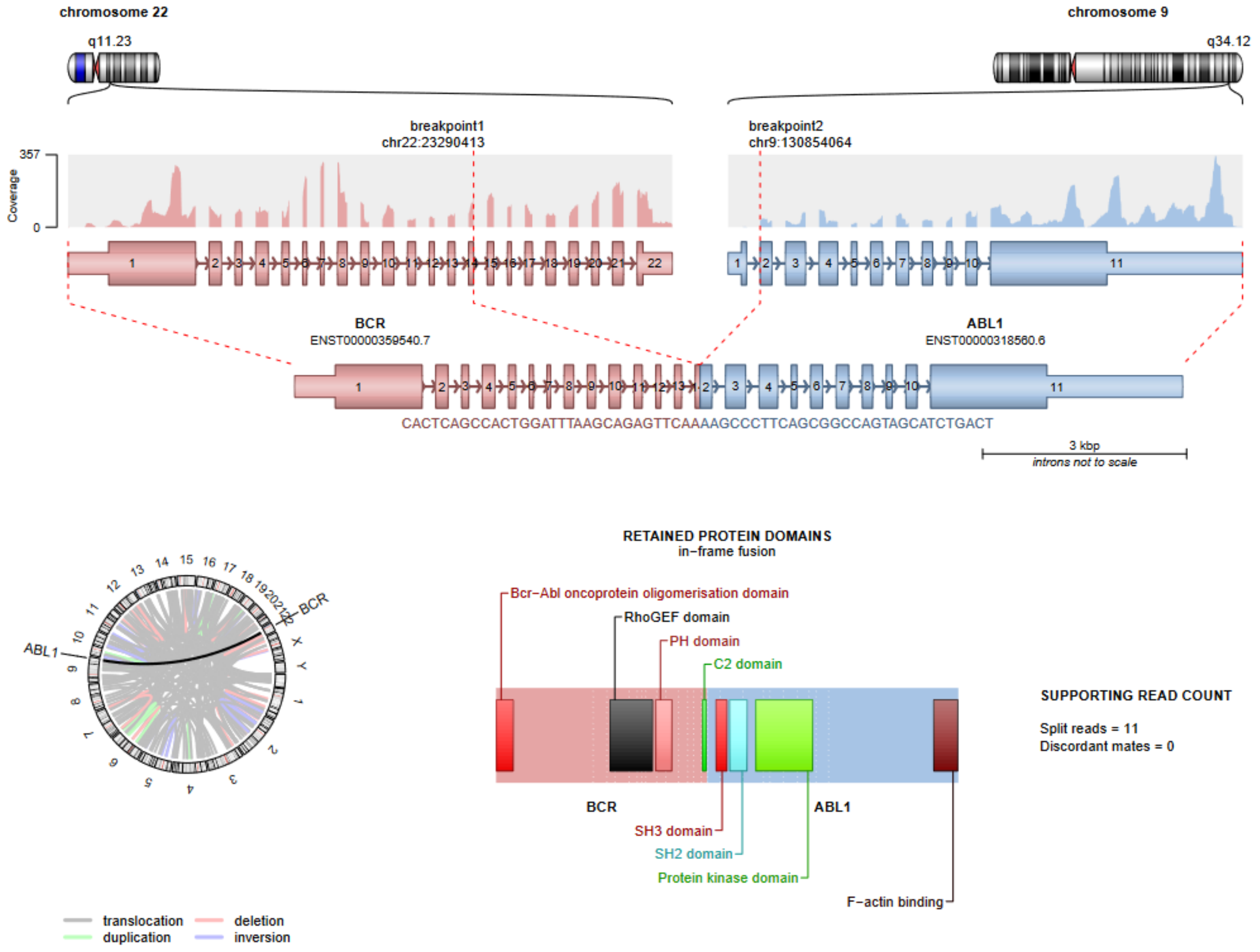
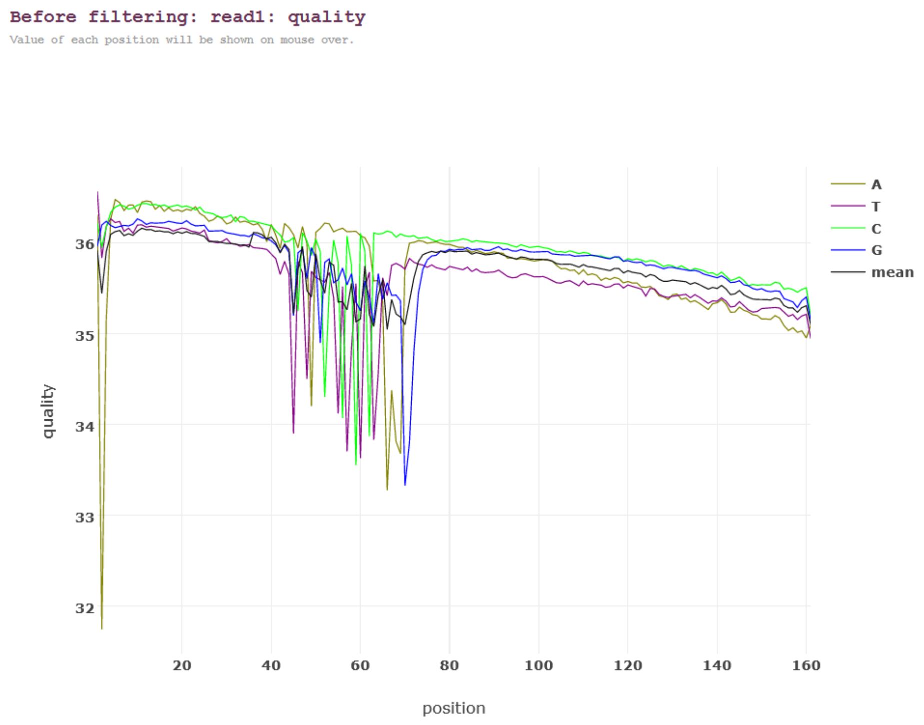


Figure 7.7: The Arriba PDF visualisation of the BCR::ABL1 fusion gene detected by FusionInspector (sample p1_S4)



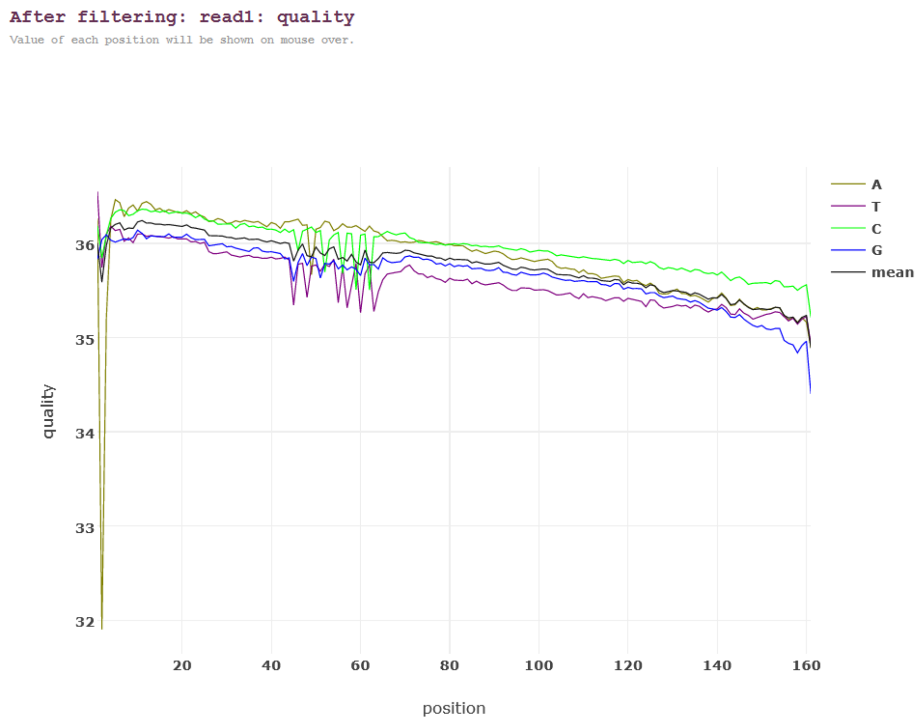


Figure 7.8: Example of Fastp HTML visualisation of quality control and filtering results (sample p1_S4). Reads quality before and after filtering.

7.2.3 Detected Fusion Gene Candidates

As the result of our solution is intended to be a summary of predicted fusion candidates for expert analysis, here we provide merely an aggregation of the analysis outcome for the two of the testing datasets, see Table 7.3.

Table 7.3: Summary of testing analysis results

	Number of fusions reported		Detected BCR::ABL1	
Caller	Sample p1_S4	Sample p2_S10	Sample p1_S4	Sample p2_S10
JAFFA	275	272	Yes	Yes
FusionCatcher	71	82	Yes	Yes
Arriba	7	13	Yes	Yes
STAR-Fusion	13	7	Yes	Yes
Cicero	120	172	Yes	Yes
FusionInspector	245	240	Yes	Yes

8 Discussion

The aim was to build a bioinformatics pipeline for RNAseq data analysis of leukemia patients with the emphasis on fusion gene detection. We mapped currently available fusion gene detection tools and their methodology and selected five callers, JAFFA, Arriba, STAR-Fusion, Cicero, and FusionCatcher, that matched our criteria.

Since fusion callers generally vary in their approaches, they differ in sensitivity to different gene fusions. Thus, using multiple tools leads to a wider range of results and higher probability of reporting the relevant fusions for the given sample. Although we provide a level of validation by including FusionInspector, our solution serves as a tool that delivers a preprepared summary of fusion gene candidates and their visualisation that are intended to be further analysed by an expert.

As one of the main motivations for this thesis was the need of a tool that overcomes the issues with portability and resource management that come with such a complex bioinformatics analysis, we followed current trends and leveraged Nextflow as the programming framework.

The pipeline comes with prepared options to execute the processes locally or with Kubernetes executor without any need to modify the code. To provide enough flexibility, changing the target system for execution can be easily achieved through a configuration file.

Testing the solution on two real datasets demonstrated that building a pipeline with Nextflow not only automated the analysis, but also provided efficient resource management and parallelization of distinct tasks of both single sample and multiple samples analysis.

As far as we can interpret the detected fusion candidates in the testing datasets without an expert analysis, the fusion gene BCR::ABL1 was reported in both samples as expected based on the patients diagnosis.

We provide a tool that was developed in a modular approach. It is structured in such a way that including another module for different analysis (e.g. RNAseq differential expression analysis) on the same data can be easily achieved in the future.

9 Conclusions

In this thesis we focused on leveraging the features of Nextflow to overcome the issues that come along with developing a complex bioinformatic pipeline for RNAseq analysis. The aim was to provide a tool for fusion gene detection that is portable, easily deployed and well documented.

To reach that, we mapped current trends in methodology of RNAseq data processing and fusion gene detection and came with a workflow with a summary table of fusion candidates for subsequent expert analysis on the output.

We provide our solution along with documentation in a form of GitLab project that can be run locally from any machine with Linux and Nextflow, without any additional installation steps required.

List of Literature

[1] F. Mertens, B. Johansson, T. Fioretos, and F. Mitelman, ‘The emerging complexity of gene fusions in cancer’, *Nat Rev Cancer*, vol. 15, no. 6, pp. 371–381, Jun. 2015, doi: 10.1038/nrc3947.

[2] H. Wu, X. Li, and H. Li, ‘Gene fusions and chimeric RNAs, and their implications in cancer’, *Genes & Diseases*, vol. 6, no. 4, pp. 385–390, Dec. 2019, doi: 10.1016/j.gendis.2019.08.002.

[3] B. J. Haas, A. Dobin, B. Li, N. Stransky, N. Pochet, and A. Regev, ‘Accuracy assessment of fusion transcript detection via read-mapping and de novo fusion transcript assembly-based methods’, *Genome Biol*, vol. 20, no. 1, p. 213, Dec. 2019, doi: 10.1186/s13059-019-1842-9.

[4] D. C. Corney, ‘RNA-seq Using Next Generation Sequencing’, *MATER METHODS*, vol. 3, Aug. 2013, doi: 10.13070/mm.en.3.203.

[5] P. Di Tommaso, M. Chatzou, E. W. Floden, P. P. Barja, E. Palumbo, and C. Notredame, ‘Nextflow enables reproducible computational workflows’, *Nat Biotechnol*, vol. 35, no. 4, pp. 316–319, Apr. 2017, doi: 10.1038/nbt.3820.

[6] Istvan Albert, *The Biostar Handbook*, 2nd Edition. Biostar Genomics, 2021. [Online]. Available: https://www.biostarhandbook.com/index.html

[7] H. F. Lodish, Ed., *Molecular cell biology*, 5th ed. New York: W.H. Freeman, 2003.

[8] ‘Talking Glossary of Genomic and Genetic Terms’, *National Human Genome Research Institute*. https://www.genome.gov/genetics-glossary

[9] S. R. Bolsover, Ed., *Cell biology: a short course*, 2nd ed. Hoboken, N.J: Wiley-Liss, 2004.

[10] Polyak K. and Meyerson M., ‘Overview: Gene Structure’, in *Holland-Frei Cancer Medicine*, 6th ed.Hamilton (ON): BC Decker, 2003. Accessed: Jan. 26, 2022. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK12983/

[11] G.-W. Li and X. S. Xie, ‘Central dogma at the single-molecule level in living cells’, *Nature*, vol. 475, no. 7356, pp. 308–315, Jul. 2011, doi: 10.1038/nature10315.

[12] S. Gunes and A. M. Mahmutoglu, ‘Transcriptomics and Oxidative Stress in Male Infertility’, in *Oxidants, Antioxidants and Impact of the Oxidative Status in Male Reproduction*, Elsevier, 2019, pp. 249–260. doi: 10.1016/B978-0-12-812501-4.00023-7.

[13] Y. Wang, N. Wu, J. Liu, Z. Wu, and D. Dong, ‘FusionCancer: a database of cancer fusion genes derived from RNA-seq data’, *Diagn Pathol*, vol. 10, no. 1, p. 131, Dec. 2015, doi: 10.1186/s13000-015-0310-4.

[14] J. Li *et al.*, ‘A functional genomic approach to actionable gene fusions for precision oncology’, *Sci. Adv.*, vol. 8, no. 6, p. eabm2382, Feb. 2022, doi: 10.1126/sciadv.abm2382.

[15] G. Picco *et al.*, ‘Functional linkage of gene fusions to cancer cell fitness assessed by pharmacological and CRISPR-Cas9 screening’, *Nat Commun*, vol. 10, no. 1, p. 2198, May 2019, doi: 10.1038/s41467-019-09940-1.

[16] C. Kumar-Sinha, S. Kalyana-Sundaram, and A. M. Chinnaiyan, ‘Landscape of gene fusions in epithelial cancers: seq and ye shall find’, *Genome Med*, vol. 7, no. 1, p. 129, Dec. 2015, doi: 10.1186/s13073-015-0252-1.

[17] S. Uhrig *et al.*, ‘Accurate and efficient detection of gene fusions from RNA sequencing data’, *Genome Res.*, vol. 31, no. 3, pp. 448–460, Mar. 2021, doi: 10.1101/gr.257246.119.

[18] M. Engvall, N. Cahill, B.-I. Jonsson, M. Höglund, H. Hallböök, and L. Cavelier, ‘Detection of leukemia gene fusions by targeted RNA-sequencing in routine diagnostics’, *BMC Med Genomics*, vol. 13, no. 1, p. 106, Dec. 2020, doi: 10.1186/s12920-020-00739-4.

[19] J. Schröder, A. Kumar, and S. Q. Wong, ‘Overview of Fusion Detection Strategies Using Next-Generation Sequencing’, in *Tumor Profiling*, S. S. Murray, Ed., New York, NY: Springer New York, 2019, pp. 125–138. doi: 10.1007/978-1-4939-9004-7_9.

[20] S. Faderl, M. Talpaz, Z. Estrov, and H. M. Kantarjian, ‘Chronic Myelogenous Leukemia: Biology and Therapy’, *Ann Intern Med*, vol. 131, no. 3, p. 207, Aug. 1999, doi: 10.7326/0003-4819-131-3-199908030-00008.

[21] R. Hehlmann, A. Hochhaus, and M. Baccarani, ‘Chronic myeloid leukaemia’, *The Lancet*, vol. 370, no. 9584, pp. 342–350, Jul. 2007, doi: 10.1016/S0140-6736(07)61165-9.

[22] E. Vuelta, I. García-Tuñón, P. Hernández-Carabias, L. Méndez, and M. Sánchez-Martín, ‘Future Approaches for Treating Chronic Myeloid Leukemia: CRISPR Therapy’, *Biology*, vol. 10, no. 2, p. 118, Feb. 2021, doi: 10.3390/biology10020118.

[23] A. K. Gupta and U. Gupta, ‘Next generation sequencing and its applications’, in *Animal Biotechnology*, Elsevier, 2020, pp. 395–421. doi: 10.1016/B978-0-12-811710-1.00018-5.

[24] R. Pereira, J. Oliveira, and M. Sousa, ‘Bioinformatics and Computational Tools for Next-Generation Sequencing Analysis in Clinical Genetics’, *JCM*, vol. 9, no. 1, p. 132, Jan. 2020, doi: 10.3390/jcm9010132.

[25] The SAM/BAM Format Specification Working Group, ‘Sequence Alignment/Map Format Specification’, *samtools*, Mar. 06, 2021. https://github.com/samtools/hts-specs/blob/master/SAMv1.pdf (accessed Nov. 22, 2021).

[26] Illumina, Inc., ‘Advantages of paired-end and single-read sequencing’, *Illumina*. https://www.illumina.com/science/technology/next-generation-sequencing/plan-experiments/paired-end-vs-single-read.html (accessed Jan. 26, 2022).

[27] Q. Wang, J. Xia, P. Jia, W. Pao, and Z. Zhao, ‘Application of next generation sequencing to human gene fusion detection: computational tools, features and perspectives’, *Briefings in Bioinformatics*, vol. 14, no. 4, pp. 506–519, Jul. 2013, doi: 10.1093/bib/bbs044.

[28] S. R. Head *et al.*, ‘Library construction for next-generation sequencing: Overviews and challenges’, *BioTechniques*, vol. 56, no. 2, pp. 61–77, Feb. 2014, doi: 10.2144/000114133.

[29] M. Sagoff, ‘Data deluge and the human microbiome project’, vol. 28, pp. 71–78, Jun. 2012.

[30] R. McFadyen, ‘Intro to NGS Data Analysis Workflow’, *Diagnostech*, Aug. 03, 2020. https://diagnostech.co.za/intro-to-ngs-data-analysis-workflow/ (accessed Jan. 16, 2022).

[31] S. Zhao *et al.*, ‘Bioinformatics for RNA-Seq Data Analysis’, in *Bioinformatics - Updated Features and Applications*, I. Y. Abdurakhmonov, Ed., InTech, 2016. doi: 10.5772/63267.

[32] B. J. Haas and M. C. Zody, ‘Advancing RNA-Seq analysis’, *Nat Biotechnol*, vol. 28, no. 5, pp. 421–423, May 2010, doi: 10.1038/nbt0510-421.

[33] Z. Wang, M. Gerstein, and M. Snyder, ‘RNA-Seq: a revolutionary tool for transcriptomics’, *Nat Rev Genet*, vol. 10, no. 1, pp. 57–63, Jan. 2009, doi: 10.1038/nrg2484.

[34] ‘Functional genomics (II): Common technologies and data analysis methods’. 2016. Accessed: Jan. 28, 2022. [Online]. Available: https://www.ebi.ac.uk/training/online/courses/functional-genomics-ii-common-technologies-and-data-analysis-methods/

[35] V. M. Kvam, P. Liu, and Y. Si, ‘A comparison of statistical methods for detecting differentially expressed genes from RNA-seq data’, *American Journal of Botany*, vol. 99, no. 2, pp. 248–256, Feb. 2012, doi: 10.3732/ajb.1100340.

[36] S. Zhao, L. Xi, and B. Zhang, ‘Union Exon Based Approach for RNA-Seq Gene Quantification: To Be or Not to Be?’, *PLoS ONE*, vol. 10, no. 11, p. e0141910, Nov. 2015, doi: 10.1371/journal.pone.0141910.

[37] M. Zhang *et al.*, 'Quantification of gene expression while taking into account RNA alternative splicing', *Genomics*, vol. 111, no. 6, pp. 1517–1528, Dec. 2019, doi: 10.1016/j.ygeno.2018.10.009.

[38] M. D. Robinson and A. Oshlack, 'A scaling normalization method for differential expression analysis of RNA-seq data', *Genome Biol*, vol. 11, no. 3, p. R25, 2010, doi: 10.1186/gb-2010-11-3-r25.

[39] J. Costa-Silva, D. Domingues, and F. M. Lopes, 'RNA-Seq differential expression analysis: An extended review and a software tool', *PLoS ONE*, vol. 12, no. 12, p. e0190152, Dec. 2017, doi: 10.1371/journal.pone.0190152.

[40] T. Samazan, 'Next Generation Sequencing Data: Tertiary Analysis', *Bioinformatics - Blog*, Sep. 07, 2017. https://bioinfoinc.com/blog/next-generation-sequencing-data-tertiary-analysis/ (accessed Feb. 02, 2022).

[41] A. Oshlack, M. D. Robinson, and M. D. Young, 'From RNA-seq reads to differential expression results', *Genome Biol*, vol. 11, no. 12, p. 220, 2010, doi: 10.1186/gb-2010-11-12-220.

[42] S. Kumar, S. K. Razzaq, A. D. Vo, M. Gautam, and H. Li, 'Identifying fusion transcripts using next generation sequencing', *WIREs RNA*, vol. 7, no. 6, pp. 811–823, Nov. 2016, doi: 10.1002/wrna.1382.

[43] Y. W. Asmann *et al.*, 'A novel bioinformatics pipeline for identification and characterization of fusion transcripts in breast cancer and normal cell lines', *Nucleic Acids Research*, vol. 39, no. 15, pp. e100–e100, Aug. 2011, doi: 10.1093/nar/gkr362.

[44] G. R. Oliver *et al.*, 'A tailored approach to fusion transcript identification increases diagnosis of rare inherited disease', *PLoS ONE*, vol. 14, no. 10, p. e0223337, Oct. 2019, doi: 10.1371/journal.pone.0223337.

[45] T. N. Vu, W. Deng, Q. T. Trac, S. Calza, W. Hwang, and Y. Pawitan, 'A fast detection of fusion genes from paired-end RNA-seq data', *BMC Genomics*, vol. 19, no. 1, p. 786, Dec. 2018, doi: 10.1186/s12864-018-5156-1.

[46] B. J. Haas *et al.*, 'Targeted *in silico* characterization of fusion transcripts in tumor and normal tissues via FusionInspector', Bioinformatics, preprint, Aug. 2021. doi: 10.1101/2021.08.02.454639.

[47] W. Hu *et al.*, 'Development and validation of an RNA sequencing panel for gene fusions in soft tissue sarcoma', *Cancer Science*, vol. 113, no. 5, pp. 1843–1854, May 2022, doi: 10.1111/cas.15317.

[48] Seqera Labs, 'Nextflow training', *2. Introduction*, 2020. https://seqera.io/training/ (accessed Feb. 02, 2022).

[49] CESNET, z. s. p. o, 'MetaCentrum VO', *MetaCentrum VO*, Nov. 13, 2019. https://www.metacentrum.cz/en/VO/metavo/ (accessed Nov. 21, 2021).

[50] E. Conrad, S. Misenar, and J. Feldman, 'Domain 3: Security Engineering (Engineering and Management of Security)', in *CISSP Study Guide*, Elsevier, 2016, pp. 103–217. doi: 10.1016/B978-0-12-802437-9.00004-7.

[51] R. J. Anthony, 'The Architecture View', in *Systems Programming*, Elsevier, 2016, pp. 277–382. doi: 10.1016/B978-0-12-800729-7.00005-4.

[52] W. Tian and Y. Zhao, 'An Introduction to Cloud Computing', in *Optimized Cloud Resource Management and Scheduling*, Elsevier, 2015, pp. 1–15. doi: 10.1016/B978-0-12-801476-9.00001-X.

[53] J. McGovern, S. Tyagi, M. E. Stevens, and S. Mathew, 'Practical Considerations', in *Java Web Services Architecture*, Elsevier, 2003, pp. 689–726. doi: 10.1016/B978-155860900-6/50021-X.

[54] 'Introduction', *Metacentrum Cloud Documentation*. https://docs.cloud.muni.cz/ (accessed Feb. 02, 2022).

[55] Bashari Rad, Babak, Bhatti, Harrison, and Ahmadi, Mohammad, 'An Introduction to Docker and Analysis of its Performance', *IJCSNS International Journal of Computer Science and Network Security*, vol. 173, p. 8, Mar. 2017.

[56] C. Anderson, ‘Docker [Software engineering]’, *IEEE Softw.*, vol. 32, no. 3, pp. 102-c3, May 2015, doi: 10.1109/MS.2015.62.
[57] Merkel, Dirk, ‘Docker: lightweight linux containers for consistent development and deployment’, *Linux journal*, vol. 2014, no. 239, p. 2, 2014.
[58] E. Kim, K. Lee, and C. Yoo, ‘On the Resource Management of Kubernetes’, in *2021 International Conference on Information Networking (ICOIN)*, Jeju Island, Korea (South): IEEE, Jan. 2021, pp. 154–158. doi: 10.1109/ICOIN50884.2021.9333977.
[59] Md. S. Islam Shamim, F. Ahamed Bhuiyan, and A. Rahman, ‘XI Commandments of Kubernetes Security: A Systematization of Knowledge Related to Kubernetes Security Practices’, in *2020 IEEE Secure Development (SecDev)*, Atlanta, GA, USA: IEEE, Sep. 2020, pp. 58–64. doi: 10.1109/SecDev45635.2020.00025.
[60] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, ‘Borg, Omega, and Kubernetes: Lessons learned from three container-management systems over a decade’, *Queue*, vol. 14, no. 1, pp. 70–93, Jan. 2016, doi: 10.1145/2898442.2898444.
[61] S. Chen, Y. Zhou, Y. Chen, and J. Gu, ‘fastp: an ultra-fast all-in-one FASTQ preprocessor’, *Bioinformatics*, vol. 34, no. 17, pp. i884–i890, Sep. 2018, doi: 10.1093/bioinformatics/bty560.
[62] A. Dobin *et al.*, ‘STAR: ultrafast universal RNA-seq aligner’, *Bioinformatics*, vol. 29, no. 1, pp. 15–21, Jan. 2013, doi: 10.1093/bioinformatics/bts635.
[63] H. Li *et al.*, ‘The Sequence Alignment/Map format and SAMtools’, *Bioinformatics*, vol. 25, no. 16, pp. 2078–2079, Aug. 2009, doi: 10.1093/bioinformatics/btp352.
[64] N. M. Davidson, I. J. Majewski, and A. Oshlack, ‘JAFFA: High sensitivity transcriptome-focused fusion gene detection’, *Genome Med*, vol. 7, no. 1, p. 43, Dec. 2015, doi: 10.1186/s13073-015-0167-x.
[65] D. Nicorici *et al.*, ‘FusionCatcher - a tool for finding somatic fusion genes in paired-end RNA-sequencing data’, Bioinformatics, preprint, Nov. 2014. doi: 10.1101/011650.
[66] L. Tian *et al.*, ‘CICERO: a versatile method for detecting complex and diverse driver fusions using cancer RNA sequencing data’, *Genome Biol*, vol. 21, no. 1, p. 126, Dec. 2020, doi: 10.1186/s13059-020-02043-x.
[67] P. Danecek *et al.*, ‘Twelve years of SAMtools and BCFtools’, *GigaScience*, vol. 10, no. 2, p. giab008, Jan. 2021, doi: 10.1093/gigascience/giab008.
[68] Brian Haas and Alex Dobin, ‘FusionInspector: In silico Validation of Fusion Transcript Predictions’, *FusionInspector: In silico Validation of Fusion Transcript Predictions*. https://github.com/FusionInspector/FusionInspector/wiki
[69] SciLifeLab, ‘standalone_scripts’, *GitHub*. https://github.com/SciLifeLab/standalone_scripts
[70] Brian Haas, ‘CTAT_HumanFusionLib’, *GitHub*, Dec. 30, 2022. https://github.com/FusionAnnotator/CTAT_HumanFusionLib/wiki