UNIVERSITY OF BERGEN

DEPARTMENT OF INFORMATICS

A Snakemake workflow for analysing alternative splicing events from short-read RNA-seq data

Author: Madelen Holberg

Supervisors: Sushma Nagaraja Grellscheid and Franziska Stilla Maria Görtler

UNIVERSITETET I BERGEN

Det matematisk-naturvitenskapelige fakultet

April 13, 2022

Abstract

The DNA of all living organisms consists of coding and non-coding sections. RNA and proteins are formed by translating coding sections of the DNA into RNA, and then to proteins. The coding sequence translated into one special protein is not always identical. Proteins resulting from the same gene but with different sequences are called homologes. The process of combining protein-coding sections of an RNA-sequence to attain distinct isoforms of a protein is called alternative splicing. Alternative splicing is a an important mechanism in gene expression and as a mechanism for increasing proteome diversity. Due to alternative splicing, one gene can code for multiple homologes of one protein (Greenberg and Soreq 2013). One computational tool, among others, for identifying alternatively spliced genes is rMATS. rMATS finds annotated and novel alternative splice sites for unpaired and paired RNA replicates.

In 2012, Snakemake was published as a workflow management system able to combine all the steps required for data analysis (Köster and Rahmann 2012). This makes the analysis reproducible, adaptable and transparent.

In this thesis, we designed a Snakemake workflow for analysis of alternative splicing events. This combines the steps for splicing analysis in a single document. Further, we used rMATS as the tool for detecting alternative splicing events in our experimental data and investigated the differences between our treatment and control group for some of the interesting genes.

We constructed a workflow in Snakemake which is suitable for analysing alternative splicing events. Furthermore, several instances of alternative splicing events in the control group of our case study were found. These splic-

ing events were regarded in the context of retinitis pigmentosa. We conclude that the Snakemake workflow is an acceptable program for further research of alternative splicing. The developed Snakemake workflow was made public and available on GitHub (*https://github.com/MadelenH/Snakemake*).

Acknowledgments

First and foremost, I would like to thank my supervisors, Sushma Grellscheid and Franziska Görtler, for their guidance and involvement throughout my degree. Their support and feedback have made this thesis possible.

I would also like to thank my family for their love and support over the past years.

Contents

List of Figures

Listings

Acronyms

A3SS Alternative 3' Splice Site. 59

DGE Differential Gene Expression. 15

DNA Deoxyribonucleic Acid. 1

ESTs Expressed Sequence Tags. 7

IJC Included Junction Counts. 58

KDM6A Lysine Demethylase 6A. 56

MMP Maximal Mappable Prefix. 29

mRNA messenger RNA. 1, 5

MTF2 Metal Response Element Binding Transcription Factor 2. 55

MXE Mutually Exclusive Exons. 55

NGS Next Generation Sequencing. 15

pre-mRNA Precursor-messenger RNA. 6

PRPF3 Pre-mRNA Processing Factor 3. 58

PSI Percent Spliced In. 10

RI Retained Introns. 56

rMATS Replicate Multivariate Analysis of Transcript Splicing. 9

RP Retinitis Pigmentosa. 13

SE Skipped Exons. 57

SJC Skipped Juntion Counts. 58

STAR Spliced Transcripts Alignment to a Reference. 29

Chapter 1

Introduction

1.1 Motivation

The genetic information of all living organisms are stored in the double helix called Deoxyribonucleic Acid (DNA). The DNA encodes protein molecules through transcription of a gene into messenger RNA (mRNA), which in turn is translated into proteins (Clancy and Brown 2008). This process is described in detail in Section 2.1.

Prior to the translation of mRNA into proteins, non-coding portions (introns) of the RNA-sequence is extracted. Mature mRNA is produced by RNA splicing which joins the remaining protein coding sections of the mRNA (exons), but the same genes does not always lead to the same mRNA. This process is called alternative splicing (Greenberg and Soreq 2013).

The multiple mRNAs that arise from a single gene as a result of the different combinations of the exons, are called the isoforms of a protein. The proteins encoded by the different isoforms vary in their sequence and activity. The process of identifying the alternative splicing events present in a gene is essential, as misregulation can lead to splicing defects and cause diseases

(Thakur et al. 2019).

The protocol for alternative splicing analysis consists of multiple steps adapted to the user's data-set which are executed separately. In order to make the analysis less time consuming and more adaptable, a Snakemake workflow can be applied to combine every step required in one document.

1.2 Scope

A single gene can code for multiple mRNAs due to the different methods of splicing the protein-coding sections. As alternative splicing can cause genetic diseases, this thesis will cover the alternative splicing analysis of the genetic disease retinitis pigmentosa, as well as a control group, in the Snakemake workflow.

The alternative splicing analysis of the disease retinitis pigmentosa is chosen for this thesis because it's one of the most common diseases in the retina, and the fact that the previous studies done on this disease have not used the Snakemake workflow.

The scope of the study is limited to the genes PRPF31, PRPF3, KDM6A and MTF2 due to the amount of genes including alternative splicing events in both retinitis pigmentosa and the control group.

To summarize, the scope of this thesis is to use the Snakemake workflow on the case of one genetic disorder and analyse the splicing events of a few genes present there.

1.3 Aims

This thesis aims to create a Snakemake workflow document as a tool for alternative splicing analysis, containing the necessary steps to execute analysis. The document aims to be applicable to raw fastq data sequences containing minimum two conditions in the experimental design, i.e treatment control experiments.

This thesis aims to use retinitis pigmentosa data to show that the implemented snakemake pipeline delivers reasonable alternative splicing results.

1.4 Thesis Outline

1. **Chapter 2** introduces the theoretical background related to this thesis. First, a detailed description of the alternative splicing process and events are presented. Then the computational program used for analysis is introduced, as well as a description of the Snakemake workflow. Lastly, information about the case study and data used in this thesis is presented.

2. **Chapter 3** introduces the methods implemented to the Snakemake workflow. This includes quality control, trimmomatic, STAR indexing and mapping, samtools-sort and -indexing, and rMATS. Every method can be located through the list of Listings.

3. **Chapter 4** presents the results of the Snakemake workflow. This includes the repository available on GitHub.

4. **Chapter 5** presents the results of the case study. This encompasses the results of every step from Chapter 4.

5. **Chapter 6** discusses the results of the case study and the Snakemake workflow.

Chapter 2

Theory

This chapter will deal with process of alternative splicing, the different splicing event types and the workflow engine Snakemake (Mölder et al. 2021), which provides a readable Python-based definition language. It will also give an introduction to the event-based method rMATS and the data used in this thesis.

2.1 The process of alternative splicing

The DNA located in the nucleus of every cell encodes protein molecules from the nucleotides in the DNA-strand (*DNA* 2022). The genes coding for proteins are called codons, and each codon codes for a single amino acid. To create a protein from DNA, the information coding for one gene in the DNA is transcribed into a messenger RNA (mRNA). The DNA is the template used for complementary base-pairing utilizing the enzyme RNA polymerase, which generates the mature mRNA. Subsequently, the mRNA is translated with the genetic code that relates the DNA sequence to the amino acid sequence in proteins (Clancy and Brown 2008).

Before the mRNA is translated into proteins, the fragment of the RNA-sequence that is non-coding (intron) is removed, and the protein-coding sections (exons) are spliced together. This process produces mature mRNA, and is called alternative splicing. The regulated process selects different combinations of splice sites with precursor-messenger RNA (pre-mRNA) to produce variably spliced mRNAs. The multiple mRNAs that arise from a single gene as a result of the different combinations of the exons, are called the isoforms of a protein. The proteins encoded by the different isoforms vary in their sequence and activity. (Greenberg and Soreq 2013).

Previous studies done on alternative splicing, such as *"A genomic view of alternative splicing"*(Modrek and C. 2002), show that 40-60% of all human genes have evidence of at least one alternative splice form. This indicate that with alternative splicing, organisms can produce far more proteins than their number of genes in the DNA might suggest.

2.2 Alternative Splicing Events

The first concept of alternative splicing proposed by Gilbert W. in 1978 (Gilbert 1978) contradicted to the previous theory of one gene - one RNA - one protein. During the last few decades our knowledge of the concept has grown substantially. Numerous studies have reiterated how important the role in which alternative splicing has across biological systems. High-resolution mass spectrometry analyses revealed that from the approximately 20,000 human protein-coding genes, 37% of them generates multiple protein isoforms (Braelle and Giudice 2017).

The research of alternative splicing has resulted in seven main types of alternative splicing events, detected by systematic analyses of expressed se-

quence tags (ESTs) and microarray data (Wang et al. 2015). The different splicing events are depicted in Figure 2.1. ESTs are mRNA fragments obtained through single sequencing reactions executed on clones from cDNA libraries that are randomly selected (Parkinson and Blaxter 2009).

The most common type of alternative splicing in the human genome, which accounts for approximately 38% (Koren, Lev-Maor, and Ast 2007) of all alternatively spliced events, is exon skipping (cassette exon splicing, see Figure 2.1: C). This splicing event either includes or skips an intervening exon that is between two other exons on the mRNA-Seq.

The next most common splicing event is alternative 3' or 5' splice sites (approximately 18% and 8%). In these events a constitutive splice site flanks exons on one side, where there are two or several competing alternative splice sites on the other side. This results in an extension that is either included or excluded in the transcript (Koren, Lev-Maor, and Ast 2007).

The other less common splicing events are intron retention (Figure 2.1, F), constitutive splicing and mutually exclusive exons (see Figure 2.1, A and B respectively) (Le et al. 2015).

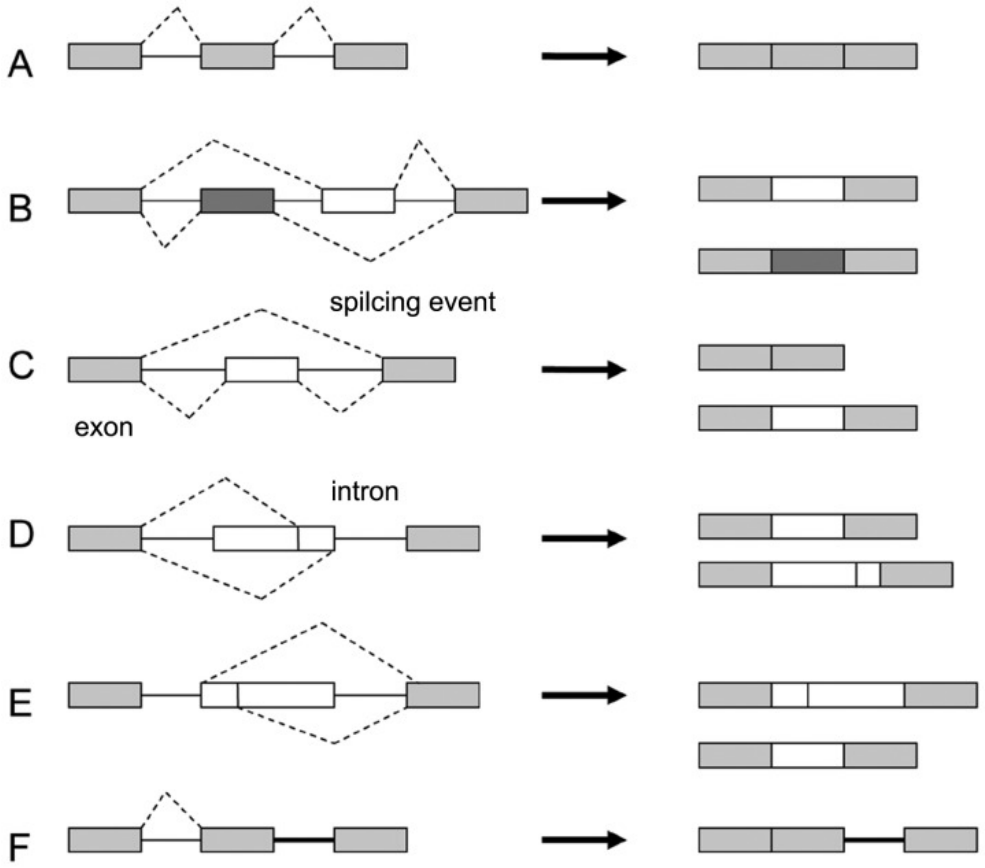

Figure 2.1: Shows the main types of alternative splicing events. (A) depicts constitutive splicing, where all the introns are spliced out, and the exons are joined. (B) is mutually exclusive exons, where only one exon (of a cluster) is included in the functional transcripts (Jin et al. 2018). (C) is cassette alternative exon, (D) and (E) are alternative 3' and 5' splice site, respectively. (F) is intron retention, where some of the introns are retained in mature mRNAs. Figure from (Wang et al. 2015).

There exists a number of computational methods for RNA-Seq analysis of alternative splicing, yet these methods do not handle replicate RNA-Seq data (MISO (Katz et al. 2010)) or model the estimation uncertainty of the proportion of isoforms in individual replicates (DiffSplice (Hu et al. 2012)).

One method that includes both these qualifications is rMATS, and this will be described in detail in Section 2.3. Here, we consider how rMATS detects differential alternative slicing events from replicate RNA-seq data, and in Section 5.3 we will go through the results received from rMATS.

2.3 rMATS - replicate multivariate analysis of transcript splicing

There are currently two major strategies applied for alternative splicing analysis. The first strategy is isoform-based. Here the full length transcripts are reconstructed and quantified regarding the analysis of differential expression, and used by tools such as cuffdiff2 (Trapnell, Hendrickson, et al. 2013) and DiffSplice (Hu et al. 2012). The second strategy is count-based. The count-based methods divide further into exon-based (e.g., limma (Shah and Pallas 2009) and edgeR (Robinson, McCarthy, and Smyth 2010)) or event-based methods (e.g., MAJIQ (Vaquero-Garcia et al. 2016) and rMATS (Shen, Park, Lu, et al. 2014)). Count-based methods configure genes into a single representation consisting of counting units. The number of sequencing reads falling on each counting unit are recorded, and differential expression analysis is used to call differentially expressed counting units (Mehmood et al. 2019).

In this thesis the event-based method Replicate Multivariate Analysis of Transcript Splicing (rMATS) (Shen, Park, Lu, et al. 2014) is used to find

annotated alternative splice events as well as novel events. rMATS is designed to detect differential alternative splicing from replicate RNA-Seq data. This is an improved version of the original MATS method (Shen, Park, Huang, et al. 2012). rMATS models the percent spliced in (PSI) for both sampling uncertainty within individuals as well as variability between samples. To do this, rMATS uses a likelihood ratio test with a user-defined threshold to see if the between-group differences of mean PSI exceed the given threshold.

For this analysis I will use rMATS which is a statistical modelling computer program, and was developed due to the lack of analytic tools that detect alternative splicing changes from replicate RNA-Seq data, while managing different types of replicate study designs. A hierarchical framework is used to model exon inclusion levels while accounting for estimation uncertainty in individual replicates and variability among replicates. rMATS models a correlation among matched base pairs, using a bivariate normal distribution with a correlation parameter, which improves the statistical power. Instead of testing the equality of exon inclusion levels between sample groups, rMATS uses a likelihood-ratio test where the alternative hypotheses are defined by users. This leads to assessing the statistical significance over any predefined magnitude of change in splicing. (Shen, Park, Lu, et al. 2014).

To analyse alternative splicing of paired replicates, rMATS uses a covariance structure so the paired replicates between two sample groups can be modeled.

2.4 Snakemake - the Workflow Engine

Working on large-scale data analyses involve execution of many command line applications and several analyzation tools. To ensure reproducibility

and to help automate these pipelines, I have utilized Snakemake.

Snakemake was first introduced in 2012 (Köster and Rahmann 2012) as the first system to support the use of automatically inferred multiple named wildcards in input and output filenames. Snakemake is a workflow engine that yields a readable Python-based definition language as well as a powerful execution environment without modifying the workflow. It is similar to other systems that have workflows which can be edited without a graphical environment (such as Pwrake (Tanaka and Tatebe 2010) and GXP Make (Taura et al. 2013)), except Snakemake complements with a syntax close to pseudocode, like the Python language (Mölder et al. 2021).

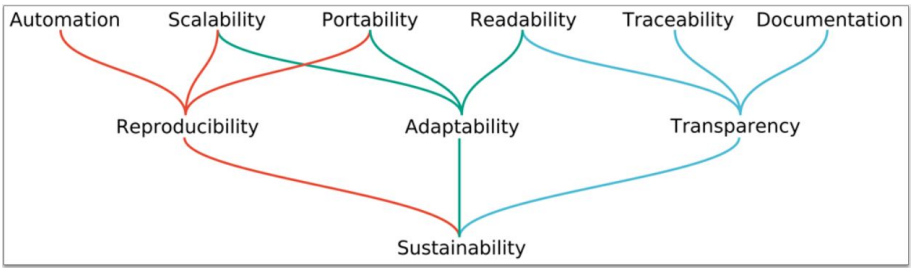

Figure 2.2: To get a sustainable data analysis, one must fulfill the requirements from the top layer first and continue down. Figure from (Mölder et al. 2021).

Snakemake supports full *in silico* reproducibility of data analyses, by having the data analyses automated, automatically deployed (portable) and able to access various computational platforms (scalable) (Figure 2.2). The data analyses are automated by fragmenting workflows into rules, where every rule describe the process of getting output files from input files. This ensures linear time needed for all the input files to be generated. For every rule in the workflow, Snakemake can use wildcards so the rules can run independently. For example in the case study of this thesis, a wildcard can

be applied to the filenames of data used so the rule can get an input and create an output that follows the pattern of the filenames. A wildcard can also be applied to find only a part of the filename, or two separate parts. The use of wildcards in Snakemake enables the program to get the names of its own and then independently go through the list to do one thing for all the files.

Snakemake is, in addition to automated, easily portable due to the directly integrated software stacks that are needed for individual rules. The scalable part of Snakemake ensures that one should only have to modify the command line parameters when adapting a workflow to a specific experiment. In addition to the achievements listed above, Snakemake's definition language is designed to allow maximal readability, which enables the user to understand or modify dependencies effortlessly. The results after processing a workflow are stored in reports containing information about input files, output files, parameters, software, and code of all involved steps. These reports enables exploration of the entire workflow adjacent to the information about their origin (Mölder et al. 2021).

A feature that can be implemented in the Snakemake file is wrappers, which generates an environment using the workflow. Snakemake executes the first wrapper located in the script, and downloads the corresponding wrapper files from GitHub. Every wrapper contains a version tag that can be replaced with an updated version (*The Snakemake Wrappers repository* 2016). Snakemake provides different wrappers and version tags for creating several different environments, which can be found on GitHub (*Wrappers* 2016).

2.5 Case Study

Retinitis Pigmentosa (RP) is a genetic disorder that entail breakdown of cells and loss of cells in the retina (*Retinitis Pigmentosa* 2021). Retina is the sensitive tissue in the back of the eye that converts the light impulses received through photoreceptors to electrical signals, and sends the signals through the optic nerve to the brain (W. Li 2021). Then the brain processes the electrical signals to the images that we see. The data of this thesis consists of samples of retinitis pigmentosa and a control group. The control group consists of samples not containing retinitis pigmentosa.

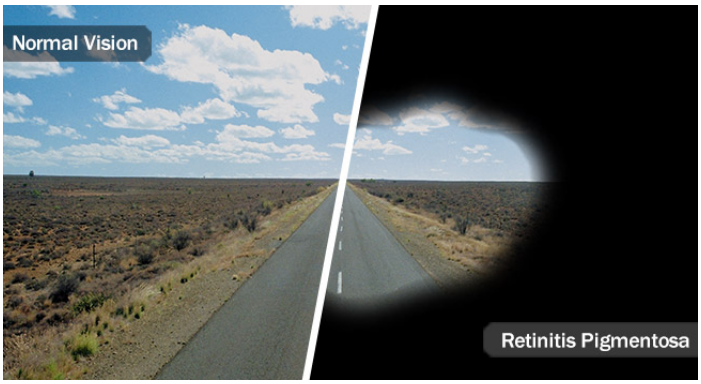

Figure 2.3: View from a person with normal vision and retinitis pigmentosa. Picture from (*Retinitis pigmentosa awareness month* 2022).

RP affects approximately 1 in 4,000 people and is resulting from various harmful mutations in more than 50 genes. If the mutations are severe the gene cannot produce the required proteins, or the proteins that are produced are toxic, which damages the photoreceptors (*Retinitis Pigmentosa* 2020). The first symptoms of RP can be noticed with loss of sight at night, and increasing loss of the visual field. Eventually, more of the visual field is lost, resulting in tunnel vision (Figure 2.3) (*Retinitis Pigmentosa* 2021).

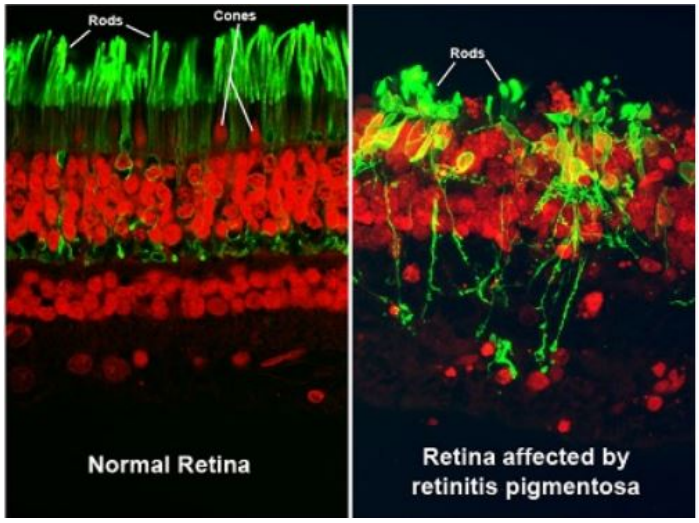

Figure 2.4: The destruction of rods and cones from patient with retinitis pigmentosa. Picture from (*Retinitis Pigmentosa* 2022).

The loss of vision is a result of loss of rods and cones in the eye. Rods are the photoreceptors in the human retina that are responsible for vision at levels with limited light. The cones in the retina are active at levels with increased light, and manage our color vision and eye color sensitivity. The rods and cones receive light signals and sends the converted electrical signals through the optic nerve to the brain (Kazilek and Cooper 2010). When someone has retinitis pigmentosa, the rods are more affected in the early stages, which results in night blindness and loss of visual field.

Eventually, the loss of rods leads to loss of cones (Figure 2.4), which results in reduced visual field (Figure 2.3) and difficulty perceiving details (Campochiaro and Mir 2018). Unfortunately, the damages of the rods and cones are irreversible, therefore in this thesis I will look at the connection between alternative splicing in genes and the number of alternative splicing events in genes with retinitis pigmentosa.

2.6 Data

In this thesis, I will look at the analysis of differential expression of genes in retinitis pigmentosa, with the main focus on genes that are overexpressed in patients with retinitis pigmentosa. Differential gene expression (DGE) analysis compares the gene expression values between sample groups. This is done to get an understanding of the molecular basis of variations in the phenotype (Rodriguez-Esteban and Jiang 2017). DNA microarrays have been used substantially for differential expression analysis, but since the decrease of the cost of sequencing (Wetterstrand 2021), RNA-seq has been used more often.

RNA-seq uses high-throughput sequencing to get information of the transcriptome of the cell, to examine the quantity of RNA in a sample using next-generation sequencing (NGS). NGS provides the order of nucleotides in entire genomes or targeted regions of RNA or DNA (*Introduction to NGS* 2021), which has provided the data used in this thesis. RNA-seq gives information about the level of transcription and the activation times of the genes, in order to understand the biology of a cell and assess changes that might indicate disease (Mackenzie 2021).

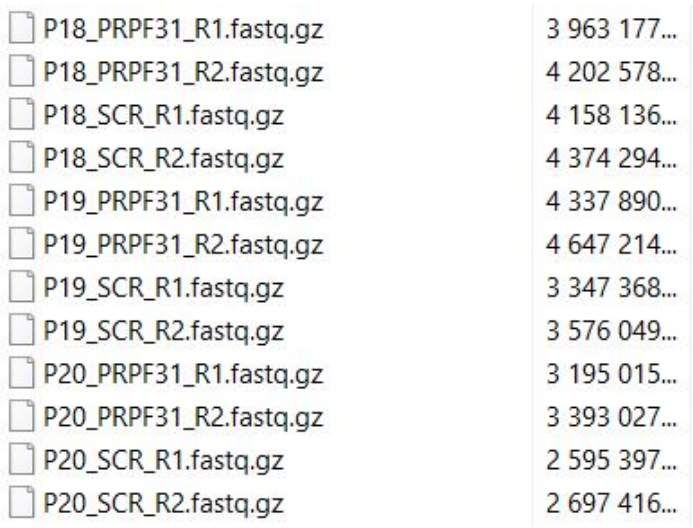

Figure 2.5: The FASTQ files used in this thesis, stored in folder raw_fastq.

The RNA-sequences used in this project are stored in FASTQ files that has converted base calls to sequence data. This process starts with the cluster generation and sequencing by synthesis with Illumina, which sequences billions of clusters on a flow cell. Base calls are made for every cluster and stored individually in base call files by the Real-Time Analysis software. The base calls are converted into sequence data when the sequencing completes, and this process is called BCL to FASTQ conversion (*FASTQ files explained* 2021).

The different files contains a single sequence corresponding to the sample's R1 FASTQ file, and R2 FASTQ for paired-end run. Every entry in the FASTQ files contains four lines that includes a sequence identifier, the sequence, a separator and the base call quality scores (*FastQ Format* 2022). The sequence identifier contains information about the sequencing run and the cluster. The sequence includes the base calls A, C, T, G and N, while the separator is simply a plus sign (*FASTQ files explained* 2021). The base call quality sores gives important information about the accuracy of the sequencing platform, and how much of the data is usable in an experiment.

In Figure 2.5 the files are labeled R1 and R2 which contains a single sequence for a paired-end run. The FASTQ files labeled SCR are the control croup sequences, while the other files contains the sequences with the disease retinitis pigmentosa. In Chapter 3 I will make use of these FASTQ-files to identify the difference of splicing levels between the control group and PRPF31-overrepresented genes in retinitis pigmentosa. The results presented in Chapter 5 reveals the different alternative splicing events that are present in the sequences.

Chapter 3

Method

In this section the protocol of analysis of alternative splicing is introduced (Figure 3.1). Each section will present the separate steps of the protocol, following the workflow of Snakemake. The results of the Snakemake workflow are shown in Chapter 4 and the results for the regarded data-sets are shown in Chapter 5.

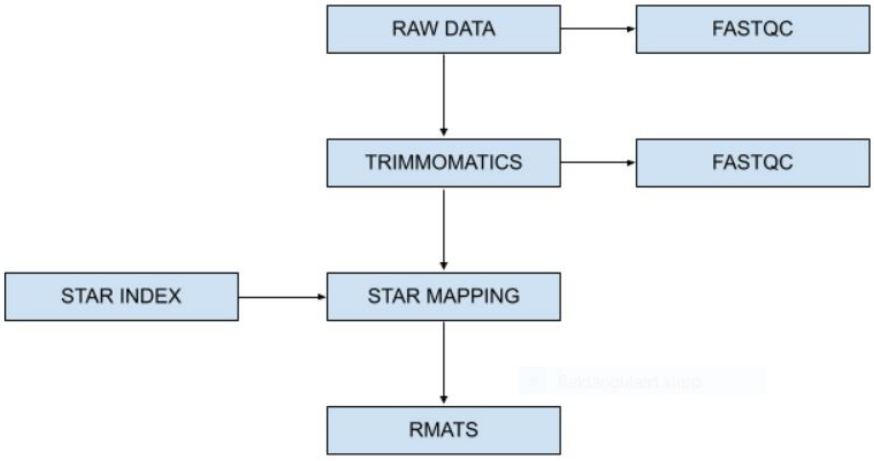

Figure 3.1: Protocol of the Snakemake workflow for analysis of alternative splicing.

Regarding the first step of Figure 3.1, starting with the raw fastq data set, a quality control will be executed to see if there are any errors in the data set. This is accomplished through FastQC (*Babraham Bioinformatics - FastQC A Quality Control tool for High Throughput Sequence Data* 2021), and described in detail in Section 3.2. Next, each sequence will be trimmed before running through FastQC again. The sequences are trimmed to remove the adapters on the sequences, explained in Section 3.3, and quality controlled again to look for errors and differences from before and after trimming. Then the reads will be mapped to a reference genome using STAR mapping, which requires a STAR index (Section 3.5). Lastly, the sequences are run with rMATS (Section 3.7).

3.1 Snakemake

Every step in the workflow protocol is adapted to the case-control study. The protocol is described in the following sections, where every section of code is adapted to the actual problem. For every rule in the workflow, a pseudo-rule at the beginning of the document is required, which initiates the Snakemake workflow. The Snakemake workflow automatically runs the next pseudo-rule when the previous rule is completed.

Overview of the location for the rules required in alternative splicing analysis:

1. **Section 3.2** includes the code for quality control (FastQC).
2. **Section 3.3** includes the code for adapter trimming with trimmomatics.
3. **Section 3.4** presents the code for creating a STAR index needed for

execution of STAR mapping.

4. **Section 3.5** shows the code for STAR mapping.

5. **Section 3.6** presents the code for sorting and indexing with samtools.

6. **Section 3.7** presents the code required before running rMATS, as well as the code for rMATS.

To execute the Snakemake workflow, I used the Conda package manager. Conda package manager (*Conda* 2017) sets up a separate environment to run the Snakemake workflow in, and is able to create, load and save everything in your working directory. By including *–use-conda* to the workflow execution command, conda is activated and Snakemake generates the necessary environments for wrappers used in the code.

To run the Snakemake file, I used the command *snakemake -j 24 –use-conda -s PATH/TO/SNAKEMAKE/WORKFLOW -p* in conda package manager in the terminal. The number 24 sets the total amount of cores available for the jobs in the Snakemake file. The path to Snakemake workflow is where the file including the rules for Snakemake is located. By adding *-rn* at the end of the Snakemake command, Snakemake makes a dry run. This means, it lists all the calculations and programs it will use and checks if the necessary requirements are specified. This enables an effective search as missing information is quickly visible.

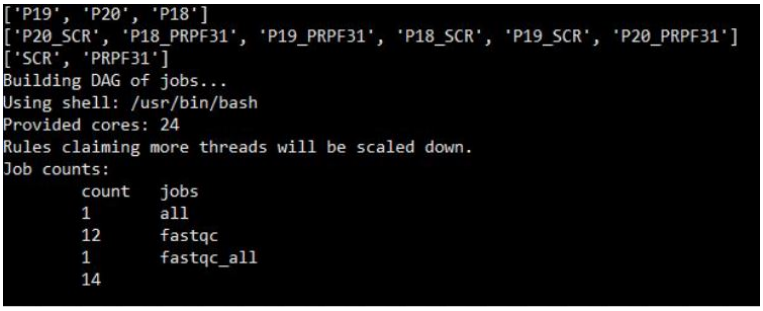

Figure 3.2: When running Snakemake workflow in the terminal, you get information about which job is running, the wildcards used, the amount of cores and the job counts.

Figure 3.2 is an example of the terminal-window when the Snakemake file has been executed without errors. Here one can see the used wildcards, which the user can inspect to see if all the data-sets are used. Furthermore, Snakemake informs through the message *"Building DAG of jobs"* that is is assessing how to connect the rules in the workflow. The terminal-window also includes the amount of used cores, as well as the rules used which helps the user to check if all the tasks you wanted to do in the workflow is really done.

In Figure 3.3, Snakemake provides the user with information on which input file is running through the rule, what output-files are created and which wildcards are used.

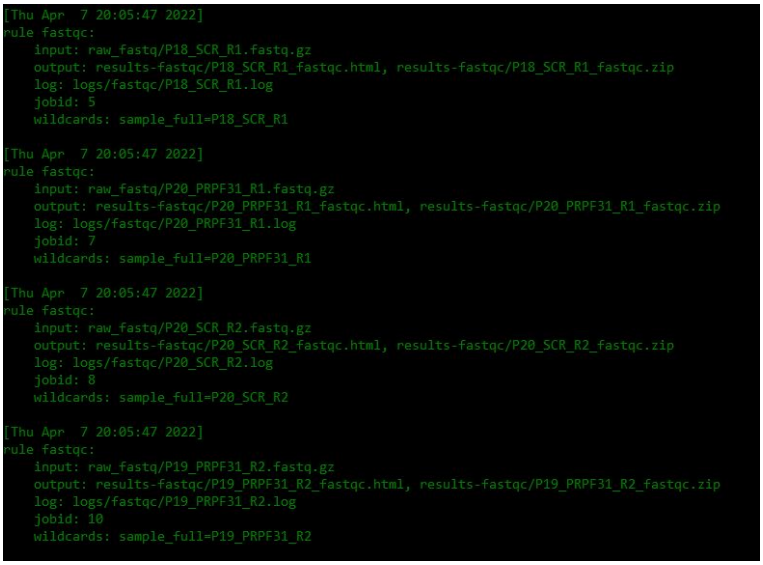

Figure 3.3: Example of the terminal-window during execution of Snakemake workflow.

From the example figure (Figure 3.3) which depicts the terminal-window while running the workflow, Snakemake provides which fastq-file is currently quality controlled, as well as the output-files created.

The results of the Snakemake workflow is presented in Chapter 4, while the results from the case study using Snakemake is presented in Chapter 5.

3.2 Quality control

Following the workflow of Snakemake (Figure 3.1), a quality control is run on the FastQ files. FastQC receives high throughput sequence data, and does a quality control which includes a modular set of analyses. These modules gives the user a notion of whether they need to be aware of any problems in their data that requires to be handled before further analysis.

FastQC provides a quality control report where one can perceive if there

are problems produced by the sequencer or the raw data set. These reports are saved in an html file, and therefore able to be viewed in any browser.

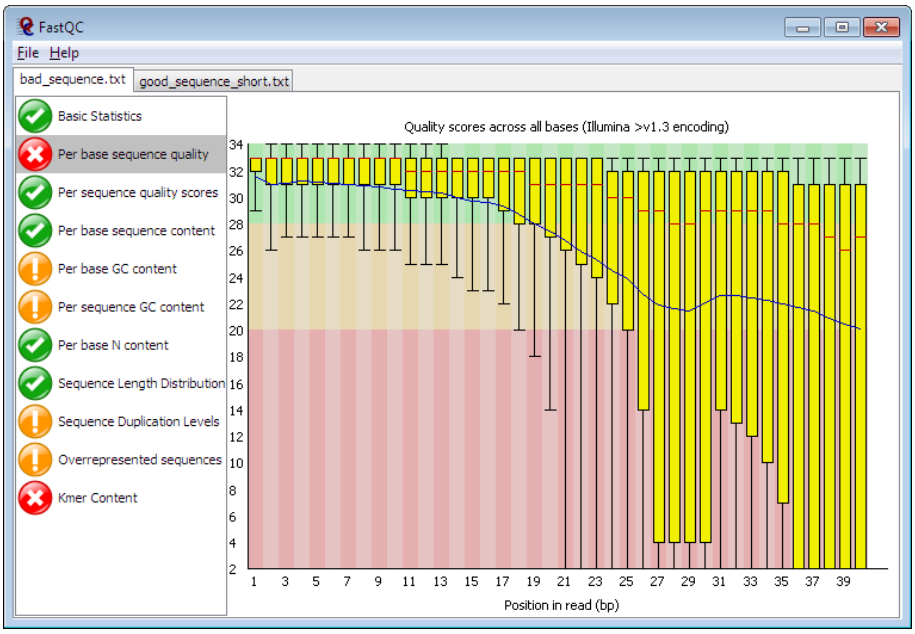

Figure 3.4: Quality control (example). The red line in the figure shows the median value, and the yellow boxes represents the inter-quartile range. The average quality score is on the x-axis and on the y-axis you have the number of sequences with that average. Figure from (*Babraham Bioinformatics - FastQC A Quality Control tool for High Throughput Sequence Data* 2021). The results of quality control in this thesis can be found in Section 5.1

Figure 3.4 is an example of the results from quality control using FastQC.

The rule in the Snakemake workflow for executing FastQC is written below.

Listing 3.1: Quality control

```
rule fastqc_all:
        input: expand("results-fastqc/{sample}_R1_fastqc.html", sample=samples),
               expand("results-fastqc/{sample}_R2_fastqc.html", sample=samples),
        output: "status/fastqc-complete"
        shell: "touch {output}" #Creat output file von output.
```

```
rule fastqc:
        input:
                "raw_fastq/{sample_full}.fastq.gz"
        output:
                html="results-fastqc/{sample_full}_fastqc.html",
                zip="results-fastqc/{sample_full}_fastqc.zip"
        params: ""
        log:
                "logs/fastqc/{sample_full}.log"
        threads: 1
        wrapper:
                "0.66.0/bio/fastqc"
```

There are four wrappers used in the Snakemake file, and each of them generates an environment using the workflow. Snakemake executes the first wrapper located in the script above, and downloads the corresponding wrapper files from GitHub. In the wrapper for quality control, i.e. *0.66.0/bio/-fastqc*, the version tag *0.66.0* can be replaced with an updated version (*The Snakemake Wrappers repository* 2016). The other three wrappers in the workflow are locates in the rule for trimmomatics, samtools-sort and samtools-index.

This script acquires the raw data set stored in the *raw_fastq* folder and analyse it with quality control. The results are accumulated in a folder called *results-fastq*, where a html-file (which can be viewed in any browser) and a zip-file is created. The zip-file contains a summary of the control, a detailed report of the raw data, and a folder where all the graphs that are presented in the html-file are stored. The results for our data sets are discussed in chapter 5.

3.3 Trimmomatic

Trimmomatic is a tool that can trim and crop fastq data and remove adapters. This is done by identifying the adapter sequences with two different ap-

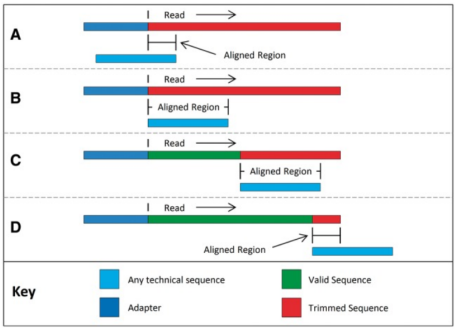

Figure 3.5: Simple mode. The initial sequence that is assessed by trimmomatic is depicted as the dark blue and red line. Figure from (Bolger AM 2014).

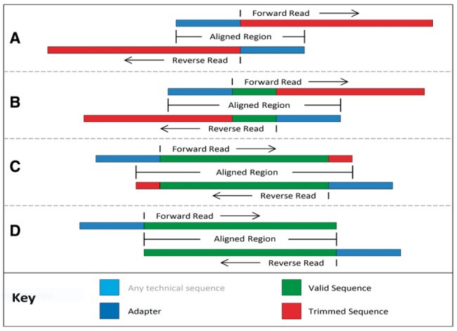

Figure 3.6: Palindrome mode. Using paired-end reads with adapters overlapping the reads, in the alignment process. Figure from (Bolger AM 2014).

proaches. The first approach, called *simple mode* (Figure 3.5), finds a rough match between the read and the sequence the user has provided. The part of the read that aligns to the provided sequence and the rest of the read after alignment are trimmed if the score of the alignment exceeds a user-defined threshold (Bolger 2021).

The second approach, *palindrome mode* (Figure 3.6), discern the adapter read-through scenario, where the ends of the reads are contaminated by adapters, caused by the DNA fragments being shorter than the read length. Palindrome mode identifies the valid sequence in each pair of read and the contaminated sequences, and globally align the sequences against each other. After the alignment, the read bases that match the adapters are trimmed. Since palindrome mode uses the characteristics of a sequence's reverse complements, this mode can only be used with paired-end data (Bolger AM 2014).

Trimmomatics is the second rule of the Snakemake workflow from figure 3.1. The *rule all* of trimmomatic is similar to the one used on quality control, only the folder where the results are to be placed is, needless to say, non-identical. The code is collected from the Snakemake file which can be found on GitHub.

Listing 3.2: Pseudo-rule for trimmomatics

```
rule trimmomatic_all:
        input: expand("trimmed/{sample}_R1.fastq.gz", sample=samples)
        output: "status/trimmomatic-complete"
        shell: "touch {output}"
```

The results from the adapter trimming are assigned to the folder called *trimmed*.

The next rule of the script takes the data from the *raw_fastq* folder, and trims the adapters:

Listing 3.3: Trimmomatics

```
rule trimmomatic_pe:
        input:
            r1="raw_fastq/{sample}_R1.fastq.gz",
            r2="raw_fastq/{sample}_R2.fastq.gz"
        output:
            r1="trimmed/{sample}_R1.fastq.gz",
            r2="trimmed/{sample}_R2.fastq.gz",
            # reads where trimming entirely removed the mate
            r1_unpaired="trimmed/{sample}_R1.unpaired.fastq.gz",
            r2_unpaired="trimmed/{sample}_R2.unpaired.fastq.gz"
        log:
            "logs/trimmomatic/{sample}.log"
        params:
            # list of trimmers (see manual)
            trimmer=["ILLUMINACLIP:/PATH/TO/Trimmomatic
            -0.38/adapters/TruSeq3-PE.fa:2:30:10:2:keepBothReads",
            "LEADING:0", "TRAILING:0", "SLIDINGWINDOW:4:15", "MINLEN:95"],
            # optional parameters
            extra="",
            compression_level="-9"
        threads:
            8
        wrapper:
            "0.66.0/bio/trimmomatic/pe"
```

The threads used in the script define the performance of the trimming, which can be user-defined or automatically determined. The */PATH/TO/TRIM-*

MOMATIC is where the trimmomatic-package is stored on the computer, which can be downloaded using Bioconda (*Trimmomatic* 2022). The full path used in this thesis can be found on GitHub.

The trimming status is written to a log file to adjust the use of processing parameters. The unpaired results are marked with *.unpaired*, and stored in the *trimmed* folder. For the parameters, *TrueSec3-PE.fa:2:30:10:2* removes any remaining adapters in the sequences, while storing both paired and unpaired reads. *SLIDINGWINDOW* finds low quality score sequence segments and removes them, and *MINLEN* is the minimum read length (Bolger AM 2014).

After the trimming, each sequence goes through another quality control (third step of figure 3.1), to see if there are any differences from the first control, while also looking for any errors from trimming. The rule for the second quality control is identical to the first, only the folders are adapted to store the results from quality control of trimmomatics (here: named *results-fastqc-trimmed*). In Section 5.1, the results from before and after trimming are explained, which shows the importance of adapter trimming.

3.4 STAR Indexing

Before the trimmed reads in the fastq files can be mapped to a genome, the genome index has to be created. The genome index allows detection of spliced reads due to its known splice-junctions from annotated gene models (Dobin 2022).

Indexing a genome only has to be done once, which makes the mapping of a genome very productive. By using the genome index, one can effectively search the genome for mismatches.

The code listed below is used to create the genome index needed for the STAR mapping.

Listing 3.4: Star Index

```
rule star_index:
        input:
            # provide your reference FASTA file
            fasta = "/PATH/TO/TASTA/FILE/Mus_musculus.GRCm38.dna.toplevel.fa",
            # provide your GTF file
            gtf = "/PATH/TO/GTF/FILE/Mus_musculus.GRCm38.101.gtf"
        output:
            directory("Genome-index")
        message:
            "Testing STAR index"
        shell:
            "STAR --runMode genomeGenerate --limitGenomeGenerateRAM 33524373088
            --runThreadN 12 --genomeDir Genome-index --genomeFastaFiles {input.fasta}
            --sjdbOverhang 99 --sjdbGTFfile {input.gtf}"
```

The script takes in a GTF-file and a fasta-file located on your computer. The paths used in this thesis can be found on GitHub. A GTF-file (Gene Transfer Format) is a file format that holds information about a genes structure. It includes one line per feature containing 9 columns of data, and track definition lines (*GFF/GTF File Format - Definition and supported options* 2021). The different features in the GTF-file contains the sequence name, a source column indicating where the annotations came from, a feature column, start and end coordinates, a score field with a degree of confidence in the feature's existence and coordinates, and a frame column indicating where the feature begins (*GTF2.2: A Gene Annotation Format (Revised Ensembl GTF)* 2021). A FASTA format represents peptide or nucleotide sequences, where amino acids or base pairs are represented with single-letter codes (*FASTA* 2022).

3.5 STAR Mapping

To get the distance between genes on a chromosome. and to identify the location of said genes, one must use genome mapping. Genome mapping

makes it easier to navigate around the genome, and to align DNA fragments to assist with the assembly of the genome. Spliced Transcripts Alignment to a Reference (STAR) is an algorithm used to align the reads in your data-set to a reference genome to find out where the reads are located in the genome (Dobin and Gingeras 2015). On account of the accessibility of STAR, it is implemented in this thesis. The algorithm consists of two steps, where the first step is *seed searching* and the second step is *clustering, stitching, and scoring*.

Seed searching is when STAR searches for a sequence that matches exactly to one or more locations on the reference genome. STAR searches for the longest matching sequences, also called Maximal Mappable Prefix (MMP), and stores this as a seed. Then the algorithm searches for the next longest sequence from the unmapped portion of the read, and stores this as another seed.

The seeds are then clustered together based on proximity to a set of anchor seeds, to form a complete read. Next, to get the best alignment for the read, the seeds are stitched together guided by a local alignment scoring scheme (*Introduction to RNA-Seq using high-performance computing - archived* 2021).

The code used for the STAR mapping is depicted below.

Listing 3.5: Star Mapping

```
rule star_pe_multi:
        input:
                fq1 = "trimmed/{sample}_R1.fastq.gz",
                fq2 = "trimmed/{sample}_R2.fastq.gz",
                index =  directory("Genome-index"),
        output:
                "STAR/{sample}/Aligned.sortedByCoord.out.bam"
        log:
                "logs/star/pe/{sample}.log"
        params:
                index = "/PATH/TO/Genome-index",
                extra = "--outSAMtype BAM SortedByCoordinate --twopassMode Basic
                --limitSjdbInsertNsj 1500000 --outSAMmultNmax 1 --quantMode
```

```
            TranscriptomeSAM GeneCounts --outFilterMatchNmin 95",
            prefix = "STAR/{sample}/",
    threads:
            8
    shell:
            "STAR {params.extra} --runThreadN {threads} \
                    --genomeDir {params.index} \
                    --readFilesIn {input.fq1} {input.fq2} \
                    --readFilesCommand zcat \
                    --outFileNamePrefix {params.prefix}"
```

This rule in the Snakemake workflow takes in the genome index created in Section 3.4 with the parameter *index* including the path to where the genome index was stored. The full path is available in on GitHub. The parameter *twopassMode Basic* takes the junctions from the first run and utilizes these as annotated junctions. The *outSAMmultNmax 1* parameter will give an output of one SAM line for each mapped read, *quantMode TranscriptomeSAM* will output alignments translated into transcript coordinates, and *outFilterMatchNmin 95* gives the output if the amount of matched bases is higher than 95. *limitSjdbInsertNsj* is the maximum number of junctions inserted to the genome, and *SortedByCoordinate* is similar to Samtools Sort (described in Section 3.6.2), which sorts the outputs by coordinate (Dobin 2022).

3.6 Samtools

Samtools is a program that interacts with high-throughput sequencing data, and needs to be deployed before running rMATS on the data set. The purpose of this is to convert the binary alignments in the bam-files into text-based sam-files (Danecek et al. 2021). Samtools can be used in several different interactions, but for the purpose of this thesis I will only mention *samtools sort* and *samtools index*.

3.6.1 Samtools Index

Samtools index indexes a coordinate-sorted BGZIP-compressed bam-file for fast access, and stores the results in *samtools-index*.

Listing 3.6: Samtools Index

```
rule samtools_index_all:
        input: expand("samtools-index/{part}.bam.bai", part=parts),

rule samtools_index:
        input: "samtools-sort/{part}.bam"
        output:
                "samtools-index/{part}.bam.bai"
        params:
                ""
        wrapper:
                "0.74.0/bio/samtools/index"
```

The files are in the BAI index format, which can handle up to 512 Mbp individual chromosomes in length (H. Li 2021a).

3.6.2 Samtools Sort

Samtools sort sorts the alignments and stores the output in a specified file (H. Li 2021b).

Listing 3.7: Samtools Sort

```
rule samtools_sort_all:
        input: expand("samtools-sort/{part}.bam", part=parts),

rule samtools_sort:
        input: get_matches
        output:
                "samtools-sort/{part}.bam"
        params:
                extra = "-m 4G",
                tmp_dir = "/tmp/"
        threads:
                8
        wrapper:
                "0.77.0/bio/samtools/sort"
```

To get the bam-files required for executing samtools, the script *get_matches* collects the results from Section 3.5 that includes files named *Aligned.sortedBy*

Coord. These files are stored in a list before they are used by samtools for sorting and indexing. The results from samtools sort are stored in the folder *samtools-sort*, and utilized later by samtools index.

Listing 3.8: Search for the required files

```
def get_matches(part):

        list = []

        for i in reps:
                star_file = f"STAR/{part}_{i}/Aligned.sortedByCoord.out.bam"
                raw_data = f"raw_fastq/{part}_{i}_R1.fastq.gz"

                if Path(raw_data).exists():
                        list.append(star_file)
        return list
```

The script above creates an empty list, and searches through the results from STAR for the required bam.file. From there on it goes through the raw data to see if a fastq-file corresponds to the name on the bam-file. If the names are similar, the bam-file is stored in the list.

3.7 rMATS

rMATS detects alternative splicing events in RNA-seq data, and is the last step of the Snakemake workflow. The absolute path to the results from samtools-sort is required to execute rMATS. These paths are collected and stored in a text file with the script below.

Listing 3.9: Get absolute path of the files

```
rule PRPF31_file:
    input:
        expand("samtools-sort/{part}.bam", part=parts)
    output:
        "samtools-sort/{group}.txt"
    log: "logs/PRPF31-create-{group}.txt"
    run: absolute_path(wildcards.group)

def absolute_path(group):
```

```
    list = []
    for fname in parts: #Go through the files in samtools-sort
        print(fname)
        print(group)
        if (str(group) == fname[3:5]): #check if the file ends with correct characters
            p = os.path.abspath(f"samtools-sort/{fname}.bam") #get absolute path of file
            list.append(p)
            print(list)
    file_paths = ','.join(list)
    write_path = f"samtools-sort/{group}.txt"
    print(f"write_path = {write_path}")
    with open(write_path, "w") as file:
        file.write(file_paths) #write the paths to text file
    return file_paths
```

The script for the absolute paths are executed with *rule PRPF31_file*, which attains the bam-files in the result folder of samtools-sort. The absolute paths are included in the text file if the filenames of the results ends with the appropriate characters.

As there are two main differences of the filenames in this thesis, i.e. *PRPF31* for the treatment group and *SCR* for the control group, the script creates one file for each group.

Following in the Snakemake workflow, the rule for rMATS utilizes the results gathered from the scripts above and stores the results in the *results-rmats* directory.

Listing 3.10: rMATS

```
rule rmats:
    input:
        gtf = "/PATH/TO/GTF/FILE/Mus_musculus.GRCm38.101.gtf",
        b1 = expand("samtools-sort/{group}.txt",group=["PRPF31","SCR"]),
        hups = expand("samtools-index/{part}.bam.bai", part=parts),
        #b2 = "samtools-sort/{group}.txt",
    output:
        od = directory("results-rmats"),
        tmp = temp(directory("temp-rmats")),#with temp() the directories
        will deleted afterwards
        flag = temp(touch("results-rmats/flag"))#Snakemake wants for every
        input an output file. As we create only directories, we fake
        snakemake with a fake ouptut-file using variables (group).
    params:
        # optional parameters
        extra="-t paired --readLength 95",
    threads: 24
    log: "logs/rmats_logfile_{group}.txt"
```

```
    shell:
        "rmats-corrected.py --b1 {input.b1[0]} --b2 {input.b1[1]} --gtf {input.gtf}
        {params.extra} --nthread {threads} --od {output.od} --tmp
        {output.tmp} > {log} 2>&1"
```

The rule for rMATS obtains the gtf-file as input, including the path to where the file is located on your computer, as well as the files with the absolute paths and the results from *samtools-index* (Section 3.6.1). The script uses temporary directories to store data while running, and these are deleted after the workflow is completed to reduce the amount of used storage space. As commented in the script, *flag* is used as an output-file for Snakemake, however the outputs in the workflow are directories and will not be placed in *flag*.

The shell command specifies the different inputs, number of threads per inch, output-files, and the temporary directories.

3.8 Pseudo-rules for executing the Snakemake workflow

For every rule in the Snakemake workflow, a psudo-rule is required for the execution of the rule. These are written at the beginning of the Snakemake file, and Snakemake moves through every pseudo-rule chronologically.

Listing 3.11: Pseudo-rules

```
rule all:
        input:
             "status/fastqc-complete",
             "status/trimmomatic-complete",
             "status/fastqc-trim-complete",
             "Genome-index",
             expand("STAR/{sample}/Aligned.sortedByCoord.out.bam", sample=samples),
             expand("samtools-sort/{part}.bam", part=parts),
             expand("samtools-index/{part}.bam.bai", part=parts),
             expand("samtools-sort/{group}.txt",group=["PRPF31","SCR"]),
             "results-rmats/flag",
```

Each pseudo-rule points to a rule in the Snakemake workflow, i.e. *status/fastqc-complete* points to *rule fastqc_all* from Section 3.2. With Snakemake it is effortless to run the Snakemake file and all the rules will be executed in chronological order. If one should however, want to run only one rule at a time, it is possible to ad # at the beginning of the pseudo-rule you don't want executed.

Further, Chapter 4 and 5 presents the results of the Snakemake workflow and the results of the case study.

Chapter 4

Results of Snakemake

In this chapter, the adaptation of the Snakemake protocol to the case-control study of retinitis pigmentosa is discussed, as well as the obtained results.

The results of Snakemake consists of the rules created for the protocol (Figure 3.1), including quality control, trimmomatics, STAR indexing and mapping, samtools and rMATS. By creating a Snakemake document that includes every step needed for obtaining alternative splicing events, the reproducibility, adaptability and transparency is improved for further research on alternative splicing using rMATS.

Every rule created for the Snakemake workflow in this thesis can be found on *https://github.com/MadelenH/Snakemake*. This repository includes separate files for each of the rules in the workflow, as well as the complete Snakemake file. This enables the user to collect and use any part of the workflow as seen fit, or execute the entire workflow in its whole. For instance, if someone only needs to trim the adapters of the sequences in their raw data, they can retrieve the file with the rule of trimmomatics from the repository (Figure 4.1).

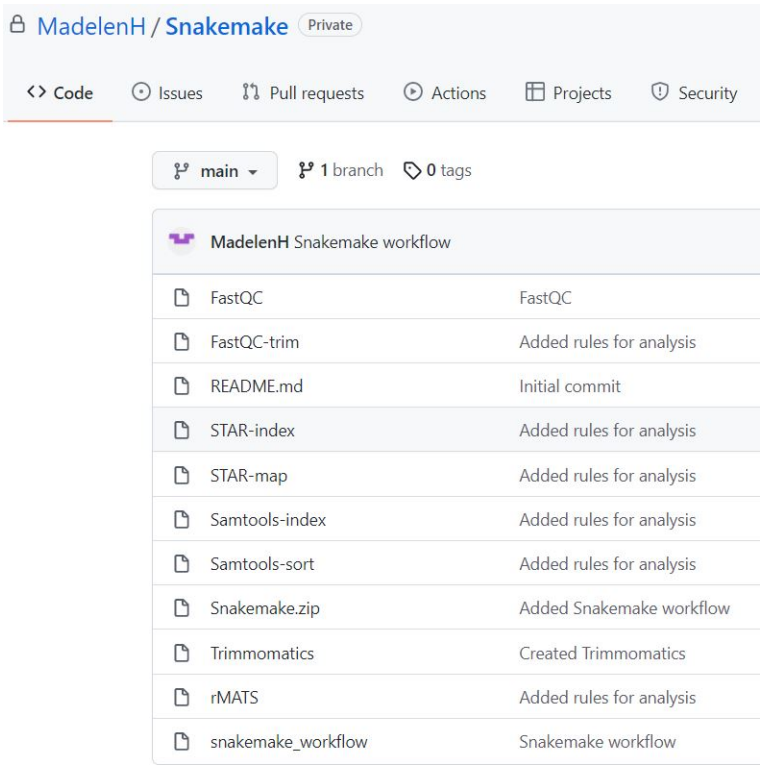

Figure 4.1: The repository on GitHub including the rules for executing alternative splicing analyses.

Using GitHub as a directory for the results of Snakemake, makes the workflow accessible for everyone to use and implement to their own data-set. It is also a free platform which can be used for any sort of document. Whether you use GitHub for one or several documents, it is without charge. GitHub encourages collaboration and communications between users, making it easy to work on projects with others (*GitHub* 2022).

Chapter 5

Results of Case Study

This chapter contains the results from the Snakemake workflow on the case study. Section 5.1 shows the results from the quality control before and after trimmomatics. Section 5.2 describes the results of STAR and samtools. Section 5.3 will be a walk-through of the results from rMATS.

5.1 Quality control before and after trimmomatics

Throughout this section, I will look at the results of the quality control for sequence P18_PRPF31, as well as compare the results of quality before and after trimming. The first graph in the html-report created by the script in Section 3.2 presents an overview of the raw data used in this thesis (Figure 5.1).

Basic Statistics

Measure	Value
Filename	P18_PRPF31_R1.fastq.gz
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	49502682
Sequences flagged as poor quality	0
Sequence length	151
%GC	51

Figure 5.1: The basic statistics of the raw data. The graph shows the total amount and length of the sequences.

The sequences have good quality, as there were no instances where they were flagged as poor quality in Figure 5.1. Every sequence in the raw data had the length of 151 base pairs before trimming. After adapter trimming (Section 3.3), the sequence length varied from 95-151 base pairs due to the removal of the adapters which can consist of several base pairs or none at all.

The next graph in the report illustrates the per base sequence quality scores. To get a better understanding of trimmomatics, I will compare the quality of the sequences before (Figure 5.2) and after (Figure 5.3) trimming. The plots below yields the distribution of quality scores at each position across all bases in the reads. As discovered in Figure 5.2, the quality score of the sequences before trimming is acceptable due to the absence of interquartile range results (i.e. the yellow boxes) below the red (or yellow) line. None of the sequences in the raw data had poor quality on the per base sequence quality scores.

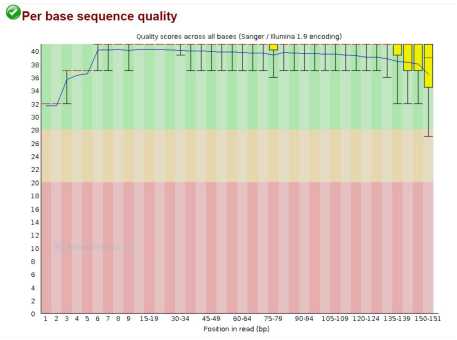

Figure 5.2: Per base sequence quality score at each position of the read in the raw data.

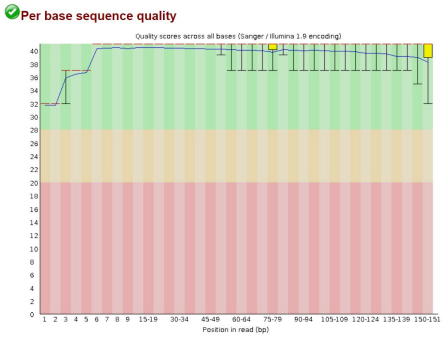

Figure 5.3: Per base sequence quality score after trimming the adapters.

Regarding Figure 5.3, it is noticeable that a few changes has occurred after trimming. The blue line running across the plot, which represents the mean quality, remains essentially the same before and after trimming. The only difference is that the mean has some higher score at the last positions in the fastq file after trimmomatics (see elevated blue line in Figure 5.3). This can additionally be seen by the elevated values of the interquartile range, meaning the quality scores are improved after trimming. (*Per Base Sequence Quality* 2021)

The report from the quality control includes a graph with *Overrepresented sequences* that are occurring in the data-set before the adapter trimming (Figure 5.4). The sequences were corrected by trimmomatics, which in turn improved the quality score.

Figure 5.4: Overrepresented sequences reveals the sequences that occur more than expected in the file. The sequence is overrepresented if it accounts for more than 0.1 % of the total reads (*Overrepresented Sequences* 2021).

Figure 5.4 identifies contamination in the reads, such as vector sequences or adapter sequences. One overrepresented sequence is detected in the report of the raw data from Figure 5.1. The sequence accounts for more than 0.1 percentage of the total reads, and the report indicates the possible source of the overrepresentation is an adapter.

To correct this error, trimmomatic identifies the adapters and removes the contamination. In every case where an overrepresented sequence occurred in the report, it was removed during the adapter trimming.

The per tile sequence quality graph (Figure 5.5) identifies the loss in quality associated with a part of the flowcell. This graph can only be assembled if the library used is Illumina. The graph can give warnings or errors if there are bubbles going through the flowcell, or if there are smudges or debris on the flowcell (*Per Tile Sequence Quality* 2021). This indicates that a warning or an error might not be a result of loss in quality.

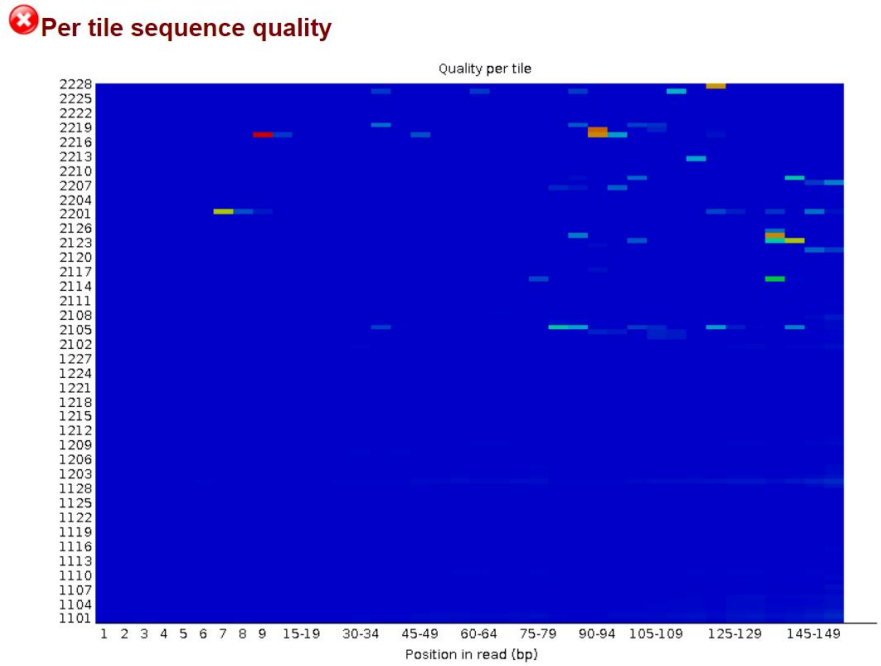

Figure 5.5: Per tile sequence quality shows the quality of each tile and searches for any loss associated with one part of the flow cell (*Per Tile Sequence Quality* 2021). The quality remained the same before and after trimming.

The red and orange colours in figure 5.5 indicated the positions where the quality for that base was low. The blue colours indicated sufficient quality for that position of the base. The plots show the same results before and after the trimming.

Figure 5.6 depicts the per sequence quality score. These results were approximately the same before and after trimming.

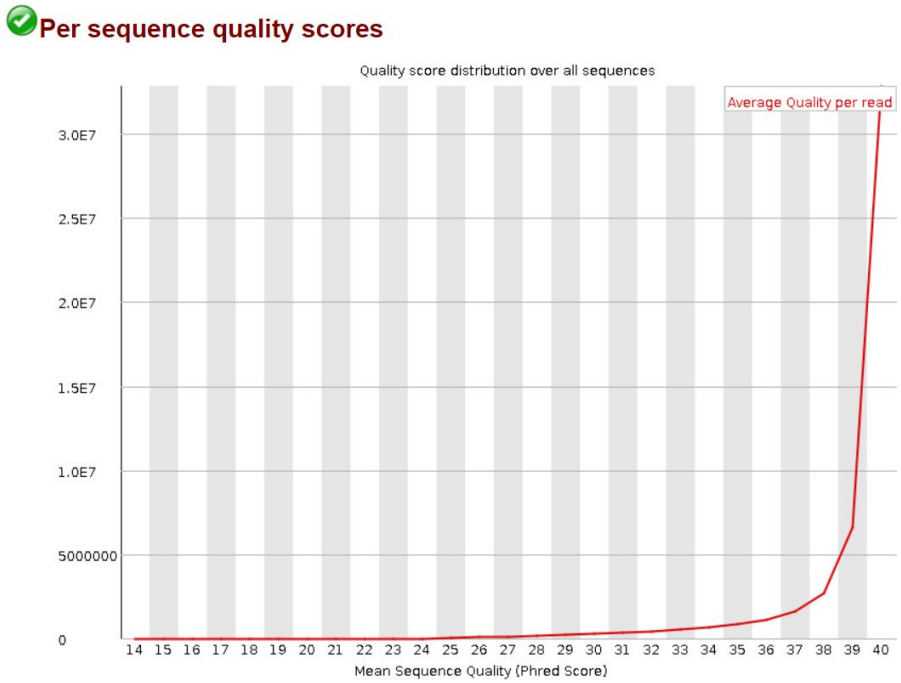

Figure 5.6: Per sequence quality score gives a representation of the average quality score (x-axis), and the amount of sequences with that score (y-axis). This graph is collected from the results before trimming (*Per Sequence Quality Scores* 2021).

The graph in Figure 5.6 remains stable on the lower quality values, which are ideal results of the reads. Had the graph viewed an increase in the lower sequences, the reads causing that result would have to be trimmed with trimmomatics. After trimming, the peak of the graph was shifted a small fraction to the right. This signifies that more sequences have higher quality score after the adapter trimming. (*Per Sequence Quality Scores* 2021)

The per base sequence content graph (Figure 5.7) in the report illustrates the proportion of each nucleotide position in the file, and the per sequence GC content graph (Figure 5.8) illustrates the guanin-cystine conten. The results for both of the graphs remains the same after trimming.

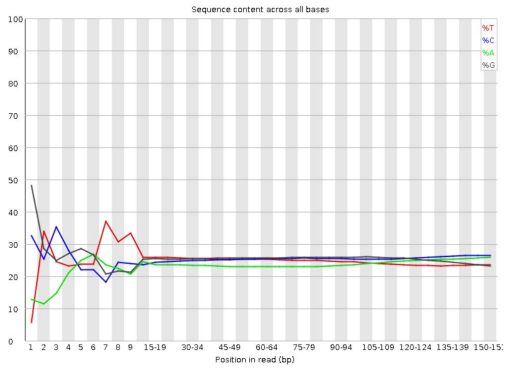

Figure 5.7: Per base sequence content before trimming.

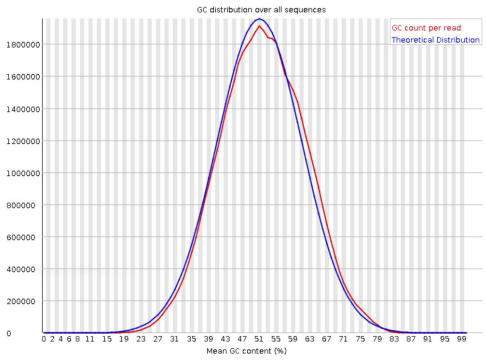

Figure 5.8: Per sequence GC content before trimming.

In figure 5.7 there is a biased selection of random primers at the first 12 base pairs, which creates a greater than 20% difference between the nucleotides A and T (red and green line) and G and C (black and blue line). The difference between A and T at the first base pairs is evident and results in an error in the QC-report, however it is not solvable with trimmomatics (*Per Base Sequence Content* 2021).

Figure 5.8 illustrates the GC content, which measures the guanin-cystine content of each sequence and compares to a theoretical distribution. As the graph illustrates, the peak of the GC content corresponds to theoretical distribution of the underlying genome (*Per Sequence GC Content* 2021).

Figure 5.9 represents the distribution of fragment sizes in the analyzed file (*Sequence Length Distribution* 2021). Before trimming the sequences had a uniform length of 151 base pairs, but after adapter trimming the length differs as illustrated in Figure 5.9. This complements the results collected in Figure 5.1, where the sequence length differs between 95 and 151 base pairs after trimming the adapters.

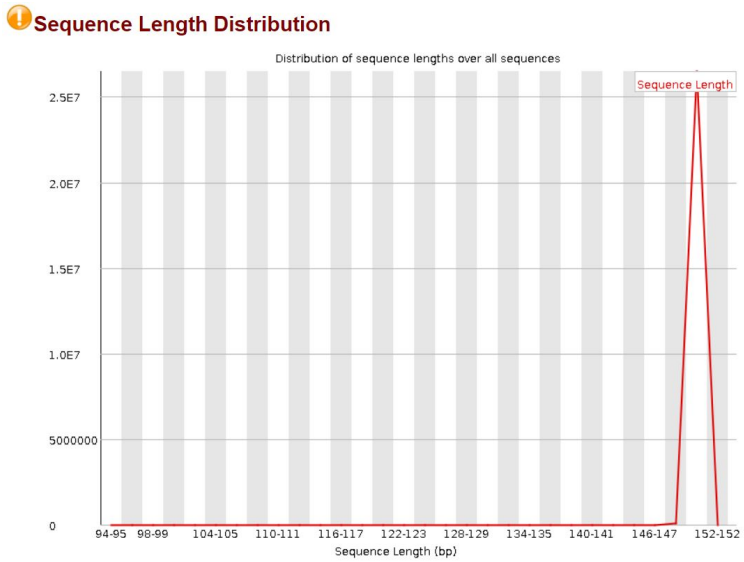

Figure 5.9: Sequence length distribution after adapter trimming.

The graph with sequence duplication levels (Figure 5.10) shows the amount of sequences containing different percentages of duplicates by counting the duplicates for every sequence in the file (*Duplicate Sequences* 2021).

The peaks in the plot at duplication level 10 represents low complexity contaminants, and the elevated lines in the far left represents a properly diverse library where most sequences only occur once in the final set. The

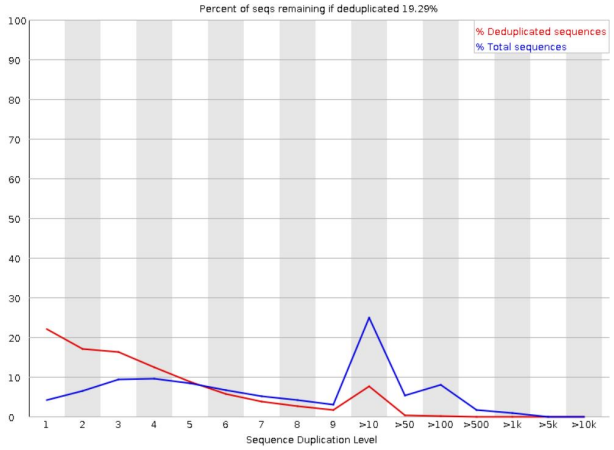

Figure 5.10: Sequence duplication levels after trimming.

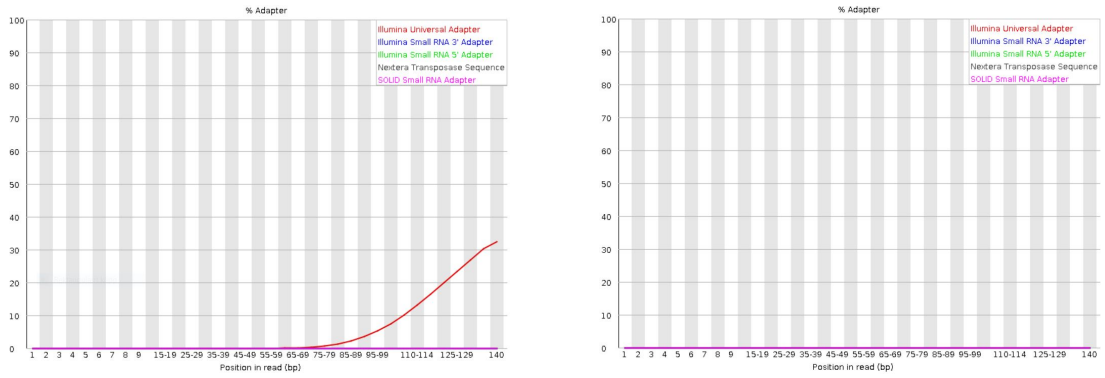

Figure 5.11: Adapter content before (left) and after (right) trimming.

graph illustrates that approximately 25% of the sequences have 10 duplicates, and less than 10% have 50 to 100 duplicates. The levels of duplicates does not change significantly after trimming. At the top of the graph, a percentage of sequences remaining if the files were deduplicated are shown. For the data-set used in this thesis, 19.29% of the sequences remains if they were deduplicated (*Duplicate Sequences* 2021). The important section of this graph is located at the far left, where we can see that 5% of the total amount of sequences have no duplicates (blue line). Duplicates arise from sequencing two or more copies of the same DNA fragment, but previous research (Ebbert et al. 2016) has shown that removing these does not effect the accuracy of identifying single nucleotide polymorphisms.

The graphs in Figure 5.11 illustrates the adapter content before and after trimming. The left graph in Figure 5.11 shows the percentage of the reads with each of the adapters listed in the box at the right corner. As illustrated in the graph from approximately 70 base pairs to 140 base pairs, the Illumina adapter is present. The quality control report issued a warning at the adapter content before trimming, meaning that that this adapter is present in more than 10% of the reads. This is also illustrated by the curve in the graph, which

shows the Illumina adapter exceeds 30% of the reads. (*Adapter Content* 2021)

The graph on the right hand side in Figure 5.11 depicts the result of the adapter content after trimming the adapters. Here we can see that the Illumina adapter is no longer present, which gives the report a good result.

5.2 STAR and samtools results

The genome index from Section 3.4 is built to run STAR on the quality controlled sequences. Indexing assists the process of effectively locating the origin of a sequence by narrowing down the sequence query in the genome (Trapnell and Salzberg 2009). The genome index contains several text-files including character length and name, and the genome parameters with paths to the fasta and gtf-file located on your computer.

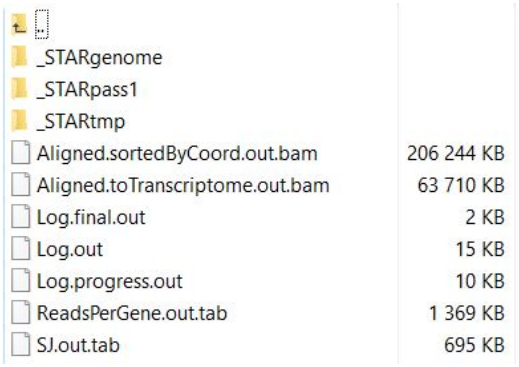

Figure 5.12: Results from mapping with STAR

Read alignment with STAR can be executed when the genome index is built, resulting in rapid search against the large reference genomes. When alignment with STAR is completed, the files in Figure 5.12 are generated for each read.

The first folder in Figure 5.12 (i.e. *_STARgenome*) contains two files; *sjdbList.out.tab* and *sjdbInfo.txt*. The first file includes coordinates of each

of the introns, and the second file is a local file for STAR. The second folder, *_STARpass1* contains a *Log.final.out* document and an *SJ.out.tab* document. *Log.final.out* contains the summary of the mapping statistics, while *SJ.out.tab* contains the sum of the counts of each splicing (Dobin 2022).

Aligned.sortedByCoord.out.bam is a file where the output is sorted by coordinate, and *Aligned.toTranscriptome* contains output alignments translated into transcript coordinates. *Log.out* contains the run-time messages what can be useful for debugging. *Log.progress.out* stores the progress statistics, and *ReadsPerGene.out.tab* contains read counts per gene (Dobin 2022).

Samtools sort selects alignments from the STAR output by using the script in Section 3.6.2 and stores the sorted files as .bam-files. Every file is sorted to efficiently access the data that is stored in the bam-files. Samtools sort is applied to *Aligned.sortedByCoord.out.bam*-files to sort them for effective access to the data. When the alignments are sorted and stored in bam-files, they are indexed using the script in Section 3.6.1. This is done so the genomic regions with overlapping alignments can be extracted. This creates files which will be used later in rMATS.

5.3 Results from rMATS

This section describes the results received from the script of rMATS (Section 3.7), which are used for estimation of the different proportions of the isoforms using reads mapped to different isoforms (Shen, Park, Lu, et al. 2014). This section includes a description of the different outputs from rMATS, as well as their purpose.

To run the rMATS script, we need to get the absolute path to the bam-files created with samtools-sort and index in separate text-files. Using the

script in Section 3.7 labeled *absolute_path*, the path to the bam-files are found and stored. These files contains the absolute paths to the PRPF31 and the SCR sequences separated.

Using the results from the absolute path-script, the results of rMATS are generated and stored in a folder named *results_rmats*, but the script also creates a temporary folder, *temp-rmats*, which will be deleted after the run is completed. The results-folder contains files with alternative 3' and 5' splice sites, novel junctions, novel splice sites and count files. Each alternative splicing event has a set of output files that corresponds to the events. (Figure 5.13)

A3SS.MATS.JC.txt	33 KB	24.03.2021 14:33:38
A3SS.MATS.JCEC.txt	35 KB	24.03.2021 14:33:40
A5SS.MATS.JC.txt	16 KB	24.03.2021 14:33:40
A5SS.MATS.JCEC.txt	18 KB	24.03.2021 14:33:41
fromGTF.A3SS.txt	508 KB	24.03.2021 14:33:30
fromGTF.A5SS.txt	289 KB	24.03.2021 14:33:30
fromGTF.MXE.txt	84 KB	24.03.2021 14:33:30
fromGTF.novelJunction.A3SS.txt	3 KB	24.03.2021 14:33:30
fromGTF.novelJunction.A5SS.txt	1 KB	24.03.2021 14:33:30
fromGTF.novelJunction.MXE.txt	2 KB	24.03.2021 14:33:30
fromGTF.novelJunction.RI.txt	1 KB	24.03.2021 14:33:30
fromGTF.novelJunction.SE.txt	28 KB	24.03.2021 14:33:30
fromGTF.novelSpliceSite.A3SS.txt	1 KB	24.03.2021 14:33:30
fromGTF.novelSpliceSite.A5SS.txt	1 KB	24.03.2021 14:33:30
fromGTF.novelSpliceSite.MXE.txt	1 KB	24.03.2021 14:33:30
fromGTF.novelSpliceSite.RI.txt	1 KB	24.03.2021 14:33:30
fromGTF.novelSpliceSite.SE.txt	1 KB	24.03.2021 14:33:30
fromGTF.RI.txt	390 KB	24.03.2021 14:33:30
fromGTF.SE.txt	1 847 KB	24.03.2021 14:33:30
JC.raw.input.A3SS.txt	7 KB	24.03.2021 14:33:38
JC.raw.input.A5SS.txt	4 KB	24.03.2021 14:33:40
JC.raw.input.MXE.txt	2 KB	24.03.2021 14:33:36
JC.raw.input.RI.txt	10 KB	24.03.2021 14:33:42
JC.raw.input.SE.txt	23 KB	24.03.2021 14:33:32
JCEC.raw.input.A3SS.txt	8 KB	24.03.2021 14:33:39
JCEC.raw.input.A5SS.txt	4 KB	24.03.2021 14:33:41
JCEC.raw.input.MXE.txt	3 KB	24.03.2021 14:33:37
JCEC.raw.input.RI.txt	13 KB	24.03.2021 14:33:43
JCEC.raw.input.SE.txt	25 KB	24.03.2021 14:33:35
MXE.MATS.JC.txt	11 KB	24.03.2021 14:33:37
MXE.MATS.JCEC.txt	12 KB	24.03.2021 14:33:37

Figure 5.13: Data structure containing the results from rMATS.

From Figure 5.13, the file *A3SS.MATS.JC.txt* shows the final output of reads with the full extent of junctions. Here, *JC* stands for Junction Counts. The file where *JC* is replaced with *JCEC*, both reads with the full extent of

junctions and reads that don't cross an exon boundary are included, which gives us both Junction Counts and Exon Counts. *fromGTF.A3SS.txt* is the output file of all alternative 3' splice site events acquired from RNA and the GTF-file. We can also see that the name of the files change according to which alternative splicing event was identified.

Continuing in the results-folder, the file *fromGTF.novelJunction.A3SS.txt* contains the 3' splice sites identified after considering the RNA, and includes annotated splice sites. File *fromGTF.novelSpliceSite* contains the events where novel splice sites are present. *JC.raw.input.A3SS* includes the count of 3' splice site events with the full extent of junctions defined by rMATS, while *JCEC.raw.input.A3SS* includes the full extent of junctions defined by rMATS and reads that don't cross an exon boundary.

The different abbreviations in the file-names stand for the different alternative splicing events. As described above, *A3SS* is alternative 3' splice site. *A5SS* stands for alternative 5' splice site, *SE* is skipped exon, *MXE* is mutually exclusive exons, and *RI* stands for retained intron.

After running rMATS in Snakemake, the terminal gives a summary of the different splicing events present (Figure 5.14). The summary informs us that there are 1,986 exon skipping events and 838 mutually exclusive exon events. There are 8,388 alternative splice site events, where 5,339 of these are 3' splice site events, and the rest are 5' splice site events. We can also see that there are 4,107 retained intron events.

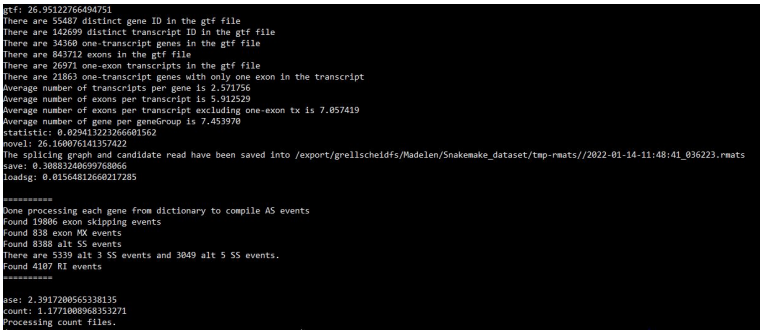

Figure 5.14: Summary of the different alternative splicing events occurring in the raw data. This summary contains the number of exon skipping events, mutually exclusive exons, alternative 3' and 5' splice sites and intron retention.

Considering the files created after running rMATS, several splicing events were found. Looking through the files including skipped exon events, the different columns in the output file gives information about the exons on each row. Each row contains the ID of the gene, the gene symbol, the position of the nucleotide that is at the upstream end or downstream end of the exons. It also includes the number of reads for each sample where the exon is included in the transcript and where it got skipped.

In the file with the alternative 3' splice site, the first columns are the same as for skipped exons. The following columns shows the start-base and end-base of the long and short exon, as well as the flanking exon start and exon end. The next columns includes the p-value as well as the amount of exons included from the treatment and the control group in the transcript.

The file with the alternative 5' splice sites includes the same columns as the 3' splice site-file. However, in the file with intron retention the columns have some differences. The first columns are the same as in every alternative splicing event. Following, we have the position of the nucleotide at the

upstream and downstream end of the exon. This is similar to the columns in the files with skipped exons, where the upstream exon occurs toward the 5' end from the transcription site and the downstream exon occurs towards the 3' end (*Terminology of Molecular Biology for Upstream/Downstream* 2022) (Figure 5.15).

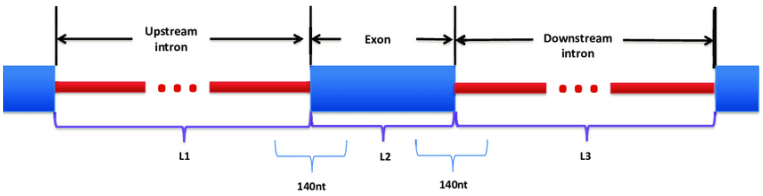

Figure 5.15: Illustration of the upstream and downstream end of the exon. Figure from (Louadi et al. 2019).

Figure 5.15 explains the position of the upstream and downstream end of an exon. These positions are written in the text-file, for both start and end of the upstream and downstream exon.

The next columns in the file shows the number of reads of the sample with the treatment group that were assigned to alternative splicing events where the exon was included, as well as the number of reads from the control group where the exon was included. We also get the information of where the exons were skipped for both the treatment group and the control group. Further, we get the level of difference between the amount of exons included from the control group and the treatment group, which is stored under the column marked *IncLevelDifference*.

In Chapter 6, I will focus on the genes that can be associated with retinitis pigmentosa, as well as genes that occur in several alternative splicing events. I will also look at how the Snakemake workflow has optimized the protocol for finding alternative splicing events in genes.

Chapter 6

Discussion and Conclusion

Analyzing alternative splicing events requires the RNA-Seq technology that gives the best results with low cost and which uses less time. Using rMATS to analyze the splicing events makes the process both cost and time efficient. With the Snakemake workflow, the results of the alternative splicing events are easily attained, and the results of rMATS are straightforward and easy to interpret.

In this chapter, I will give my interpretation of the results from Chapter 4 and 5, summarize the key findings and discuss the implications of the results. I will discuss the results from the entire protocol, but my main focus is the results from rMATS.

The results from Chapter 4 informs that the rules required for alternative splicing analysis are implemented to the Snakemake workflow. This enables the user to adapt every rule to their own repositories, and facilitates further research on alternative splicing.

The results from Section 5.1 indicate that the sequences used in this paper have a good quality after running trimmomatics, and that running a quality control on the sequences before the rest of the protocol is essential, due

to the large impact it will have on the results later on. STAR and samtools from Section 5.2 gives the necessary information to use for executing rMATS, which includes the absolute path to the different sequences and a star index to map reads. Mapping sequences to a reference genome allows the user to get the specific location of the different reads and the expression level of genes (Dobin 2022).

The results from rMATS gives - as explained in Section 5.3 - several folders containing a respective amount of genes and their position in the reads. In the following sections (Section 6.1, 6.2, 6.3 and 6.4) I will focus on a few genes and discuss the information the folders provides. For analysis of these genes, I have used the threshold of $p-value < 0.05$ and $IncLevelDifference > 2.5\%$ to analyse genes with significant values.

Further, I will present possible future work (Section 6.6) and a table containing the genes filtered with the threshold $p-value < 0.05$ and $IncLevelDifference < -0.1$ or $IncLevelDifference > 0.1$.

6.1 PRPF31 mutations causing retinitis pigmentosa

The function of PRPF31 is that it codes for the spliceosome hPRP31, which removes intron sequences from pre-mRNA. PRPF31 plays an important part in the spliceosome, as it is required for the construction of the RNA-protein sub-units U4/U5/U6 tri-snRNP complex which is one of the building blocks of the spliceosome (Makarova et al. 2002). The functioning of PRPF31 is important as the main amount of pre-mRNA introns are spliced by the major spliceosome. When mutations occurs in the PRPF31 gene the photoreceptors are degenerated due to haploinsufficiency, which means that one copy of the

gene is inactivated and the other copy is inadequate to produce the protein (Gregory-Evans, Pennesi, and Weleber 2013).

Mutations of the gene PRPF31 is the most common cause of retinitis pigmentosa, so it is included in this thesis due to its relation to the disease. Previous studies on mutations of PRPF31, show that mutated PRPF31 have high levels of transcripts with retained intron and alternative 3' splice sites (Buskin et al. 2018). I could however not find any alternative splicing events of the gene PRPF31.

A possible explanation for this is that the mutations of the gene from the treatment group shows the non-penetrance form of RP (Rose and Bhattacharya 2016). This form of RP is described as *variant haploinsufficiency*. This means that disease can in some cases be produced with the absence of a second wild-type PRPF31, and in some cases not. This depends on the nature of the wild-type allele and the nature of the mutant allele inherited (Wheway et al. 2020).

6.2 MTF2 is a potential novel gene of interest for retinitis pigmentosa

This section includes the results for the gene MTF2, where the majority of the detected alternative splicing events were located in the control group.

Metal Response Element Binding Transcription Factor 2 (MTF2) is a protein coding gene that facilitates transcription corepressor and methylated histone binding activity (*MTf2 metal response element binding transription factor 2 [Homo sapiens (human)]* 2022). MTF2 can also be associated with the disease retinitis pigmentosa.

This gene occurred in both mutually exclusive exons (MXE) and retained

intron (RI). The values of included junction counts (IJCs) of the treatment group in MXE and RI were *7,2,2* and *4,1,13*, while the skipped junction counts (SJCs) were *9,2,3* and *0,1,10*. For the control group the, the IJCs were *28,9,7* and *6,10,5*, while for SJCs the values were *9,5,3* and *0,0,0*. This concludes that for both the treatment group and the control group, the respective exons were included. The p-value for MTF2 in MXE was *0.01185* and the *IncLevelDifference* was *-0.26*, while for RI the p-value was *0.00000768* and the *IncLevelDifference* was *-0.367*. In both instances of the gene the *IncLevelDifference* is significantly less than 0, meaning more exons from the control group are included in the transcript than from the treatment group.

The results of MTF2 shows that the majority of alternative splicing events were located in the genes from the control group. This implies that in the treatment group the exons with splicing events were skipped, which could be correlated to retinitis pigmentosa. However, I could not find any previous research done on alternative splicing events in MTF2, meaning it could be a possibility to focus on in further research.

6.3 The same gene can have multiple different types of splicing events

This section presents the gene KDM6A, which included several different splicing events from the results of rMATS.

Lysine Demethylase 6A (KDM6A) is a protein coding gene that provides instructions for creating lysine-specific demethylase 6A (*KDM6A gene* 2020). Lysine-specific demethylase 6A is an enzyme that functions as a histone demethylase, which is involved in transcription (Kooistra and Helin 2012).

Mutations in the gene KDM6A can results in cancer, and it can be associated with the Kabuki Syndrome (Miyake et al. 2012).

KDM6A occurred once in the folder with MXE events, and twice in the folder containing skipped exon events (SE). The legnth of the first exon in MXE was 121 base pairs, and the second exon was only 55 base pairs. The first instance of KDM6A in SE had the length of 156 base pairs, while the second had 121 base pairs.

The IJCs of the treatment group in MXE and SE was *5,0,2*, *14.12,3* and *16,10,4*, while the SJCs where *0,2,4*, *0,1,4* and *0,1,4*. From the result of the control group, the IJCs were *20,3,1*, *0,14,9* and *29,3,3* with the SJC values *0,0,0*, *2,7,4* and *0,0,0*. Here we can see that the IJCs have higher values than the SJC, meaning the exons with the alternative splicing events were included in the transcript.

The p-value for the gene in MXE was *0.0001395* and the *IncLevelDifference* was *-0.575*. In SE the p-value for the genes were *0.0450* and *0.0113*, and the *IncLevelDifference* were *0.366* and *-0.263*. The gene from MXE and one of the genes in SE has significantly low values of *IncLevelDifference*, meaning the majority of exons included are from the control group. However, the second gene in SE has a significantly high value of *IncLevelDifference*, which means a substantial amount of skipped exon events can be found in both the treatment group and the control group of KDM6A.

A study on mutations and splicing events in lung cancer (Liu et al. 2012), shows that mutations of the KDM6A gene decreases the gene copies in the cell line resulting in a nonfunctional demethylase 6A enzyme. This implies that mutations of KDM6A disrupts the regulation of genes, leading to uncontrolled cell division and the development of cancer.

6.4 Considering known genes affected by PRPF31

After running rMATS, you need to do some filtering on the results. The reason for this is that rMATS gives the value *NA* in the column *IncLevel* for events where no reads can support the events detected in the bam-file, which leads to inaccurate p-values. Here is an example of a gene with the *NA* value in the results.

The gene pre-mRNA processing factor 3 (PRPF3) is a protein coding gene, which also plays an important part in pre-mRNA splicing as a component of the U4/U6-U5 tri-snRNP complex in assembly of the spliceosome (*PRPF3 pre-mRNA processing factor 3 [Homo sapiens (human)]* 2022).

In the results file with skipped exon events, the gene occurred twice with different exon start base and exon end. They also differ in base pair length, where one exon contains 167 base pairs while the other only has 96. The IJC values for both the genes of the treatment sample (*12,2,0* and *14,8,11*) has higher values than SJC (*0,6,0* and *7,0,0*). This is also the same for the samples from the control group, meaning the respective exons were included in the transcript. The *IncLevelDifference* for the first gene is *-0.428* and the p-value is *1.0*, while for the second occurrence of PRPF3 the *IncLevelDifference* is *-0.093* and p-value is *0.00685*. However, I found the value *NA* in the columns of *IncLevel* for the gene with p-value *1.0*. When rMATS has an NA-value in Inclevel, it sets the p-value automatically to 1.0 regardless of what the actual p-value was. This means that this exon of the gene in the SE event can not be taken into consideration, as the values of this gene is incorrect. Nevertheless, the *IncLevelDifference* of the second gene is close to zero, meaning there is not a significant difference between the amount of exons included from the treatment group and the control group.

The gene PRPF3 could also be found in the results from alternative 3'

splice site (A3SS) events. The values of the SJCs and IJCs for the treatment group are *60,71,34* and *0,0,2*, respectively. Looking at the values for the control group where SJC is *29,29,16* and IJC is *6,1,2*, we can see that SJC is higher in both the treatment group and the control group. This means that the respective exon with the alternative 3' splice site got skipped. This differs from the samples of PRPF3 in the results of skipped exons, where the value of IJC were higher than SJC. The p-value of the gene in A3SS was *0.001353* and the *IncLevelDifference* was *-0.067*. As the *IncLevelDifference* is close to zero, there is no significant difference between the control group and the treatment group.

The results collected from the alternative splicing analysis of PRPF3 implies that the skipped exon event of alternative splicing is essential for the function of the retina, as the exon with the splicing event was included in the transcript. We can also see that the exon is present in the treatment group due to the low *IncLevelDifference*, meaning PRPF3 is not affected by the PRPF31 mutation. Correspondingly, the majority of the exons with alternative 3' splice site got skipped for both the treatment group and the control group, meaning it is not prevalent in either of the groups. The exon with the *NA* value is not included in this interpretation.

The results from the studies done by *Graziotto J, et al.* (Graziotto et al. 2008) supports the claim that the gene PRPF3 is vital for the retina. Graziotto also implies that high levels of expression in the retina could be related to retinitis pigmentosa. However, this conclusion also specifies that the relation to RP is through a toxic effect of the T494M and P493S mutations, which the research done by *Vaclavik V. et al.* (Veclavik et al. 2010) show is not the case due to the rarity of the mutation causing RP.

Two studies done on mutations in splicing factors ((Tanackovic et al.

2011) and (Ivings et al. 2008)) of PRPF3 and PRPF31 both found that retained intron events were linked to the disease retinitis pigmentosa. The results I have from PRPF3 does not include any events of intron retention, which supports the results done by Veclavik et al. 2010 that shows the gene PRPF3 is important in the retina and rarely shows any mutation that effects the visual field.

6.5 Conclusion

The research done in this thesis sets a foundation for acknowledging that alternative splicing events is fundamental for the creation of protein-isoforms. Furthermore, we can see the importance of filtering the results of rMATS, as the results with *NA*-values are not included in the analysis of splicing events.

The expectation that the Snakemake workflow could be implemented for alternative splicing analyses is satisfied. The document contains the necessary steps for adapting and executing the workflow on patient-control datasets. It is a reproducible and transparent program, available for all on GitHub (https://github.com/MadelenH/Snakemake).

6.6 Future Work

As this thesis was limited to the use of rMATS as the strategy for detection of alternative splicing events, a further expanding of the analysis by including several different analysing tools would be a logic direction of future work. The analysing tool MAJIQ (Vaquero-Garcia et al. 2016) could be included, which would lead to other information than what rMATS produces.

Further, the workflow could be adapted onto other treatment-control

data-sets, which would produce other genes for examination of alternative splicing events.

Furthermore, a tool for filtering the results of rMATS would be profitable, as there are several instances of exons with invalid results as a consequence of *NA*-values.

Further, analysis of alternative splicing events on the genes in Table 6.1 in the *Attachment*, which were filtered with the parameters $p - value < 0.05$ and $IncLevelDifference < -0.1$ and $IncLevelDifference > 0.1$ would be informative.

Attachment

Table 6.1: Table of genes from rMATS.

Genes				
ID	GeneID	GeneSymbol	P-value	IncLevelDifference
109	ENSMUSG00000001542	Ssbp2	0.00149033285961	-0.335
183	ENSMUSG00000024921	Ell2	0.00419506076937	0.433
232	ENSMUSG00000046139	Smarca2	0.000148231110837	0.287
272	ENSMUSG00000028487	Patl1	0.0131955561534	-0.28
325	ENSMUSG00000037369	Bnc2	0.000473727275681	-0.324
456	ENSMUSG00000032656	Kdm6a	0.000139537335285	-0.575
574	ENSMUSG00000026743	Tle4	0.00198780878424	0.456
739	ENSMUSG00000108037	Mllt10	0.0556945794724	0.264
780	ENSMUSG00000029267	U2surp	0.0208002461295	0.364
838	ENSMUSG00000000416	Mtf2	0.0118497729653	-0.26
937	ENSMUSG00000038784	Cttnbp2	0.00190886841867	-0.296
956	ENSMUSG00000027881	Cnot4	0.000889749789558	0.387
1118	ENSMUSG00000021693	Prpf38b	0.00753056215146	0.365
1414	ENSMUSG00000068917	Ctps	0.00000157135293	-0.25
1688	ENSMUSG00000032580	Clk2	0.0188671974258	-0.488

1724	ENSMUSG00000008305	Rbm5	0.0215699218297	0.375
1739	ENSMUSG00000060992	Tle1	0.0532866092743	0.266
2409	ENSMUSG00000020954	Copz1	0.000008974821893	-0.257
2575	ENSMUSG00000031278	Strn3	0.043542106157	0.291
2670	ENSMUSG00000001542	Acsl4	0.000677552510764	0.644
2710	ENSMUSG00000039153	Ell2	0.00322073478998	-0.657
2981	ENSMUSG00000017428	Runx2	0.0198901202062	0.404
3146	ENSMUSG00000050965	Psmd11	0.0210836395635	0.364
3154	ENSMUSG00000018363	Prkca	0.00304463129153	-0.466
3310	ENSMUSG00000030035	Smurf2	0.0252042574166	0.302
3315	ENSMUSG00000003345	Wbp1	0.00010426641919	-0.488
3434	ENSMUSG00000027206	Rhot1	0.0314073127613	-0.259
3439	ENSMUSG00000030232	Cops2	0.000004093721472	0.559
3599	ENSMUSG00000020886	Aebp2	0.881396400082	-0.35
3719	ENSMUSG00000030272	Dlg4	0.000566414617775	0.302
3751	ENSMUSG00000029267	Camk1	0.0933241180037	-0.511
3768	ENSMUSG00000020390	Mtf2	0.000007681656392	-0.367
3888	ENSMUSG00000020128	Ube2b	0.000181947271694	-0.667
3889	ENSMUSG00000020128	Vps54	0.0319179512268	0.261
4067	ENSMUSG00000029422	Vps54	0.00112648408772	0.312
4314	ENSMUSG00000021134	Rsrc2	0.000001270977904	-0.676
4746	ENSMUSG00000002028	Srsf5	0.000041890210156	0.505
4863	ENSMUSG00000020453	Kmt2a	0.000374080857467	-0.444
4973	ENSMUSG00000030330	Patz1	0.000416679287734	0.251
5836	ENSMUSG00000036977	Ing4	0.000001704652491	-0.667
6186	ENSMUSG00000028487	Anapc10	0.00147835096746	-0.304

6205	ENSMUSG00000008575	Bnc2	0.000931360183919	0.284
6206	ENSMUSG00000008575	Nfib	0.000020068943276	-0.256
6786	ENSMUSG00000027881	Nfib	0.000139083413819	-0.614
7242	ENSMUSG00000037369	Prpf38b	0.0385345666508	-0.492
7247	ENSMUSG00000037369	Kdm6a	0.0112624267444	-0.263
7267	ENSMUSG00000027763	Kdm6a	0.0450247133386	0.366
7658	ENSMUSG00000041658	Mbnl1	0.000001103292651	-0.385
8245	ENSMUSG00000000838	Rragb	0.00390818476533	-0.341
8580	ENSMUSG00000019984	Fmr1	0.00182837855967	-0.255
9176	ENSMUSG00000027829	Med23	0.0302654662141	-0.256
9436	ENSMUSG00000048154	Ccnl1	0.0159448992205	-0.272
9437	ENSMUSG00000067567	Kmt2d	0.00131870392319	0.333
9563	ENSMUSG00000027109	Hdac8	0.00350703971126	0.449
9797	ENSMUSG00000030516	Sp3	0.0215227956947	0.333
10104	ENSMUSG00000034621	Tjp1	0.0223886345926	-0.324
10106	ENSMUSG00000034621	Gpatch8	0.00130519879553	-0.303
10108	ENSMUSG00000034621	Gpatch8	0.000005118934763	-0.352
10378	ENSMUSG00000038861	Gpatch8	0.000030830303010	0.57
10391	ENSMUSG00000035696	Pi4kb	0.0375334590809	0.292
10499	ENSMUSG00000034269	Rnf38	0.0323115670912	-0.394
10684	ENSMUSG00000051285	Setd5	0.00420572495078	-0.295
11426	ENSMUSG00000055436	Cul3	0.00695401546023	-0.303
11539	ENSMUSG00000026150	Srsf11	0.000005111123728	-0.261
11672	ENSMUSG00000056211	Mff	0.0151753056445	-0.325
11675	ENSMUSG00000056211	R3hdm1	0.000182666745039	-0.451
12054	ENSMUSG00000041757	R3hdm1	0.000084018035313	0.667

12474	ENSMUSG00000045103	Plekha6	0.0178383629363	-0.309
12614	ENSMUSG00000008200	Dmd	0.000120163738626	-0.383
12747	ENSMUSG00000032525	Fnbp4	0.000628549268013	0.266
13351	ENSMUSG00000021495	Wdr20	0.0223186075211	-0.316
13446	ENSMUSG00000027455	Fam193b	0.000005578275961	0.453
14562	ENSMUSG00000040274	Nsfl1c	0.000559211570161	0.35
14974	ENSMUSG00000029647	Cdk6	0.000248175231776	-0.303
16998	ENSMUSG00000038346	Arglu1	0.000001588912951	0.718
18072	ENSMUSG00000020453	Zfp384	0.0363549140471	-0.26
19101	ENSMUSG00000037876	Atxn2l	0.000123981853508	-0.667
19175	ENSMUSG00000021709	Jmjd1c	0.0380928702539	0.272

Bibliography

Adapter Content (2021). URL: https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3%5C%20Analysis%5C%20Modules/10%5C%20Adapter%5C%20Content.html (visited on 10/02/2021).

Babraham Bioinformatics - FastQC A Quality Control tool for High Throughput Sequence Data (2021). URL: https://www.bioinformatics.babraham.ac.uk/projects/fastqc/ (visited on 07/20/2021).

Bolger, Anthony (2021). *Trimmomatic (v2)*. URL: https://www.genepattern.org/modules/docs/Trimmomatic/ (visited on 09/10/2021).

Bolger AM Lohse M, Usadel B (Aug. 2014). "Trimmomatic: a flexible trimmer for Illumina sequence data". In: *Bioinformatics*. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btu170. URL: https://pubmed.ncbi.nlm.nih.gov/24695404.

Braelle, Francisco E and Jimena Giudice (May 2017). "Alternative splicing as a regulator of development and tissue identify". In: *Nature reviews. Molecular cell biology* 18.7, pp. 437–451. ISSN: 1471-0080. DOI: 10.1038/nrm.2017.27. URL: https://pubmed.ncbi.nlm.nih.gov/28488700.

Buskin, Adriana et al. (Oct. 2018). "Disrupted alternative splicing for genes implicated in splicing and ciliogenesis causes PRPF31 retinitis pigmentosa". In: *Nature Communications*. ISSN: 2041-1723. DOI: 10.1038/s41467-018-06448-y. URL: https://doi.org/10.1038/s41467-018-06448-y.

Campochiaro, Peter A. and Tahreem A. Mir (2018). "The mechanism of cone cell death in Retinitis Pigmentosa". In: *Progress in Retinal and Eye Research* 62, pp. 24–37. ISSN: 1350-9462. DOI: https://doi.org/10.1016/j.preteyeres.2017.08.004. URL: https://www.sciencedirect.com/science/article/pii/S135094621730071X.

Clancy, Suzanne and William Brown (2008). *Tanslation: DNA to mRNA to Protein*. URL: https://www.nature.com/scitable/topicpage/translation-dna-to-mrna-to-protein-393/ (visited on 04/07/2022).

Conda (2017). URL: https://docs.conda.io/en/latest/ (visited on 04/06/2022).

Danecek, Petr et al. (Feb. 2021). "Twelve years of SAMtools and BCFtools". In: *GigaScience* 10.2. giab008. ISSN: 2047-217X. DOI: 10.1093/gigascience/giab008. eprint: https://academic.oup.com/gigascience/article-pdf/10/2/giab008/36332246/giab008.pdf. URL: https://doi.org/10.1093/gigascience/giab008.

DNA (2022). URL: https://www.britannica.com/science/DNA (visited on 04/07/2022).

Dobin, Alexander (Jan. 2022). *STAR manual 2.7.10a*. URL: https://github.com/alexdobin/STAR/blob/master/doc/STARmanual.pdf (visited on 04/12/2022).

Dobin, Alexander and Thomas R Gingeras (Sept. 2015). "Mapping RNA-seq Reads with STAR". In: *Current protocols in bioinformatics* 51. ISSN: 1934-3396. DOI: 10.1002/0471250953.bi1114s51. URL: https://10.1002/0471250953.bi1114s51.

Duplicate Sequences (2021). URL: https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3%5C%20Analysis%5C%20Modules/8%5C%20Duplicate%5C%20Sequences.html (visited on 09/29/2021).

Ebbert, MTW et al. (July 2016). "Evaluating the necessity of PCR duplicate removal from next-generation sequencing data and a comparison of approaches." In: *BMC Bioinformatics*. ISSN: 1471-2105. DOI: 10.1186/s12859-016-1097-3. URL: https://doi.org/10.1186/s12859-016-1097-3.

FASTA (2022). URL: https://zhanggroup.org/FASTA/ (visited on 04/12/2022).

FASTQ files explained (2021). URL: https://support.illumina.com/bulletins/2016/04/fastq-files-explained.html (visited on 11/30/2021).

FastQ Format (2022). URL: https://learn.gencore.bio.nyu.edu/ngs-file-formats/fastq-format/ (visited on 04/07/2022).

GFF/GTF File Format - Definition and supported options (2021). URL: https://www.ensembl.org/info/website/upload/gff.html (visited on 09/20/2021).

Gilbert, W (1978). "Why genes in pieces?" In: *Nature* 271, p. 501. ISSN: 1476-4687. DOI: 10.1038/271501a0. URL: https://doi.org/10.1038/271501a0.

GitHub (2022). URL: https://github.com/about (visited on 02/11/2022).

Graziotto, J et al. (Aug. 2008). "Decreased Levels of the RNA Splicing Factor Prpf3 in Mice and Zebrafish Do Not Cause Photoreceptor Degeneration". In: *Biochemistry and Molecular Biology*. DOI: doi.org/10.1167/iovs.07-1483. URL: https://doi.org/10.1167/iovs.07-1483.

Greenberg, D.S. and H. Soreq (2013). "Alternative Splicing". In: *Brenner's Encyclopedia of Genetics (Second Edition)*. Ed. by Stanley Maloy and Kelly Hughes. Second Edition. San Diego: Academic Press, pp. 97–98. ISBN: 978-0-08-096156-9. DOI: https://doi.org/10.1016/B978-0-12-374984-0.00043-7. URL: https://www.sciencedirect.com/science/article/pii/B9780123749840000437.

Gregory-Evans, Kevin, Mark E. Pennesi, and Richard G. Weleber (2013). "Chapter 40 - Retinitis Pigmentosa and Allied Disorders". In: *Retina (Fifth Edition)*. Ed. by Stephen J. Ryan et al. Fifth Edition. London: W.B. Saunders, pp. 761–835. ISBN: 978-1-4557-0737-9. DOI: https://doi.org/10.1016/B978-1-4557-0737-9.00040-0. URL: https://www.sciencedirect.com/science/article/pii/B9781455707379000400.

GTF2.2: A Gene Annotation Format (Revised Ensembl GTF) (2021). URL: https://mblab.wustl.edu/GTF22.html (visited on 10/21/2021).

Hu, Y et al. (Nov. 2012). "DiffSplice: the genome-wide detection of differential splicing events with RNA-seq". In: *Nucleic acids research* 41. ISSN: 1362-4962. DOI: 10.1093/nar/gks1026. URL: https://pubmed.ncbi.nlm.nih.gov/23155066.

Introduction to NGS (2021). URL: https://www.illumina.com/science/technology/next-generation-sequencing.html (visited on 11/30/2021).

Introduction to RNA-Seq using high-performance computing - archived (2021). URL: https://hbctraining.github.io/Intro-to-rnaseq-hpc-O2/lessons/03_alignment.html (visited on 12/06/2021).

Ivings, L et al. (Dec. 2008). "Evaluation of splicing efficiency in lymphoblastoid cell lines from patients with splicing-factor retinitis pigmentosa". In: *Molecular vision*, pp. 2357–2366. ISSN: 1090-0535.

Jin, Yongfeng et al. (May 2018). "Mutually exclusive alternative splicing of pre-mRNAs". In: *Wiley interdisciplinary reviews. RNA*. ISSN: 1757-7004. DOI: 10.1002/wrna.1468.

Katz, Y et al. (Nov. 2010). "Analysis and design of RNA sequencing experiments for identifying isoform regulation". In: *Nature methods* 7, pp. 1009–1015. ISSN: 1548-7105. DOI: 10.1038/nmeth.1528. URL: https://pubmed.ncbi.nlm.nih.gov/21057496.

Kazilek, CJ and K Cooper (Jan. 2010). *Rods and Cones*. URL: https://askabiologist.asu.edu/rods-and-cones (visited on 04/07/2022).

KDM6A gene (Aug. 2020). URL: https://medlineplus.gov/genetics/gene/kdm6a/ (visited on 04/08/2022).

Kooistra, S and K Helin (2012). "Molecular mechanisms and potential functions of histone demethylases". In: *Nature Reviews Molecular Cell Biology* 13, pp. 297–311. ISSN: 1471-0080. DOI: 10.1038/nrm3327. URL: https://doi.org/10.1038/nrm3327.

Koren, E, G Lev-Maor, and G Ast (May 2007). "The Emergence of Alternative 3' and 5' Splice Site Exons from Constitutive Exons". In: URL: https://doi.org/10.1371/journal.pcbi.0030095.

Köster, Johannes and Sven Rahmann (Aug. 2012). "Snakemake—a scalable bioinformatics workflow engine". In: *Bioinformatics* 28.19, pp. 2520–2522. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bts480. eprint: https://academic.oup.com/bioinformatics/article-pdf/28/19/2520/819790/bts480.pdf. URL: https://doi.org/10.1093/bioinformatics/bts480.

Le, Kain-qin et al. (Oct. 2015). "Alternative splicing as a biomarker and potential target for drug discovery". In: *Acta Pharmacologica Sinica* 36, pp. 1212–1218. ISSN: 1745-7254. DOI: 10.1038/aps.2015.43. URL: https://doi.org/10.1038/aps.2015.43.

Li, H (2021a). *Samtools index - indexes SAM/BAM/CRAM files*. URL: http://www.htslib.org/doc/samtools-index.html (visited on 07/20/2021).

— (2021b). *Samtools sort - sorts BAM/SAM/CRAM files*. URL: http://www.htslib.org/doc/samtools-sort.html (visited on 07/20/2021).

Li, Wei (2021). *netthinnen*. URL: https://sml.snl.no/netthinnen (visited on 10/20/2021).

Liu, J et al. (2012). "Genome and transcriptome sequencing of lung cancers reveal diverse mutational and splicing events". In: *Genome research* 22. ISSN: 1549-5469. DOI: 10.1101/gr.140988.112.

Louadi, Zakaria et al. (Aug. 2019). "Deep Splicing Code: Classifying Alternative Splicing Events Using Deep Learning". In: *Genes* 10. DOI: 10.3390/genes10080587.

Mackenzie, RJ (2021). *RNA-Seq: Basics, Applications and Protocol*. URL: https://www.technologynetworks.com/genomics/articles/rna-seq-basics-applications-and-protocol-299461 (visited on 11/30/2021).

Makarova, Olga V. et al. (Mar. 2002). "Protein 61K, encoded by a gene (PRPF31) linked to autosomal dominant retinitis pigmentosa, is required for U4/U6*U5 tri-snRNP formation and pre-mRNA splicing". In: *The EMBO journal*, pp. 1148–1157. ISSN: 1460-2075. DOI: 10.1093/emboj/21.5.1148. URL: https://doi.org/10.1093/emboj/21.5.1148.

Mehmood, Arfa et al. (Dec. 2019). "Systematic evaluation of differential splicing tools for RNA-seq studies". In: *Briefings in Bioinformatics* 21.6, pp. 2052–2065. ISSN: 1477-4054. DOI: 10.1093/bib/bbz126. eprint: https://academic.oup.com/bib/article-pdf/21/6/2052/34672001/bbz126.pdf. URL: https://doi.org/10.1093/bib/bbz126.

Miyake, N et al. (Oct. 2012). "KDM6A Point Mutations Cause Kabuki Syndrome". In: DOI: 10.1002/humu.22229. URL: https://doi.org/10.1002/humu.22229.

Modrek, B. and Lee C. (2002). "A genomic view of alternative splicing". In: Nature Genetics.30, pp. 13–19. ISSN: 1546-1718. DOI: 10.1038/ng0102-13. URL: https://doi.org/10.1038/ng0102-13.

Mölder, F et al. (2021). "Sustainable data analysis with Snakemake [version 2; peer review: 2 approved". In: URL: (https://doi.org/10.12688/f1000research.29032.2).

MTf2 metal response element binding transription factor 2 [Homo sapiens (human)] (2022). URL: https://www.ncbi.nlm.nih.gov/gene/22823 (visited on 04/05/2022).

Overrepresented Sequences (2021). URL: https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3%5C%20Analysis%5C%20Modules/9%5C%20Overrepresented%5C%20Sequences.html (visited on 09/29/2021).

Parkinson, John and Mark Blaxter (2009). "Expressed sequence tags: an overview". In: *Methods in molecular biology*, pp. 1–12. ISSN: 1064-3745. DOI: 10.1007/978-1-60327-136-3_1.

Per Base Sequence Content (2021). URL: https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3%5C%20Analysis%5C%20Modules/4%5C%20Per%5C%20Base%5C%20Sequence%5C%20Content.html (visited on 09/27/2021).

Per Base Sequence Quality (2021). URL: https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3%5C%20Analysis%5C%20Modules/2%5C%20Per%5C%20Base%5C%20Sequence%5C%20Quality.html (visited on 09/27/2021).

Per Sequence GC Content (2021). URL: https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3%5C%20Analysis%5C%20Modules/5%5C%20Per%5C%20Sequence%5C%20GC%5C%20Content.html (visited on 09/27/2021).

Per Sequence Quality Scores (2021). URL: https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3%5C%20Analysis%5C%

20Modules/3%5C%20Per%5C%20Sequence%5C%20Quality%5C%20Scores.html (visited on 09/27/2021).

Per Tile Sequence Quality (2021). URL: https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3%5C%20Analysis%5C%20Modules/12%5C%20Per%5C%20Tile%5C%20Sequence%5C%20Quality.html (visited on 09/29/2021).

PRPF3 pre-mRNA processing factor 3 [Homo sapiens (human)] (2022). URL: https://www.ncbi.nlm.nih.gov/gene/9129 (visited on 03/09/2022).

Retinitis Pigmentosa (2020). URL: https://medlineplus.gov/genetics/condition/retinitis-pigmentosa/#resources (visited on 04/10/2022).

Retinitis Pigmentosa (2021). URL: https://www.nei.nih.gov/learn-about-eye-health/eye-conditions-and-diseases/retinitis-pigmentosa (visited on 10/20/2021).

Retinitis Pigmentosa (2022). URL: https://visionaware.org/your-eye-condition/retinitis-pigmentosa/ (visited on 03/24/2022).

Retinitis pigmentosa awareness month (2022). URL: https://www.salusuhealth.com/Eye-Institute/News/News-Stories/Retinitis-Pigmentosa-Awareness-Month.aspx# (visited on 03/24/2022).

Robinson, MD, DJ McCarthy, and GK Smyth (Jan. 2010). "edgeR: a Bioconductor package for differential expression analysis of digital gene expression data". In: *Bioinformatics*. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btp616. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2796818/.

Rodriguez-Esteban, R and X Jiang (Oct. 2017). "Differential gene expression in disease: a comparison between high-throughput studies and the literature". In: *BCM Med Genomics* 10. DOI: https://doi.org/10.1186/

s12920-017-0293-y. URL: https://bmcmedgenomics.biomedcentral.com/articles/10.1186/s12920-017-0293-y#citeas.

Rose, AM and SS Bhattacharya (2016). "Variant haploinsufficiency and phenotypic non-penetrance in PRPF31-associated retinitis pigmentosa". In: *Clinical genetics* 90, pp. 118–126. ISSN: 1399-0004. DOI: 10.1111/cge.12758. URL: https://doi.org/10.1111/cge.12758.

Sequence Length Distribution (2021). URL: https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3%5C%20Analysis%5C%20Modules/7%5C%20Sequence%5C%20Length%5C%20Distribution.html (visited on 09/29/2021).

Shah, SH and JA Pallas (Jan. 2009). "Identifying differential exon splicing using linear models and correlation". In: *BMC Bioinformatics*. ISSN: 1471-2105. DOI: 10.1186/1471-2105-10-26. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2636774/.

Shen, Shihao, Juw Won Park, Jian Huang, et al. (2012). "MATS: a Bayesian framework for flexible detection of differential alternative splicing from RNA-Seq data". In: *Nucleic Acids Res* 40.8. DOI: https://doi.org/10.1093/nar/gkr1291. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3333886/.

Shen, Shihao, Juw Won Park, Zhi-xiang Lu, et al. (Dec. 2014). "rMATS: robust and flexible detection of differential alternative splicing from replicate RNA-Seq data". In: *Proceedings of the National Academy of Sciences of the United States of America* 111.51. DOI: https://doi.org/10.1073/pnas.1419161111. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4280593/.

Tanackovic, G et al. (June 2011). "PRPF mutations are associated with generalized defects in spliceosome formation and pre-mRNA splicing in

patients with retinitis pigmentosa". In: *Human molecular genetics* 20, pp. 2116–2130. ISSN: 1460-2083. DOI: 10.1093/hmg/ddr094. URL: https://doi.org/10.1093/hmg/ddr094.

Tanaka, Masahiro and Osamu Tatebe (2010). "Pwrake: A Parallel and Distributed Flexible Workflow Management Tool for Wide-Area Data Intensive Computing". In: *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*. HPDC '10. Chicago, Illinois: Association for Computing Machinery, pp. 356–359. ISBN: 9781605589428. DOI: 10.1145/1851476.1851529. URL: https://doi.org/10.1145/1851476.1851529.

Taura, Kenjiro et al. (2013). "Design and implementation of GXP make — A workflow system based on make". In: *Future Generation Computer Systems* 29.2. Special section: Recent advances in e-Science, pp. 662–672. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2011.05.026. URL: https://www.sciencedirect.com/science/article/pii/S0167739X11001087.

Terminology of Molecular Biology for Upstream/Downstream (2022). URL: https://www.genscript.com/biology-glossary/12162/UpstreamDownstream (visited on 04/05/2022).

Thakur, Prasoon K. et al. (2019). "Bioinformatics Approaches for Studying Alternative Splicing". In: *Encyclopedia of Bioinformatics and Computational Biology*. Ed. by Shoba Ranganathan et al. Oxford: Academic Press, pp. 221–234. ISBN: 978-0-12-811432-2. DOI: https://doi.org/10.1016/B978-0-12-809633-8.20228-8. URL: https://www.sciencedirect.com/science/article/pii/B9780128096338202288.

The Snakemake Wrappers repository (2016). URL: https://snakemake-wrappers.readthedocs.io/en/stable/ (visited on 04/07/2022).

Trapnell, Cole, Davic G Hendrickson, et al. (Jan. 2013). "Differential analysis of gene regulation at transcript resolution with RNA-seq". In: *Nature Biotechnology*, pp. 46–53. ISSN: 1546-1696. DOI: 10.1038/nbt.2450. URL: https://doi.org/10.1038/nbt.2450.

Trapnell, Cole and Steven L Salzberg (May 2009). "How to map billions of short reads onto genomes". In: *Nature biotechnology*. ISSN: 1546-1696. DOI: 10.1038/nbt0509-455.

Trimmomatic (2022). URL: https://anaconda.org/bioconda/trimmomatic (visited on 04/07/2022).

Vaquero-Garcia, J et al. (Feb. 2016). "A new view of transcriptome complexity and regulation through the lens of local splicing variations". In: *eLife*. ISSN: 2050-084x. DOI: 10.7554/eLife.11752. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4801060/.

Veclavik, Veronica et al. (Mar. 2010). "Variable phenotypic expressivity in a Swiss family with autosomal dominant retinitis pigmentosa due to a T494M mutation in the PRPF3 gene." In: *Molecular vision*. ISSN: 1090-0535.

Wang, Yan et al. (2015). "Mechanism of alternative splicing and its regulation". In: *Biomedical reports* 3.2. ISSN: 2049-9442. DOI: 10.3892/br.2014.407.

Wetterstrand, Kris A (2021). *The Cost of Sequencing a Human Genome*. URL: https://www.genome.gov/about-genomics/fact-sheets/Sequencing-Human-Genome-cost (visited on 09/15/2021).

Wheway, G et al. (2020). "Mutation spectrum of PRPF31, genotype-phenotype correlation in retinitis pigmentosa, and opportunities for therapy". In: *Experimental eye research* 192. ISSN: 1096-0007. DOI: 10.1016/j.exer.2020.107950.

Wrappers (2016). URL: `https://snakemake-wrappers.readthedocs.io/en/stable/wrappers.html` (visited on 04/06/2022).