NUCLEOTIDE SEQUENCE ANALYSIS PRACTICAL WORK

-WORKBOOK-

UNIVERSITY OF PRIMORSKA

FACULTY OF MATHEMATICS, NATURAL SCIENCES AND INFORMATION TECHNOLOGIES

APPLICATION OF RNA-SEQ AND MICROARRAY TRANSCRIPTOME PROFILES IN GENE EXPRESSION STUDY

Organism: Arabidopsis Thaliana

Ivana Štrbac, Marija Rakić, Sara Milovanova

PART 1

INFO ABOUT RESEARCH AND SEQUENCING DATASET

Link to the BioProject record

a) What was the aim of the study, described in the scientific article?

The aim of the study was to test the feasibility of using tiny root sections for transcriptome analysis. They compared RNA sequencing (RNA‑Seq) to microarrays in characterizing genes that are relevant to spaceflight.

b) Provide some info about the dataset you will work with and which sequencing technology was used.

The seedlings of a plant known as *Arabidopsis thaliana* Columbia ecotype were grown on sterile solid media plates and adjusted to pH 5.72 with 1 M KOH. The plates were vertically oriented in growth chambers with continuous light at a constant temperature of 19°C. From a single plate eight‑day‑old seedlings were harvested and placed in RNA*later* and stored in a freezer at −20°C. Afterwards seedlings were moved to a room temperature and transferred to an open Petri dish from which the researchers selected the primary roots and dissected them into appropriate zones. Three independent seedlings were used as three biological replicates for each of the transcriptome analyses. The root tips were dissected into Zone 1 (0.5 mm from the tip, including the root cap and root division zones) and Zone 2 (1.5‑mm sections including the root elongation and root differentiation zone). *Measuring of genes was done using microarray and RNA‑Seq. The main difference between RNA-Seq and microarrays is that the RNA-Seq allows for full sequencing of the whole transcriptome while the microarrays only profiles predefined transcripts or genes through hybridization.* In further analysis the transcripts identified as significantly changed at fold change of 2 or greater from the two techniques were used.

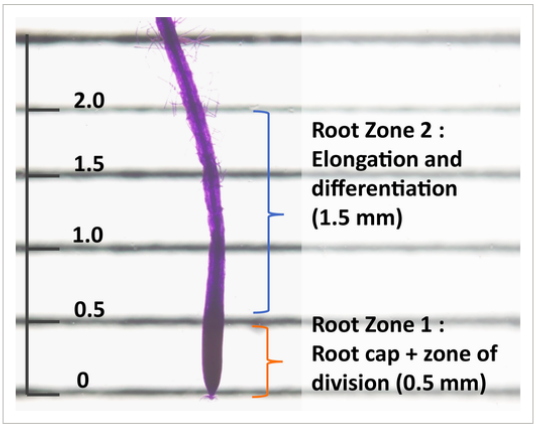


Figure 1

Root dissection shown in eight-day-old *Arabidopsis thaliana* Columbia ecotype seedlings, grown on vertically placed Phytagel plates under continuous light. The root apex was sectioned into two zones: Zone 1 (0.5 mm; root cap and meristematic zone) and Zone 2 (1.5 mm; transition, elongation, and growth-terminating zone). The root tip shown is an eight-day-old, RNA*later*-preserved seedling stained with toluidine blue observed under a dissecting microscope. Scale is in millimeters.

c) **Which kit was used for sequencing library preparation? Does this kit preserve strand information (stranded library) or not?**

For RNA-Seq library preparation ClonTech SMART-Seq V4 Ultra Low Input RNA Kit for sequencing combined with Illumina Nextera DNA Sample Preparation Kit was used on five nanograms of total RNA. The SMART-Seq v4 Ultra Low Input RNA Kit for Sequencing stands for "**S**witching **M**echanism **a**t 5′ End of **R**NA **T**emplate". This technology enriches full-length cDNAs because it is based on the template switching activity of reverse transcriptases. In addition, it adds defined PCR adapters directly to both ends of the first-strand cDNA. Therefore, the final cDNA libraries contain the 5′ end of the mRNA and maintain a true representation of the original mRNA transcripts. Hence the kit preserves the strand information. A schematic outline of the technology workflow is shown in Figure 2 below.

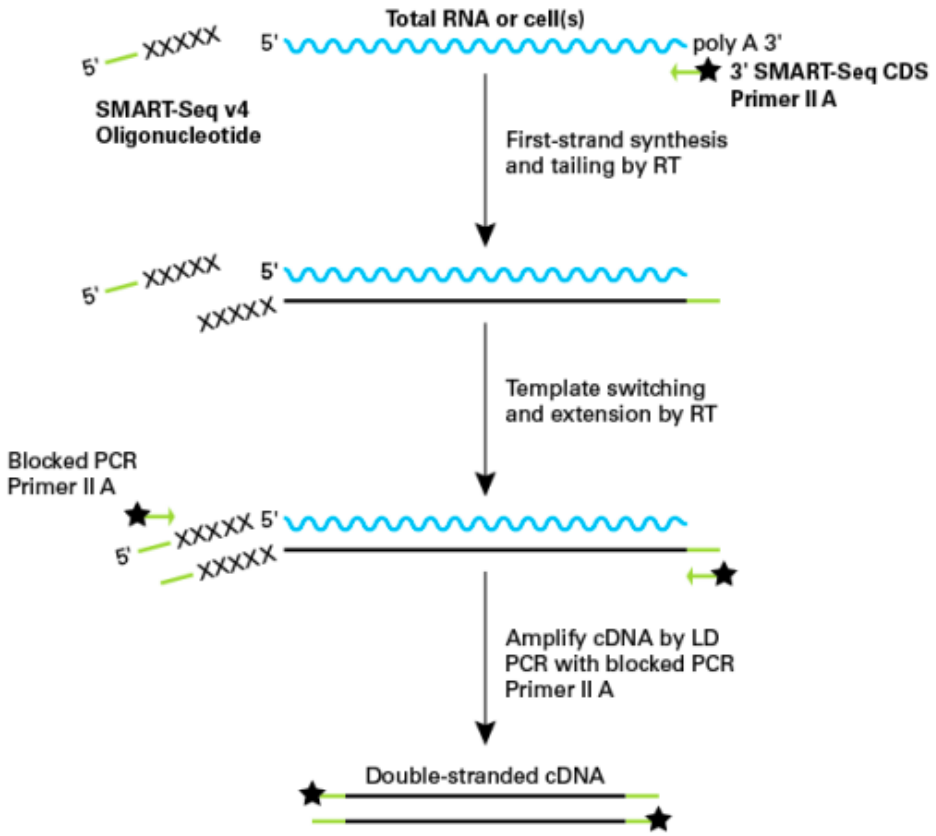


Figure 2. Flowchart of SMART cDNA synthesis. The SMART-Seq v4 Oligonucleotide, 3′ SMART-Seq CDS Primer II A, and PCR Primer II A all contain a stretch of identical sequence. The black star indicates a chemical block on the 5′ end of the oligonucleotide.

d) What is the advantage of stranded mRNA library preparation compared to non-stranded libraries?

Stranded RNA-seq retains strand information of a read, so we can resolve read ambiguity in overlapping genes transcribed from opposite strands, which provides a more accurate quantification of gene expression levels compared with traditional non-stranded RNA-seq.

DOWNLOADING THE DATA

Every operation is performed in Linux. Also, in order to do this, we had to install the SRA toolkit earlier, as well as Anaconda or Miniconda which are installers for the conda, Python and other useful packages.

The SRA Toolkit from NCBI is a collection of tools and libraries for using data in the INSDC Sequence Read Archives. In the SRA Toolkit we can download data using Prefetch. This program downloads Runs, or the sequence files in the compressed SRA format, and all additional data necessary to convert the Run from the SRA format to a more commonly used format. Prefetch can be used to correct and finish an incomplete run download. Fastq-dump and

sam-dump are also part of the SRA toolkit and can be used to convert the prefetched Runs from compressed SRA format to fastq or sam format. Additionally, we can always avoid the prefetch step and download and convert the run in one step by entering just the run accession without the .sra extension in our fastq-dump or sam-dump command.

EXAMPLES:
One Run: $ prefetch SRR000001

A list of Runs: $ prefetch --option-file SraAccList.txt

$ fastq-dump --split-files SRR11180057.sra
$ fastq-dump --split-files SRR11180057

PRACTICAL PART

1. $ prefetch SRR7287369

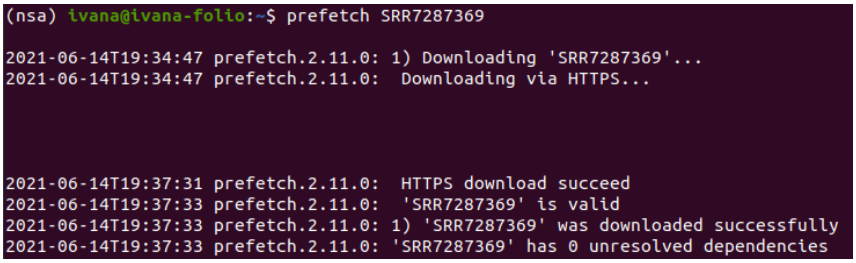


Figure 3. - Result of prefetch function

2. $ fastq-dump -X 10 -Z SRR7287369

- Here "-X" stands for maximum spots id, the number 10 means that we want to read to spots, and "-Z" means that we will have an output to stdout and all split data become joined into a single stream.

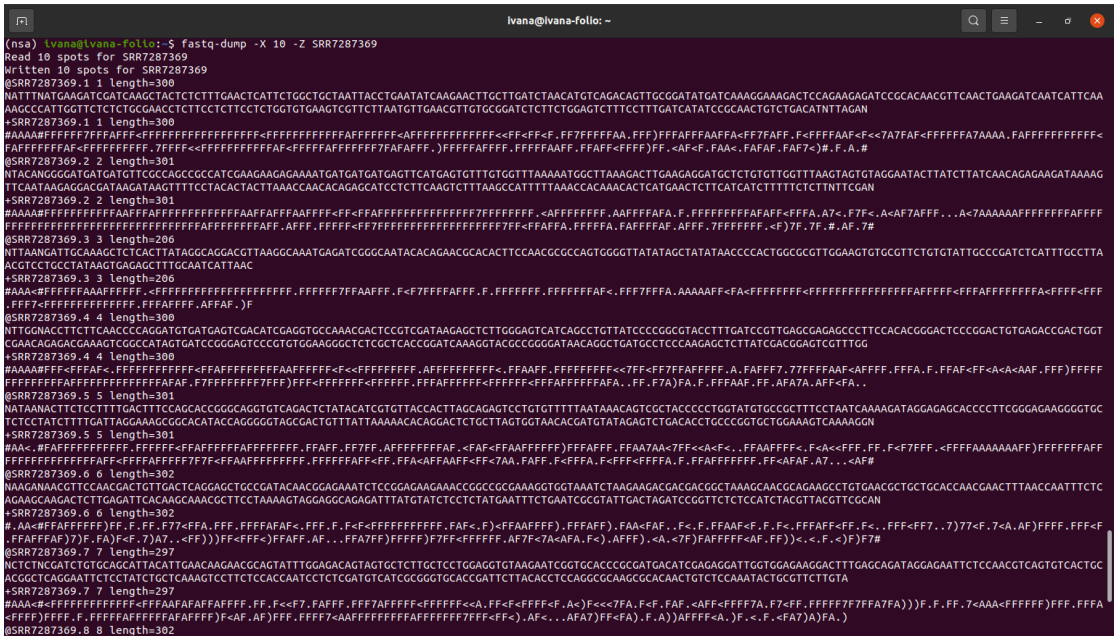


Figure 4. - Result of first seven spots read with the second function done

Figure 5. - Result including last three spots read with the second function done

1. $ fastq-dump -X 5 -Z SRR7287369 --split-3

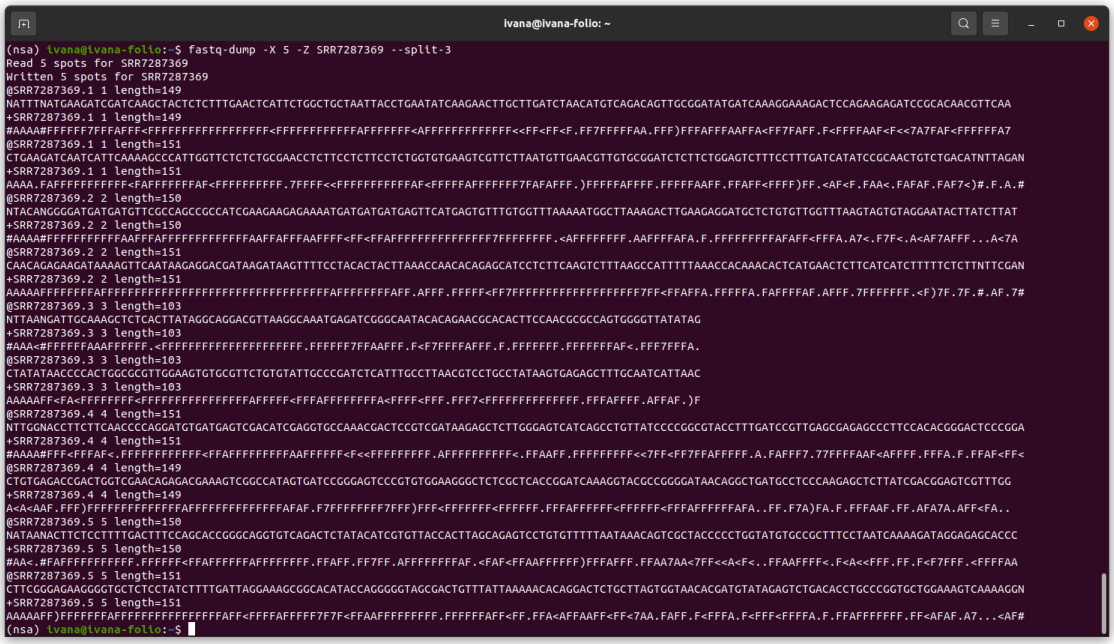


Figure 6. - Result of five sports when adding “--split-3”

- Prints the first five spots (-X 5) to standard out (-Z). This is a useful starting point for verifying other formatting options before dumping a whole file.
- We have to use the split command (there are two types that we mentioned: --split-3 and --split-files) because we have a paired-end data, otherwise the read from one end and the read from the other end would be joined
- If in the datasets there is only one read from one molecule and the read from the other side is missing, then --split-3 function allows us three files: forward, reverse and unpaired reads, where unpaired reads will be saved in a separate file.

ADDITIONAL:

1. Creating a folder in which we write 100 spots

```
(nsa) ivana@ivana-folio:~$ mkdir nsa_test
(nsa) ivana@ivana-folio:~$ cd nsa_test/
(nsa) ivana@ivana-folio:~/nsa_test$ fastq-dump -X 100 SRR7287369 --split-3
Read 100 spots for SRR7287369
Written 100 spots for SRR7287369
```

Figure 7. - Functions for completing the above stated under number 1

By doing this we created two files in our folder, one for the forward and the other for the reverse sequence pair. Important to state here is that the name of the sequence is the SRA accession number. Then for the spots it is usually used dot after the accession number after which comes the number of the spot and after that we have the pair number, so each spot will have two pairs, after which for each of the pair is written the length of the sequence. Of course, we are aware and know that this is not the original name of the sequence obtained from the sequencing machine. Actually, the SRA database discards the original sequence names. First reason for that is because they are not standardized, so at the same time they aren't easy to deal with, and the second reason

is that using the accession number takes less memory.

2. $ fastq-dump -X 5 -Z SRR7287369 --split-3 --origfmt

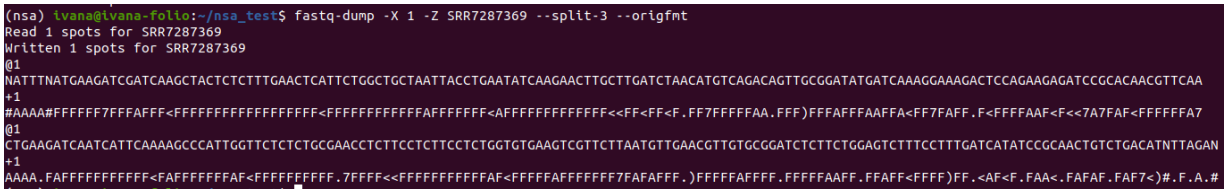


Figure 8. - Retrieving the original sequence name of the first spot

- We use the command "-origfmt" to retrieve original names in format

3. $ fastq-dump SRR7287369 --split-3 --origfmt --skip-technical --read-filter pass -clip M 25

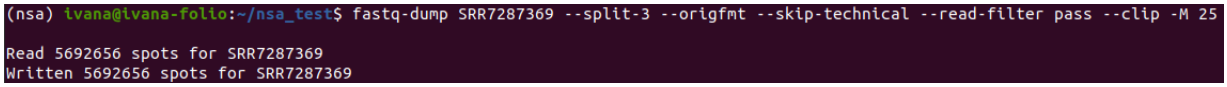


Figure 9. - Read and written sports using the above function written in the number 3

As we can see there were in total 5 692 656 spots after performing this function.

- If the sequencing was done with the "Illumina multiplex library construction protocol" the SRA entry ends up with application reads and technical reads like the following "Application Read Forward -> Technical Read Forward <- Application Read Reverse - Technical Read Reverse." We don't want the technical reads, we only want the biological reads. Therefore we use "--skip-technical" to remove those technical parts. If we omit this option and include the –split-files we actually end up with three or four files per SRA archive!
- We use "--read-filter pass" when we want to filter out reads that are all N's or otherwise completely useless. This filter can be set to pass|reject|criteria|redacted but we want only those reads that pass filtering.
- Some of the sequences in the SRA contain tags that are used for example for whole genome amplification and need to be removed. We use "--clip" as it will trim off those sequences.
- We use "-M" to filter the sequences by the minimum read length.

CHECKING QUALITY OF THE SEQUENCES

Once we have the data it is recommended to check the quality of the sequences we read and wrote in our files. FastQC is a quality control tool for high throughput sequence data and is written in Java programming language. The main goal of FastQC is to provide a simple way to do some quality control checks on raw sequence data coming from high throughput sequencing pipelines. It provides a modular set of analyses which we can use to give a quick impression of whether our data has any problems of which we should be aware before doing any further analysis.

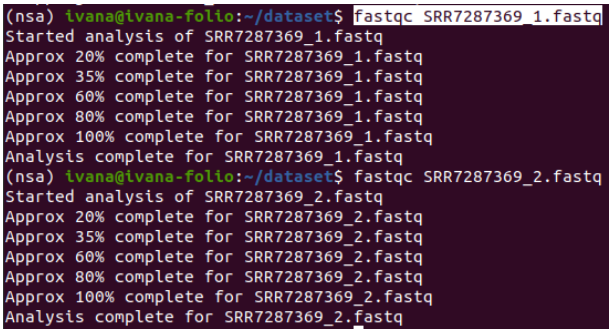
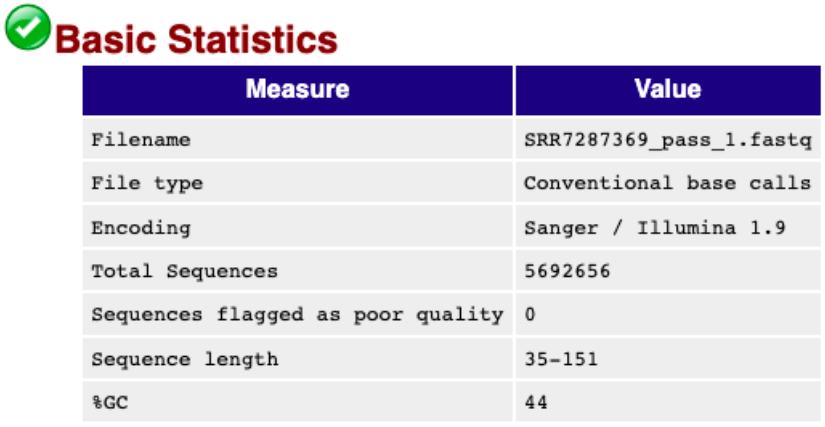


Figure 10. - Running fastqc on two pairs

As a result we got two html files each for one analysis. Therefore we can take a look at the metrics and assess the quality of our sequencing data!

FIRST FILE ANALYSIS:
The first part, which is basic statistics is the same for both files and looks like this:



Basic Statistics

Measure	Value
Filename	SRR7287369_pass_1.fastq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	5692656
Sequences flagged as poor quality	0
Sequence length	35-151
%GC	44

Figure 11. - Basic statistics of FastQC Report

So, as we can see here we have a file name, which is the only thing that is different regarding two pairs, and file type, as well as type of encoding, which is Illumina 1.9. Also, the total number of sequences is written, which is 5692656. There are no sequences flagged as poor quality in neither file, and the sequence length ranges between 35 and 151 base pairs. The GC percentage is 44.

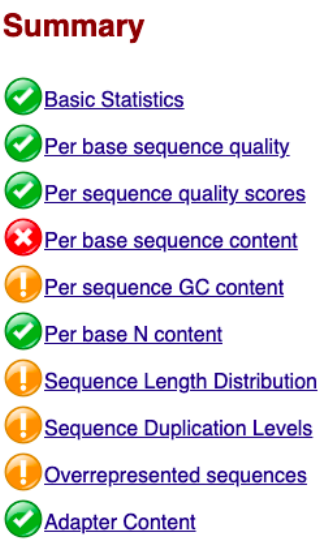


Figure 12. - FastQC Report summary

Within our report, a summary of all of the modules is given on the left-hand side of the report. Next to each module there is a sign, which is either green, yellow or red. It indicated the quality of the data. However, after looking up on the internet, I have found that we do not have to take the yellow "warning"s and red "fail"s too seriously, since they should be interpreted as flags for modules to check out.

One of the most important analysis modules is the **"Per base sequence quality"** plot. This plot provides the distribution of quality scores at each position in the read across all reads. This plot can alert us to whether there were any problems occuring during sequencing.

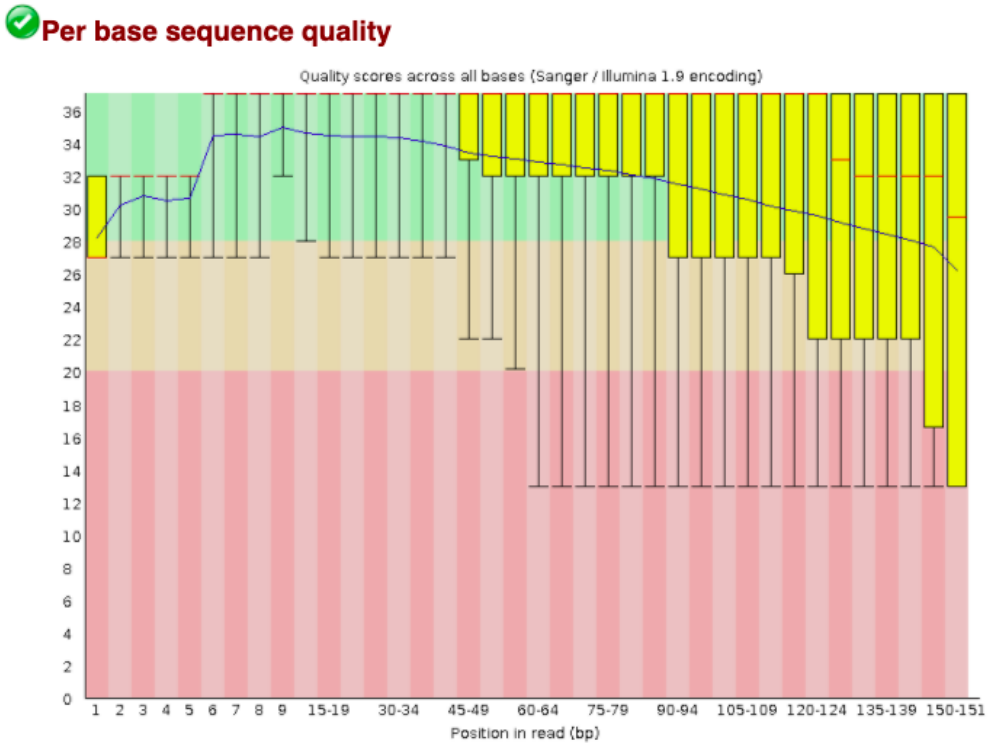


Figure 13. - Per base sequence quality plot

The y-axis gives the quality scores, while the x-axis represents the position in the read. The color coding of the plot denotes what are considered high, medium and low quality scores. The yellow box represents the 25th and 75th percentiles, with the red line as the median. The whiskers are the 10th and 90th percentiles. The blue line represents the average quality score for the nucleotide. Based on these metrics, the quality scores for the first nucleotide are quite high, with nearly all reads having scores above 27. So to produce the first box plot the q values of the first base of all sequences were used. The quality scores appear to drop going from the beginning toward the end of the reads. For reads generated by Illumina sequencing, this is not unexpected. As the sequencing approaches to the end, the chemicals used in this process are getting more and more used up, which can be one of the reasons for this decrease to happen. However, in total our plot shows the good quality, since it holds that the higher the results, the better, and as we can see the blue line is mostly in the green region of the plot, except for the last three box-plots, which dropped to the medium part but really high are located in it. At q score 20, there is a 99% probability that the base is called correctly, and at the q score 30, the probability is 99.9%. Usually, the threshold is value 20. At no point the drop goes to the red part of the plot.

The **"Per sequence quality scores"** plot gives the average quality score on the x-axis and the number of sequences with that average on the y-axis. We hope the majority of our reads have a high average quality score with no large bumps at the lower quality values.

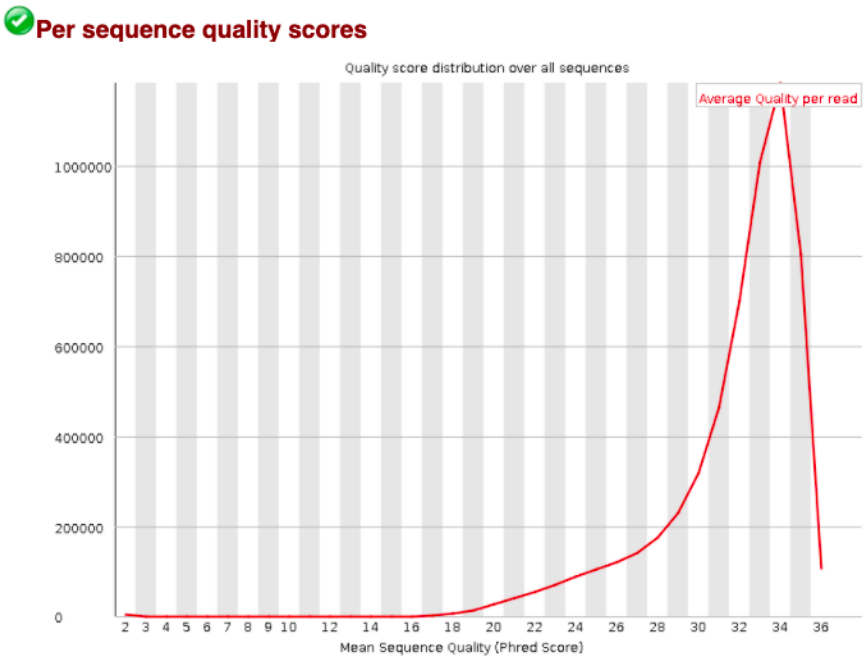


Figure 14. - Per sequence quality score plot

This data does not have any bumps at the lower quality values, and the average quality per read is quite high, around 34 Phred Score.

The next plot gives the **"Per base sequence content"**. Here for each base: alanine, guanine, thymine and cytosine, the percentage is calculated. If in our dataset we would have random sequences, then we would expect usually the flat lines for each base in the whole plot. It always

fails for RNA-seq data. This is because the first 10-12 bases result from the 'random' hexamer priming that occurs during RNA-seq library preparation. This priming is not as random as we might hope, giving an enrichment in particular bases for these initial nucleotides.

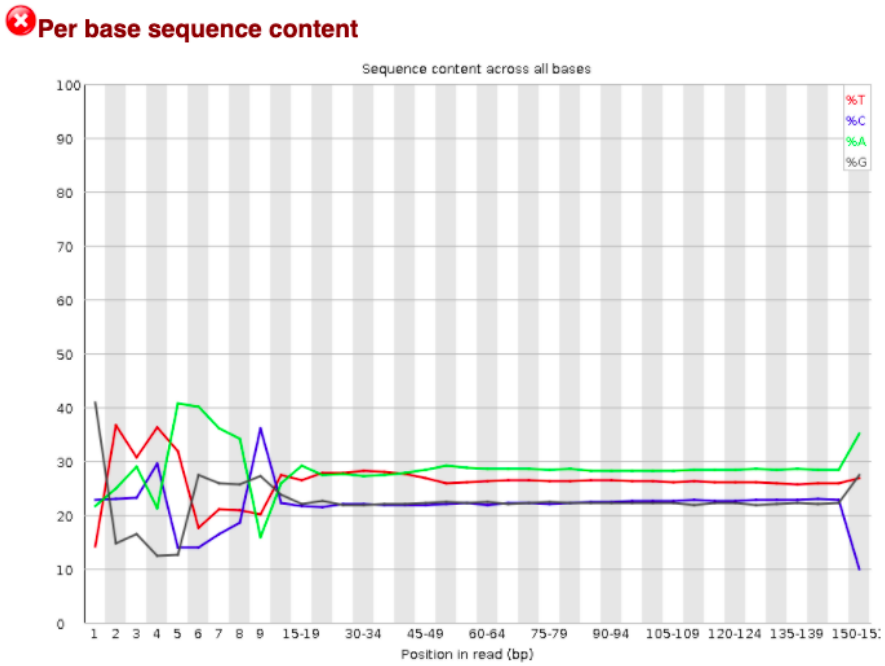


Figure 15. - Per base sequence content plot

As we can see, at the beginning we do not have flat lines, but basically a "mess", which happens if the priming is not random. Therefore it is a result of some duplicate sequences or even patterns that repeat in more sequences. This is visible only at the beginning, at the first few bases, so in our case until the 15th base pair. That is a consequence of the library preparation. Library is prepared in a way that DNA or RNA is isolated from the organism, and then nucleic acids are fragmented. They are cut into smaller pieces, depending on what size of the library we want to have. After size selection, we have in our sample only base pairs that are long that much we wanted them to be (which we did and decided in the earlier step). Then adapters, which are artificially synthesized molecules, are used as a sequencing primer, so their location tells us where the primer is attached.

The **"Per sequence GC content"** plot gives the GC distribution over all sequences. Generally it is a good idea to note whether the GC content of the central peak corresponds to the expected % GC for the organism. Also, the distribution should be normal unless overrepresented sequences (sharp peaks on a normal distribution) or contamination with another organism (broad peak).

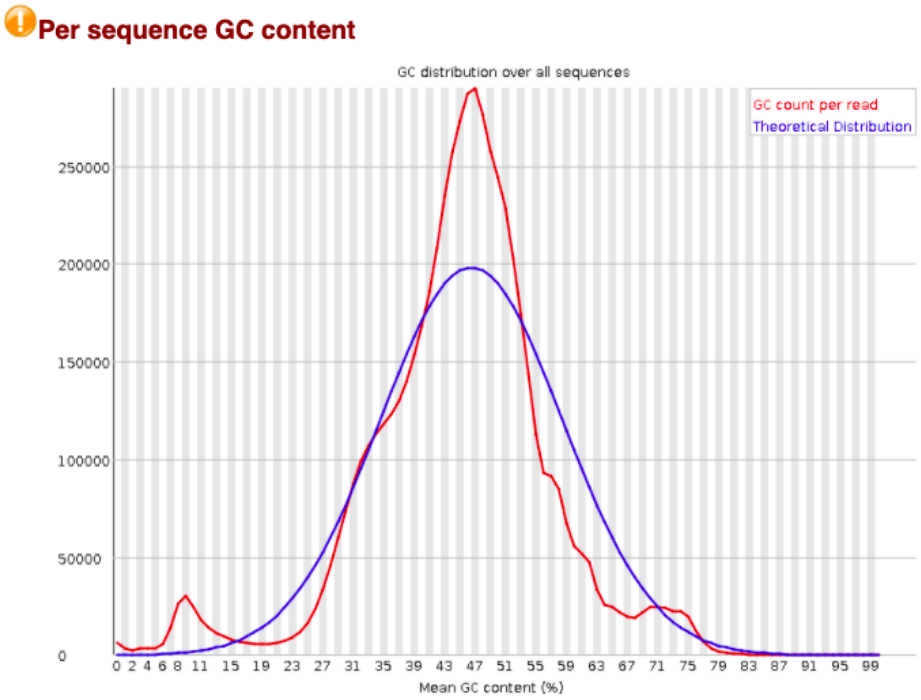


Figure 16. - Per sequence GC content distribution

This plot would indicate to some extent some type of overrepresented sequence with the sharp peak, indicating either contamination or a highly overexpressed gene. However, it is not too critical, since it mostly follows the normal distribution.

If a sequencer is unable to make a base call with sufficient confidence then it will normally substitute an N rather than a conventional base call. The module below plots out the percentage of base calls at each position for which an N was called. In our plot, there are no N bases.

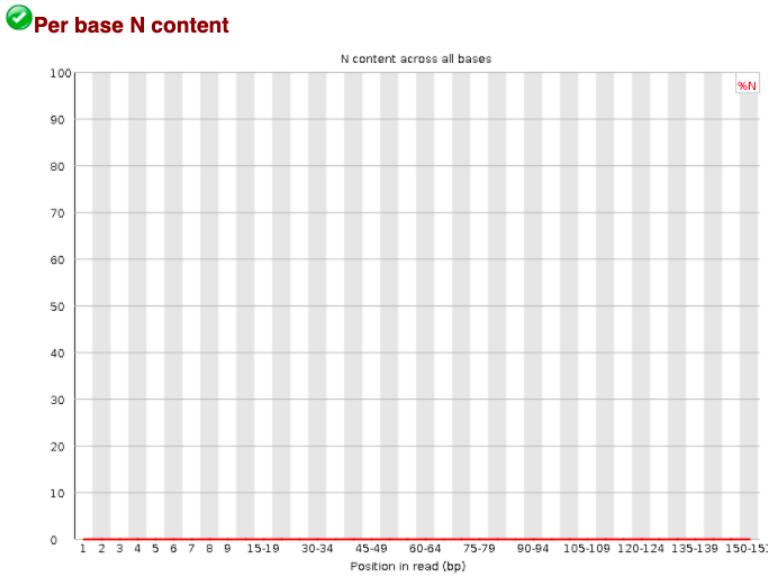


Figure 17. - Per base N content plot

The next module creates a graph showing the distribution of fragment sizes in the data file which was analysed. Some high throughput sequencers generate sequence fragments of uniform length, but others can contain reads of wildly varying lengths. Even within uniform length libraries some pipelines will trim sequences to remove poor quality base calls from the end. This module will raise a warning if all sequences are not the same length, and will raise an error if any of the sequences have zero length. Mostly this module will produce a simple graph showing a peak only at one size, but for variable length FastQ files this will show the relative amounts of each different size of sequence fragment.

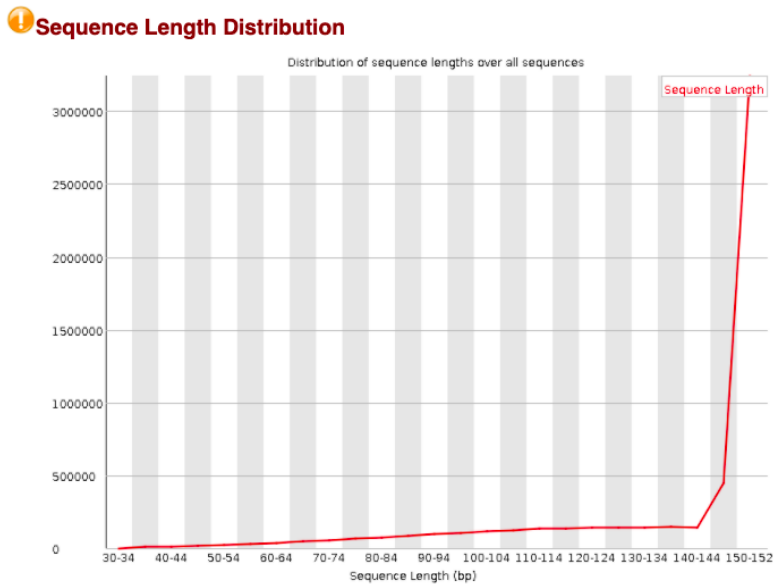


Figure 18. - Sequence length distribution plot

The next module explores numbers of duplicated sequences in the library. This plot can help identify a low complexity library, which could result from too many cycles of PCR amplification or too little starting material. For RNA-seq we don't normally do anything to address this in the analysis, but if this were a pilot experiment, we might adjust the number of PCR cycles, amount of input, or amount of sequencing for future libraries. If we observe all sequences of a dataset then we can say that around 50% of sequences that are not duplicated. However, if we are sequencing RNA, then we would have more duplications, because some genes are more expressed than others. In this analysis we seem to have a large number of duplicated sequences.

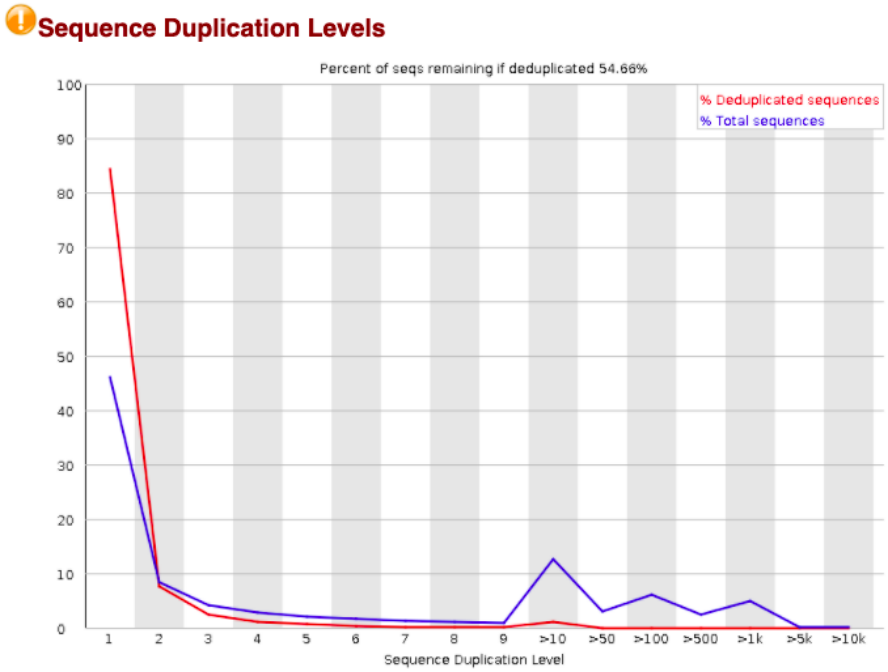


Figure 19. - Sequence duplication levels

The **"Overrepresented sequences"** table is another important module because it shows sequences, or patterns, that occur in more than 0.1% of the total number of sequences. This table aids in identifying contamination, such as vector or adapter sequences. If the %GC content was off in the above module, this table can help identify the source. If not listed as a known adapter or vector, it can help to BLAST the sequence to determine the identity.

Overrepresented sequences

Sequence	Count	Percentage	Possible Source
GTATCAACGCAGAGTACTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT	14828	0.2604759535794891	No Hit
GGTATCAACGCAGAGTACTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT	9065	0.15924025621783577	No Hit
TATCAACGCAGAGTACTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT	7794	0.1369132440112313	No Hit

Figure 20. - Overrepresented sequences table

As we can see in all three sequences, we have a long sequence of "T". It probably refers to the primer which is used.

The next and final module generates a plot that shows a cumulative percentage count of the proportion of our library which has seen each of the adapter sequences at each position. Once a sequence has been seen in a read it is counted as being present right through to the end of the read so the percentages we see will only increase as the read length goes on. This module will issue a warning if any sequence is present in more than 5% of all reads, and will issue an error if any sequence is present in more than 10% of all reads. In our case, a solid small RDA adapter is used throughout the whole sequence.

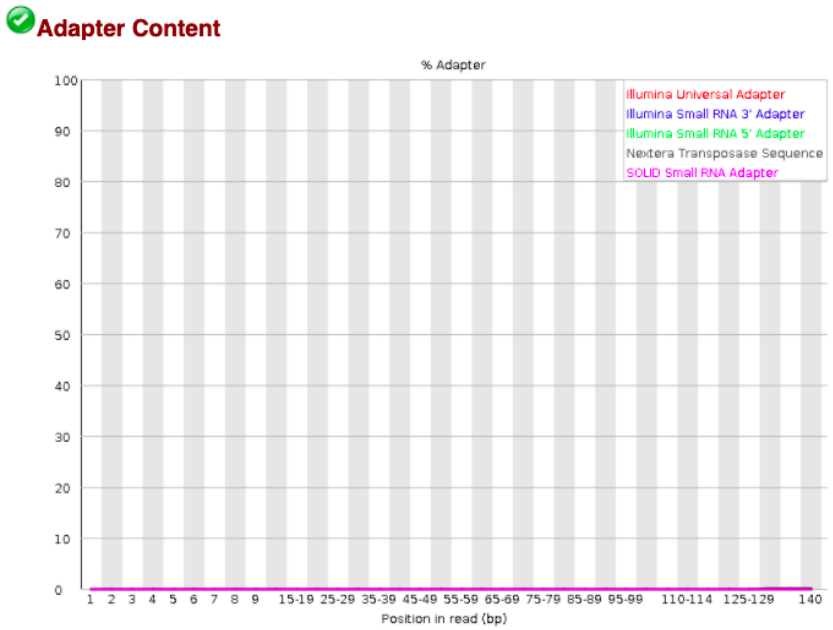


Figure 21. - Adapter content plot

SECOND FILE ANALYSIS:

As I mentioned earlier, the basic statistics part is the same for both files, except for the file name. Therefore we move to the second module.

Based on these metrics, the quality scores for the first nucleotide are quite high, with nearly all reads having scores above 27. The quality scores appear to drop going from the beginning toward the end of the reads. However, in total our plot shows the good quality, since it holds that the higher the results, the better, and as we can see the blue line is mostly in the green region of the plot, except for the last three box-plots, which dropped to the medium part but really high are located in it. So mostly the graph is really similar to the one from the first file, except for the minimal value of the box-plots in the middle part of the graph, which are now a little bit lower.

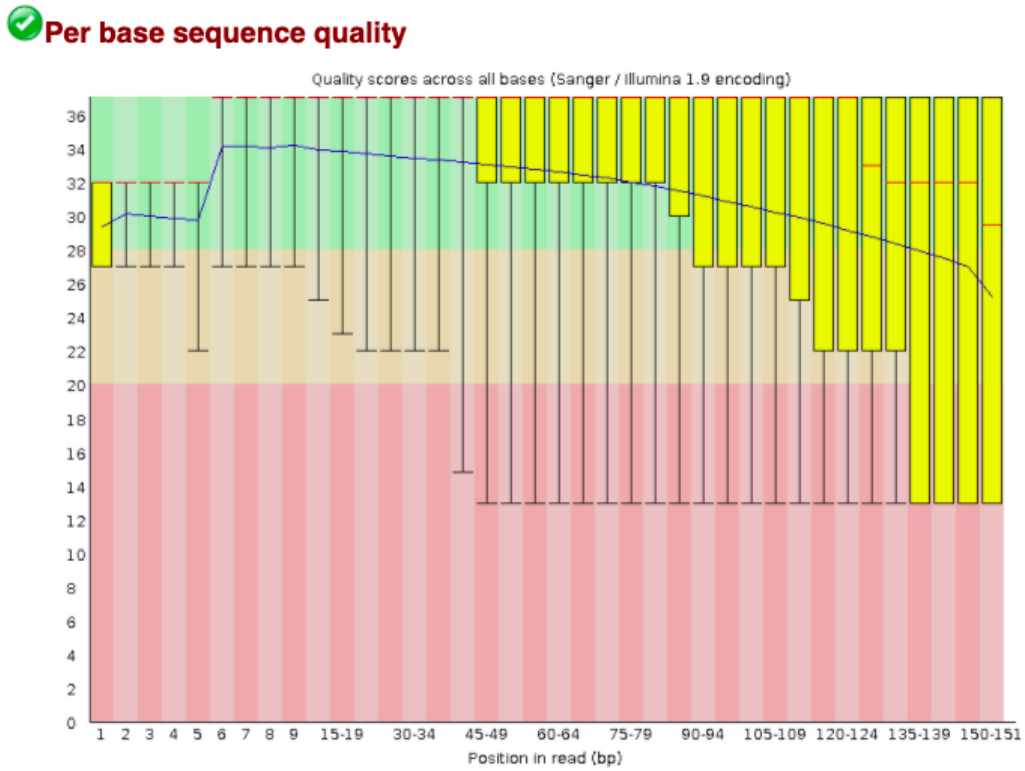


Figure 22. - Per base sequence quality plot

The per sequence quality scores plot is almost identical, with a little bit lower peak this time, and the same average quality Phred score, which is 34.

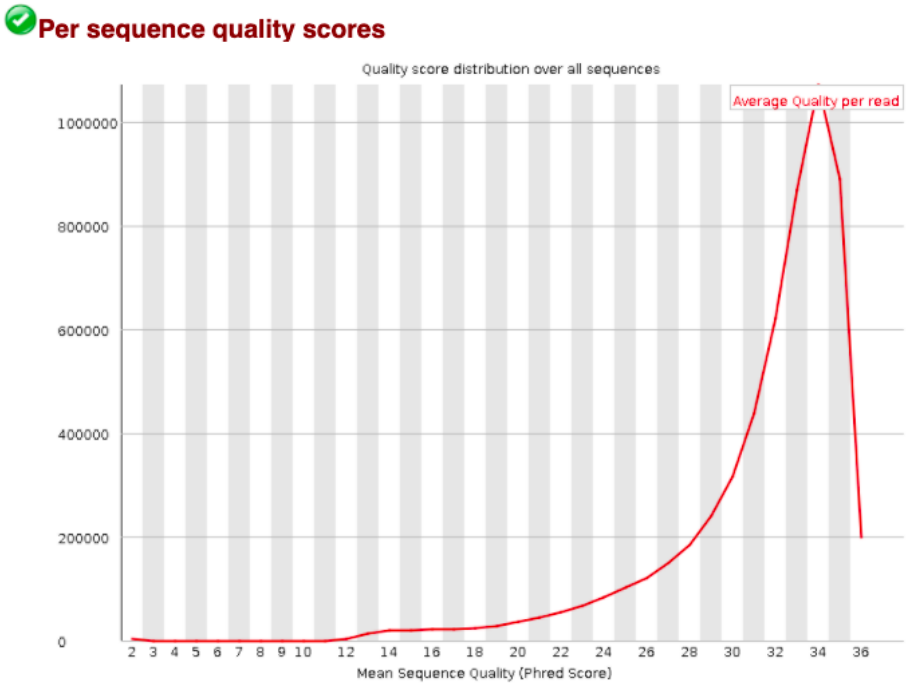


Figure 23. - Per sequence quality score plot

In per base sequence content we visualize the same as in the first file, which is the biased part at the beginning until the fifteenth base pair and after that a flatten line for each base. The same, almost identical graphs as in the first file we identify in the following six modules.

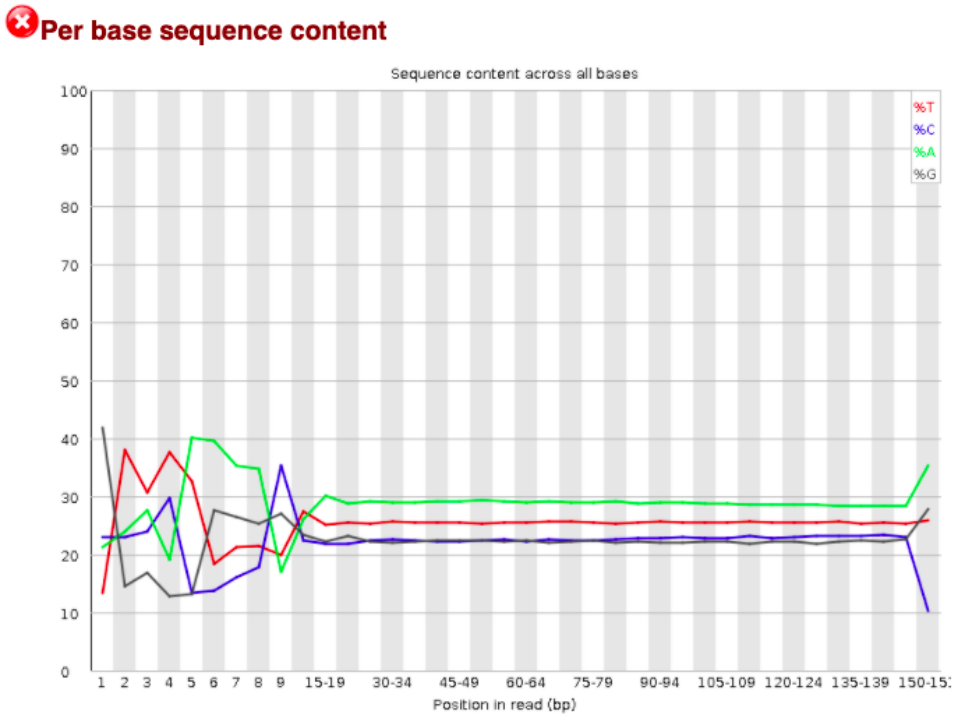


Figure 24. - Per base sequence content plot

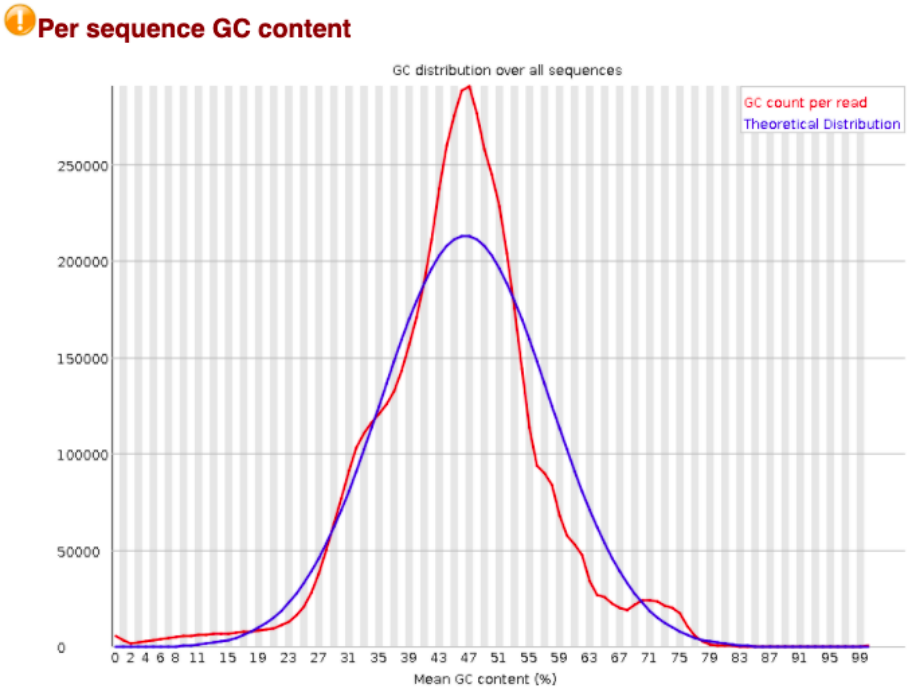


Figure 25. - Per sequence GC content

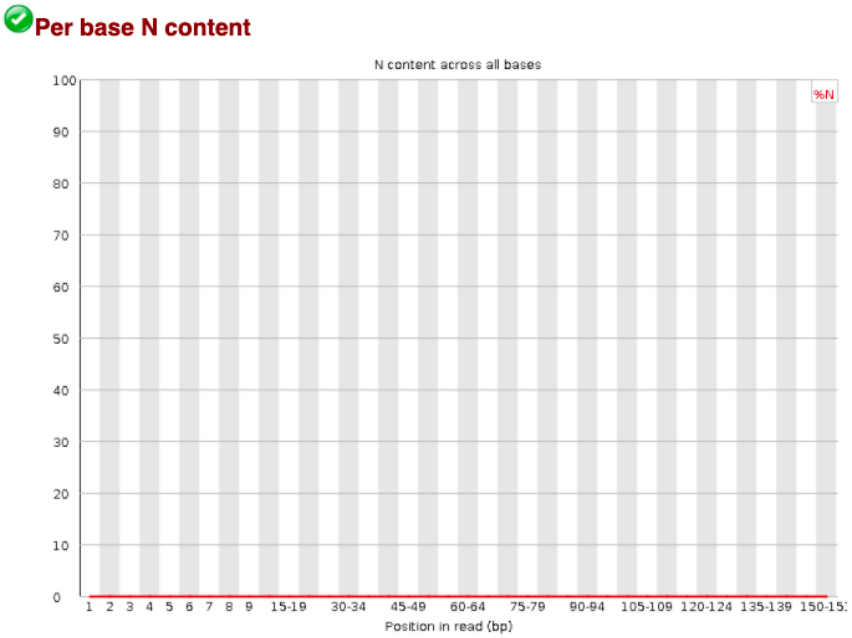


Figure 26. - Per base N content

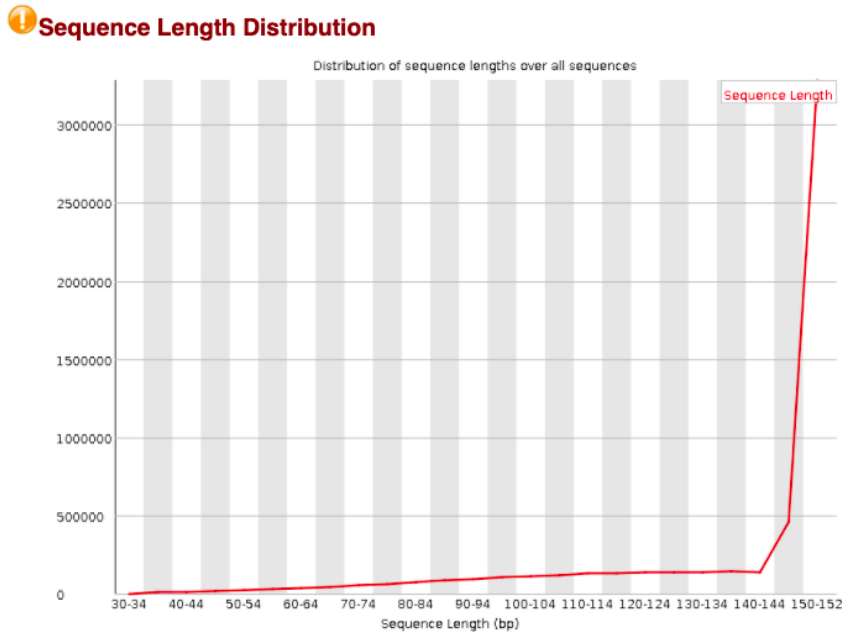


Figure 27. - Sequence length distribution

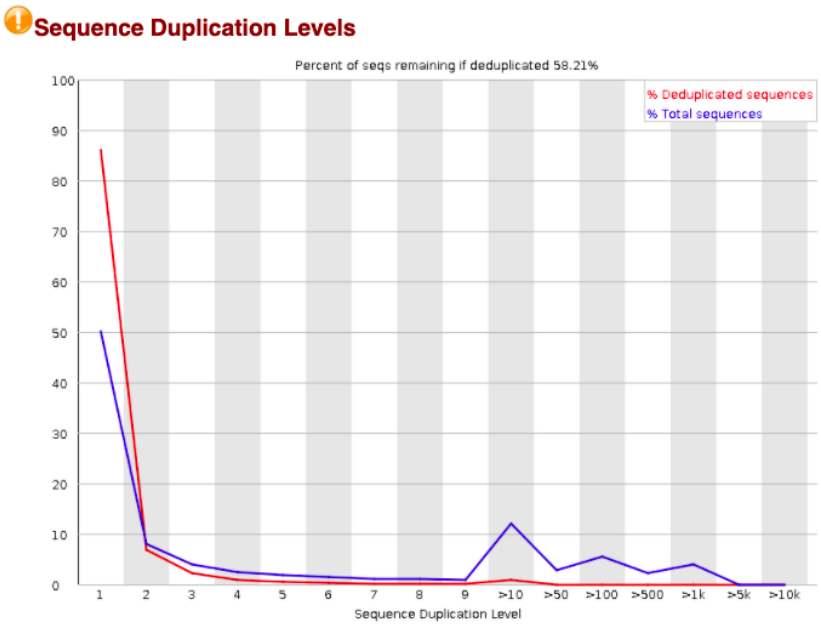


Figure 28. - Sequence duplication levels

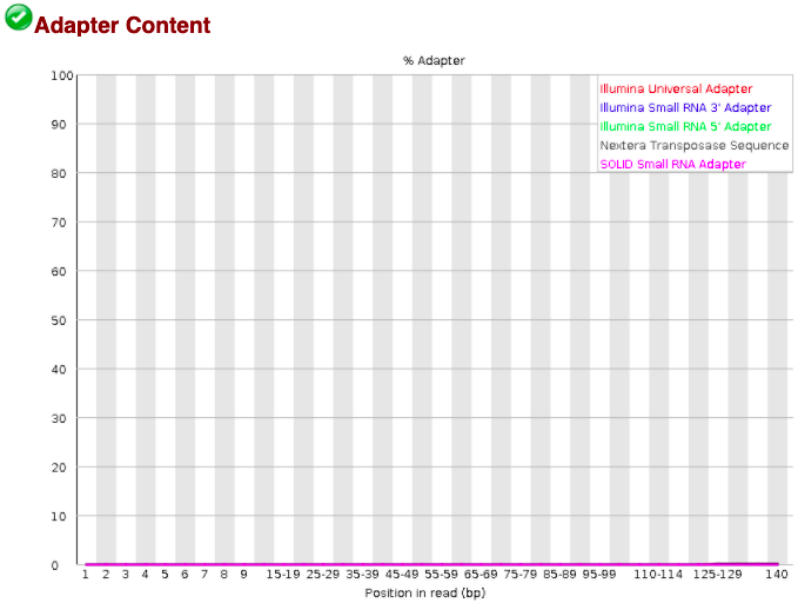


Figure 29. - Adapter content

The only difference that was already visible in the report's summary is in the overrepresented sequences module, where it is said that there are no such sequences.

Figure 30. - Overrepresented sequences

MultiQC is a tool to create a single report with interactive plots for multiple bioinformatics analyses across many samples and is written in Python. Reports are generated by scanning given

directories for recognised log files. These are parsed and a single HTML report is generated summarising the statistics for all logs found. MultiQC reports can describe multiple analysis steps and large numbers of samples within a single plot, and multiple analysis tools making it ideal for routine fast quality control.

I installed MultiQC using conda from bioconda channel:
$ conda install -c bioconda multiqc

In order to perform multiqc I placed wanted html file reports in one folder. Then we enter the folder with as usual "cd" function, and do the following:

$ multiqc .

This creates a new html file with all combined data in it.

PART 2

FILTERING BASED ON QUALITY PARAMETERS, TRIMMING, ETC.

In this step we will check the presence of primer, adapters and remove low confidence bases. Two frequently used tools are Cutadapt and Trimmomatic.

Trimmomatic is a fast, multithreaded command line tool that can be used to trim and crop Illumina (FASTQ) data as well as to remove adapters. These adapters can pose a real problem depending on the library preparation and downstream application. There are two major modes of the program: Paired end mode and Single end mode. The paired end mode will maintain correspondence of read pairs and also use the additional information contained in paired reads to better find adapter or PCR primer fragments introduced by the library preparation process. Trimmomatic works with FASTQ files.

The current trimming steps are:

- ILLUMINACLIP: Cut adapter and other illumina-specific sequences from the read.
- SLIDINGWINDOW: Performs a sliding window trimming approach. It starts scanning at the 5‟ end and clips the read once the average quality within the window falls below a threshold.
- MAXINFO: An adaptive quality trimmer which balances read length and error rate to maximise the value of each read
- LEADING: Cut bases off the start of a read, if below a threshold quality
- TRAILING: Cut bases off the end of a read, if below a threshold quality
- CROP: Cut the read to a specified length by removing bases from the end
- HEADCROP: Cut the specified number of bases from the start of the read
- MINLEN: Drop the read if it is below a specified length
- AVGQUAL: Drop the read if the average quality is below the specified level

- TOPHRED33: Convert quality scores to Phred-33
- TOPHRED64: Convert quality scores to Phred-64
- PE: paired end reads

$ trimmomatic PE SRR7287369_pass_1.fastq SRR7287369_pass_2.fastq SRR7287369_pass_1_paired.fastq SRR7287369_pass_1_unpaired.fastq SRR7287369_pass_2_paired.fastq SRR7287369_pass_2_unpaired.fastq ILLUMINACLIP:NexteraPE-PE.fa:2:30:10:2:keepBothReads LEADING:3 TRAILING:3 MINLEN:36 SLIDINGWINDOW:4:15

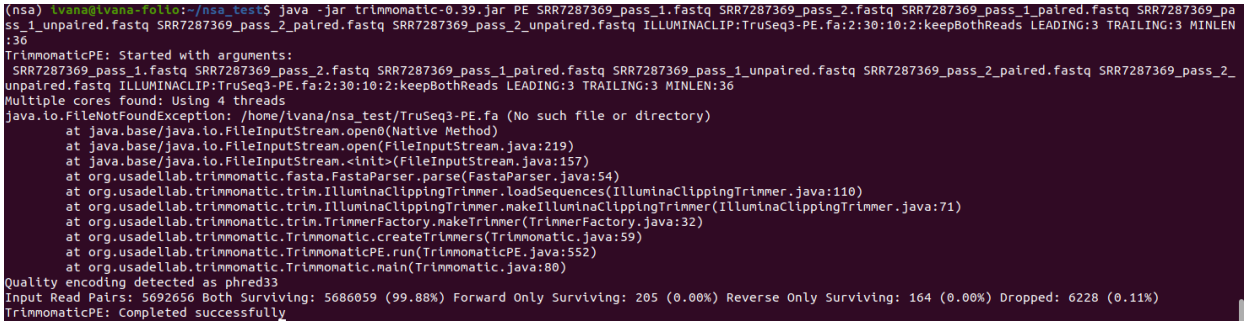


Figure 1. - Completing trimmomatic function

Using Nextera with adaptor sequences we manage to remove overrepresented parts of the sequences. Also, the number of total sequences decreased, and some bases were removed, the end of the read was improved due to a sliding window.

I would like to mention that I created a new folder in my home, where I placed all trimmed sequences, both paired and unpaired.

ALIGNMENT USING STAR

STAR is a splice-aware mapper (meaning that it aligns RNA-Seq data on genomes of eukaryotes considering introns).

a. Examine GTF/GFF file. Which information can be found in these files?

GFF stands for General Feature Format, and GTF for General Transfer Format. The GFF format consists of one line per feature, each containing 9 columns of data, plus optional track definition lines. The GTFis identical to GFF version 2. Fields must be tab-separated. Also, all but the final field in each feature line must contain a value; "empty" columns should be denoted with a '.' The GFF and GTF formats are used for annotating genomic intervals (an interval with begin/end position on a contig/chromosome).

1. seqname - name of the chromosome or scaffold. Chromosome names can be given with or without the 'chr' prefix. The seqname must be one used within Ensembl, so a standard chromosome name or an Ensembl identifier such as a scaffold ID,

without any additional content such as species or assembly.

2. source - name of the program that generated this feature, or the data source (database or project name)
3. feature - feature type name
4. start - Start position* of the feature, with sequence numbering starting at 1.
5. end - End position* of the feature, with sequence numbering starting at 1.
6. score - A floating point value.
7. strand - defined as + (forward) or - (reverse).
8. frame - One of '0', '1' or '2'. '0' indicates that the first base of the feature is the first base of a codon, '1' that the second base is the first base of a codon, and so on..
9. attribute - A semicolon-separated list of tag-value pairs, providing additional information about each feature.

b. What is the difference between GTF/GFF2/GFF3 file formats?

GFF exists in versions 2 and 3 and GTF is sometimes called “GFF 2.5”. GFF2 is a supported format in GMOD, but it is now deprecated. If we have a choice we should use GFF3. Unfortunately, data is sometimes only available in GFF2 format. GFF2 has a number of shortcomings compared to GFF3. GFF2 can only represent 2 level feature hierarchies, while GFF3 can support arbitrary levels. GFF2 also does not require that column 3, the feature type, be part of the sequence ontology. It can be any string. This often led to quality control and data exchange problems. GFF3 is the preferred format in GMOD. Therefore the two versions are similar but are not compatible and scripts usually only work with one of the other formats.

PRACTICAL PART

1. $ STAR --runThreadN 1 --runMode genomeGenerate --genomeDir . --genomeFastaFiles Arabidopsis_thaliana.TAIR10.dna.toplevel.fa --sjdbGTFfile Arabidopsis_thaliana.TAIR10.51.gtf --sjdbOverhang 149 --genomeSAindexNbases 12

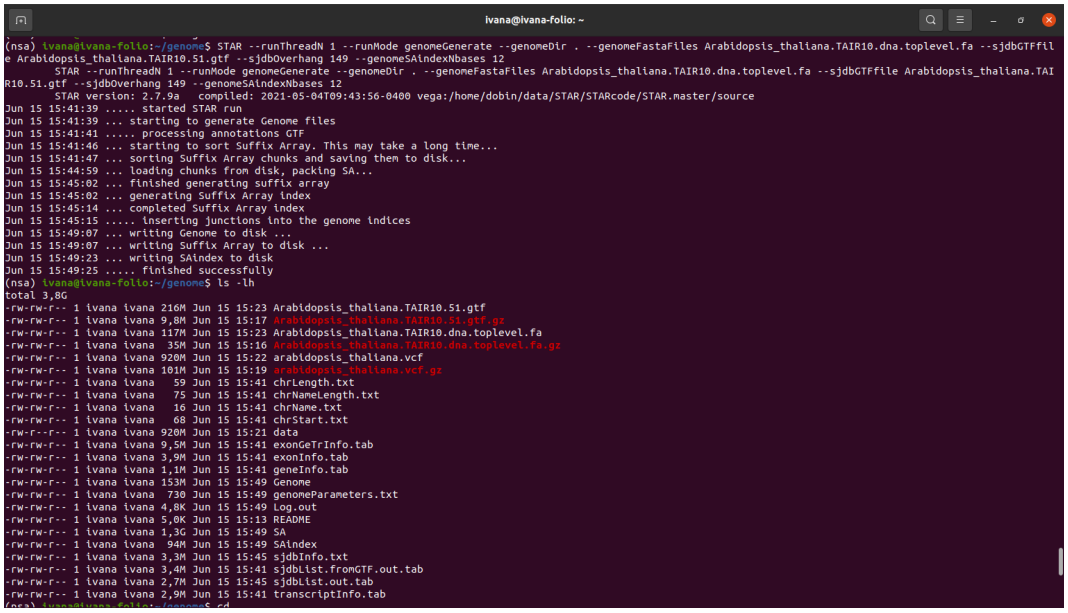


Figure 2. - Conducting STAR function

- --runThreadN NumberOfThreads
  Defines the number of threads to be used for genome generation, it has to be set to the number of available cores on the server node.
- --runMode genomeGenerate
  Directs STAR to run genome indices generation jobs.
- --genomeDir /path/to/genomeDir
  Specifies path to the directory where the genome indices are stored. This directory path will have to be supplied at the mapping step to identify the reference genome.
- --genomeFastaFiles /path/to/genome/fasta1 /path/to/genome/fasta2 ...
  Specifies one or more FASTA files with the genome reference sequences. Multiple reference sequences are allowed for each fasta file.
- --sjdbGTFfile /path/to/annotations.gtf
  Specifies the path to the file with annotated transcripts in the standard GTF format. STAR will extract splice junctions from this file and use them to greatly improve accuracy of the mapping. While this is optional, and STAR can be run without annotations, using annotations is highly recommended whenever they are available.
- --sjdbOverhang ReadLength-1
  Specifies the length, which should be equal to the ReadLength-1, where ReadLength is the length of the reads, of the genomic sequence around the annotated junction to be used in constructing the splice junctions database.
- --genomeSAindexNbases
  For small genomes, this parameter must be scaled down, with a typical value of min(14, log2(GenomeLength)/2 - 1).

2. $ STAR --runThreadN 1 --genomeDir ../genome --readFilesIn ../trimmed/SRR7287369_pass_1_paired.fastq ../trimmed/SRR7287369_pass_2_paired.fastq --outSAMattrRGline ID:SRR7287369 --outFileNamePrefix STAR_mapped_SRR7287369_

- --readFilesIn name(s) (with path) of the files containing the sequences to be mapped. If using Illumina paired-end reads, the read1 and read2 files have to be supplied. STAR can process both FASTA and FASTQ files. Multi-line FASTA files are supported.
- --outSAMattrRGline string(s): SAM/BAM read group line. The first word contains the read group identifier and must start with ”ID:”, e.g. –outSAMattrRGline ID:xxx CN:yy ”DS:z z z”. xxx will be added as an RG tag to each output alignment.
- --outFileNamePrefix
  STAR produces multiple output files. All files have a standard name, however, we can change the file prefixes using this parameter.

This parameter created five different files. STAR has the option to run mapping in two phases. The first phase does the above, so creates files, and in the second phase these files are taken into account. In the first phase STAR identifies if there were any new splice junctions, and then based on these newly idefities junctions the remapping is done.

One of the files is “_Log. final.out”. This file has information such as, when the job and mapping is started, when it was finished, as well as mapping speed, which is in millions of reads per hour. Besides that, there is the number of input reads, average input read length, and the following information refers to the unique reads. For that we have the number of uniquely mapped reads and it’s percentage, the average mapped length, the total number of splices, the number of annotated splices, the number of non-canonical splices, etc. There is also information regarding the multi-mapping reads, unmapped reads and chimeric reads.
The other file created is the “.sam” file, which has a header and alignment section. The size of the same file is usually really big, and because of that the bam file is preferred.

PART 3

CONVERTING SAM TO BAM FORMAT AND SORTING SEQUENCES ACCORDING TO THE GENOMIC COORDINATES

First we need to install samtools, which is a suite of program for interacting with high-throughput sequencing data.

1. $ conda install -c bioconda samtools

2. $ samtools view -Sb -@ 1 STARAligned.out.sam > STARAligned.out.bam

- -@ 1: defining number of threads

This function is used to convert sam to bam file, using the samtools.
Then we can compare the sizes of two files, which is huge. The size of the sam file was 3.4G and the size of the bam file is only 881M, which is a lot smaller when compared with sam file.
We again use samtools to view bam files, since the cat function cannot be applied to the bam file, which allows us to see the aligned part of the file.

3. $ samtools sort -@ 1 STARAligned.out.bam -o STARAligned.out.sorted.bam

- -o FILE: write final output to FILE rather than standard output

This function creates a new file, which is basically a sorted bam file.

4. $ samtools view STARAligned.out.sorted.bam | less

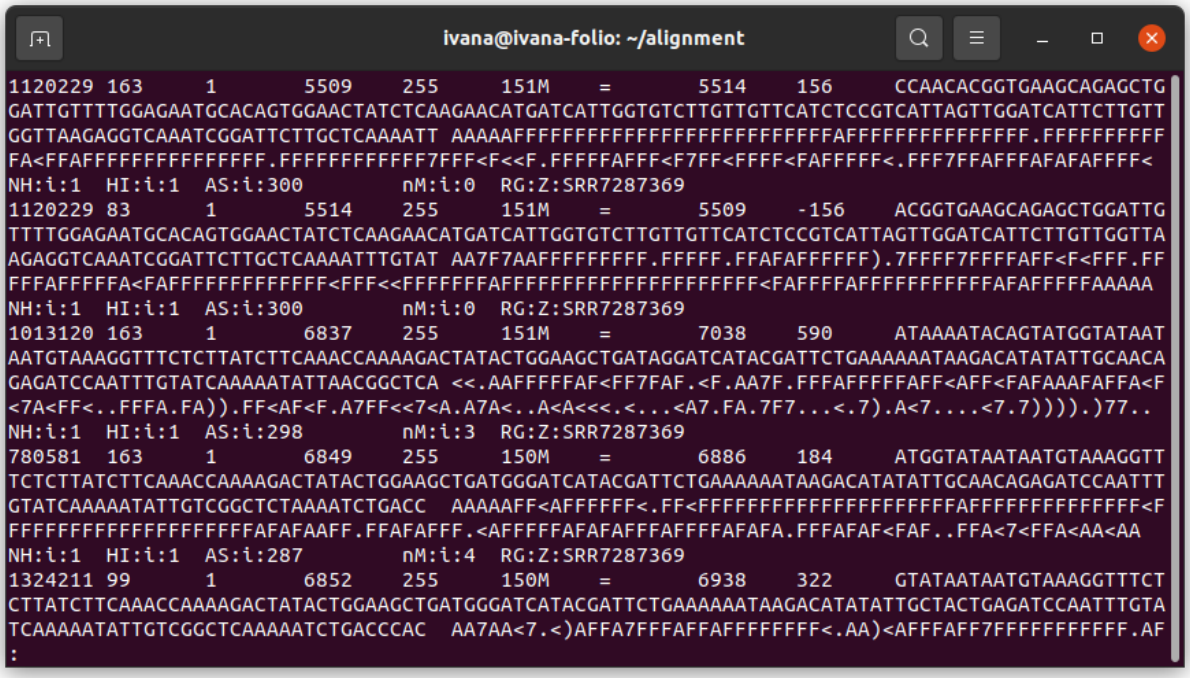


Figure 1. - Opened sorted bam file of aligned sequences

5. $ samtools index STARAligned.out.sorted.bam

This function creates the index of the bam file, which has an extension “bai”.

6. $ samtools view -b STARAligned.out.sorted.bam 1 > STARAligned.out.sorted.CHR1.bam

- -b: Saves the output in bam format

This function created a new sorted file, which includes only information regarding the first chromosome.

- 1: Extract only those information regarding the first chromosome

7. $ samtools index STARAligned.out.sorted.CHR1.bam

CHECK THE ALIGNMENT WITH TABLET OR IGV

Tablet is a lightweight, high performance graphical viewer for next generation sequence assemblies and alignments.

First we need to install it using the conda package.

1. $ conda install -c bioconda tablet

We have 5 chromosomes, mitochondrial and chloroplast DNA on our reference genome, but the reads are only present on the first chromosome. We can select each read and see the sequence ID, length. Besides that we can also see how the read mapped reference genome, in cigar. All reads align in exon regions.

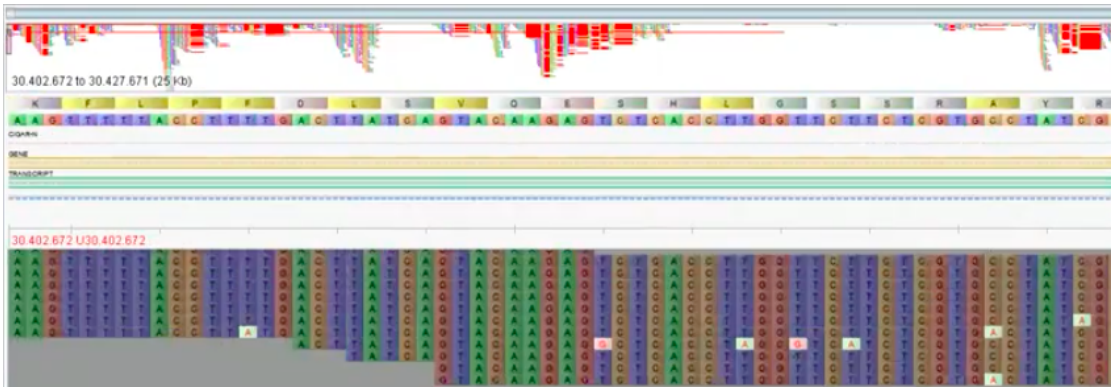


Figure 2. - User interface with opened files we worked with

Figure 3. - Close up look on specific read with more information provided

PART 4

COUNT THE NUMBER OF READS USING FEATURECOUNTS

Bioinformatic tool featureCounts will be used for counting the aligned reads per gene. Before running the featureCounts it should be checked if your dataset is strand-specific or not. You can do this with infer_experiment.py (part of RSeQC package, http://rseqc.sourceforge.net/#).

RSeqQC requires a bam file and a bed file. Bed files can be produced with BEDOPS, USING gtf2bed function.

1. $ wget http://hgdownload.cse.ucsc.edu/admin/exe/linux.x86_64/gtfToGenePred
2. $ wget http://hgdownload.cse.ucsc.edu/admin/exe/linux.x86_64/genePredToBed

   The above two functions we used to obtain two scripts: genePredToBed and gtfToGenePred. We make those two files executable them with the following two commands:

3. $ chmod +x genePredToBed
4. $ chmod +x gtfToGenePred
5. $ ./gtfToGenePred Arabidopsis_thaliana.TAIR10.51.gtf Arabidopsis_thaliana.TAIR10.51.genePred
   This function converts gtfToGenePred to genePred.
6. $ ./genePredToBed Arabidopsis_thaliana.TAIR10.51.genePred Arabidopsis_thaliana.TAIR10.51.bed12
   This function converts genePred to bed12
7. $ infer_experiment.py -r ../genome/Arabidopsis_thaliana/TAIR10.51.bed12 -i STAR_mapped_SRR7287369_Aligned.sortedByCoord.out.bam

- -i: inserting
- infer_experiment.py: Checks if the dataset is strand-specific or not

```
This is PairEnd Data
Fraction of reads failed to determine: 0.0662
Fraction of reads explained by "1++,1--,2+-,2-+": 0.4605
Fraction of reads explained by "1+-,1-+,2++,2--": 0.4733
```

Figure 1. - Result of infer_experiment

From the above results, we can conclude that we have non strand specific reads.

We checked these results and compared them with the analysis of the project we are following, and they wrote there that actually the reads are non strand specific. Therefore we did the analysis correctly.

8. $ conda install subread

The data input to **featureCounts** consists of:
-one or more files of aligned reads (short or long reads) in either SAM or BAM format
-a list of genomic features in either Gene Transfer Format (GTF) or General Feature Format (GFF) or Simplified Annotation Format (SAF)
The format of input reads is automatically detected (SAM or BAM). If the input contains location-sorted paired-end reads, featureCounts will automatically re-order the reads to place next to each other the reads from the same pair before counting them.

9. $ featureCounts -a ../genome/Arabidopsis_thaliana.TAIR10.51.gtf -o STAR_mapped_SRR7287369_Aligned.sortedByCoord ../alignment/STAR_mapped_SRR7287369_Aligned.sortedByCoord.out. bam

- -a < string > (annot.ext, annot.inbuilt): Provide name of an annotation file. Gzipped file is accepted.
- -o: Give the name of the output file. The output file contains the number of reads assigned to each meta-feature (or each feature if -f is specified). Note that the featureCounts function in Rsubread does not use this parameter. It returns a list object including read summarization results and other data.

10. $ featureCounts -a Arabidopsis_thaliana.TAIR10.51.gtf -s 0 -o STAR_mapped_SRR7287369_Aligned.sortedByCoord_0 STAR_mapped_SRR7287369_Aligned.sortedByCoord.out.bam

- -s < intorstring > (isStrandSpecific): Indicate if strand-specific read counting should be performed. A single integer value (applied to all input files) or a string of comma-separated values (applied to each corresponding input file) should be provided. Possible values include: 0 (unstranded), 1 (stranded) and 2 (reversely stranded). Default value is 0 (ie. unstranded read counting carried out for all input files). For paired-end reads, the strand of the first read is taken as the strand of the whole fragment.

This function gives the output in the terminal, with the title subread. It has two parts: featureCounts settings and Reading.

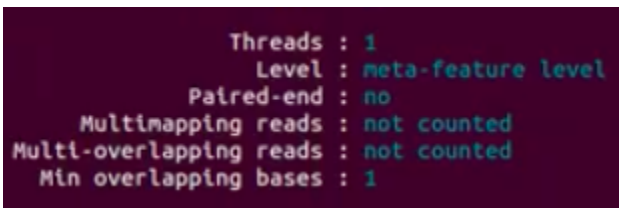


Figure 2. - featureCounts settings part

In the first part, besides the names of input, output file and annotation, there is more information presented above in the image. Also, below are two figures showing the second, running part of the output.

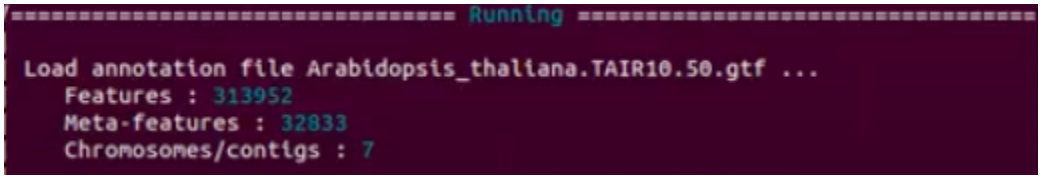


Figure 3. - Running part of featureCount related to input

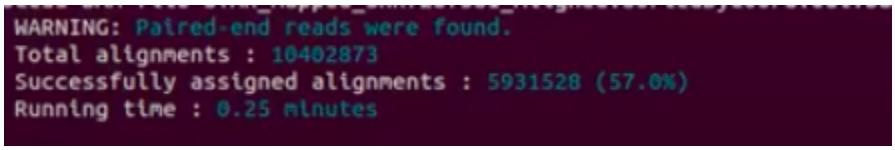


Figure 4. - Running part of featureCounts related to the process

This function created a new file with the extension ".summary" in which we can find the summary with all the needed information.

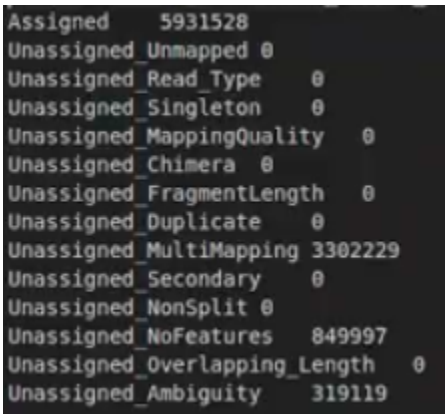


Figure 5. - Opened summary file of performed featureCons

As we can see these results show us that there are 5931528 assigned, 3302229 unassigned multi mappings, 849997 unassigned with no features and 319119 unassigned ambiguities.

11. $ featureCounts -aArabidopsis_thaliana.TAIR10.51.gtf -s 0 -O -p -o STAR_mapped_SRR7287381_Aligned.sortedByCoord_0 STAR_mapped_SRR7287381_Aligned.sortedByCoord.out.bam

- -O: If specified, reads will be allowed to be assigned to more than one matched meta-feature. Reads/fragments overlapping with more than one meta-feature/feature will be counted more than once. Note that when performing meta-feature level summarization, a read will still be counted once if it overlaps with multiple features within the same meta-feature. Also note that this parameter is applied to each individual alignment when there are more than one alignment reported for a read.

- -p (isPairedEnd): If specified, fragments (or templates) will be counted instead of reads. This option is only applicable for paired-end reads.

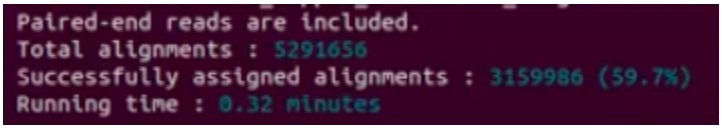


Figure 6. - Running output of the above function

12. $ featureCounts -a ../genome/Arabidopsis_thaliana.TAIR10.51.gtf -s 0 -O -p -f -o STAR_mapped_SRR7287381_Aligned.sortedByCoord_0_exon STAR_mapped_SRR7287381_Aligned.sortedByCoord.out.bam

- -f: If specified, read summarization will be performed at feature level (eg. exon level). Otherwise, it is performed at meta feature level (eg. gene level).

This function also creates an output file with the extension ".exon" in which for each exon we can find the number of reads.

PART 5

VARIANT DISCOVERY WITH GATK

First we needed to do the INSTALLING of the DOCKER using the steps from docker docs.

After this, we need to download GATK as a docker image. The GATK is the industry standard for identifying SNPs and indels in germline DNA and RNAseq data.

1. $ sudo docker pull broadinstitute/gatk

RNAseq short variant discovery (SNPs + Indels)

Identifying short variants (SNPs and Indels) in RNAseq data has three parts, which are:

- Data cleanup
- Variant discovery
- Evaluation

The complete steps of the process can be seen in the image below.

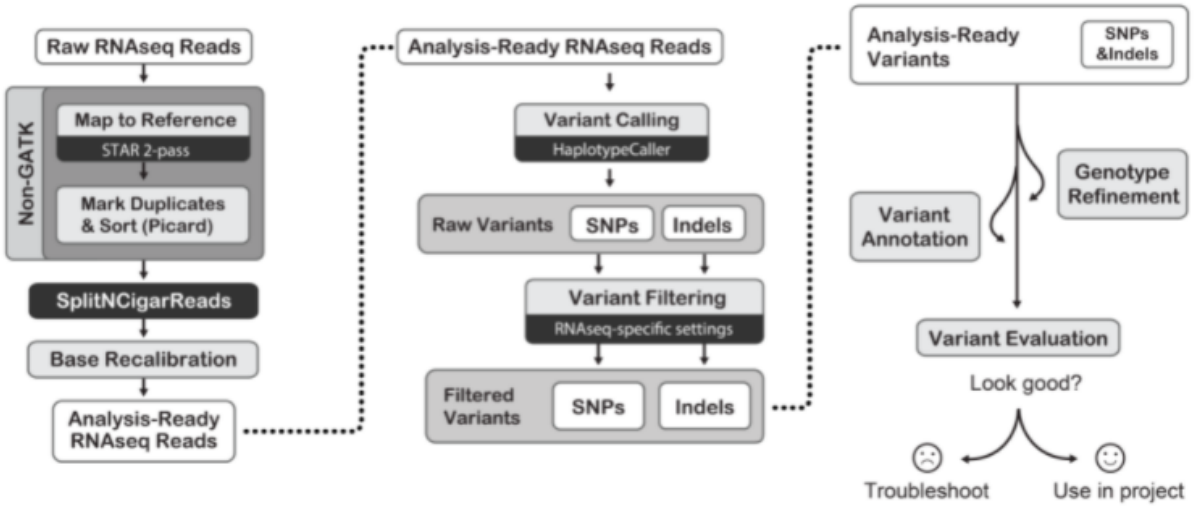


Figure 1. - RNAseq short variant discovery process

Since this part is pretty demanding, as the professor agreed with us, we will use the other sequence, not the one allocated to us. So in the following exercises we will use the one with this accession number: SRR7287361.

The first step is to mark duplicates. Before sequencing of cDNA or DNA, it should be amplified in a PCR reaction. Before sequencing there exist some identical reads present and in order to more correctly analyse the ratio between reads with mutations those duplicate reads should be identified and discarded.

1. $ sudo docker images

   We use this function to list all the images present and in order to identify the ID of the gatk image, which is e1a358670b4f.

2. $ sudo docker run --rm -name GATK_1 -v /home/ivana/nsa:/gatk -it e1a358670b4f bash

- --rm : Removing of the container
- -name: Assigning the name to the container
- -v: Connecting two folders
- -it: Installing

This function creates a gatk docker container and makes it possible to access all tools required for the further analysis.

After we enter the container, each command can be run with the use of the "./gatk commandName". However, in order to be able to do this we first use the -ls function to list all the files in the container and look for the "gatk-completion.sh".

When we tried to identify the duplicates, we got an error, which claimed that the read group that was added during the mapping was not compatible or similar, hence we needed first to solve this.

We will use AddOrReplaceReadGroups, which assigns all the reads in a file to a single new read-group. This tool accepts INPUT BAM and SAM files. We use this function to add a read group. This way we can assign all the reads in the input file to a single new read-group.

3. $ ./gatk AddOrReplaceReadGroups -I alignment/STAR_mapped_SRR7287381_withoutRG_ Aligned.out.sorted_RGadded.bam -LB Arabidopsis_roots -PL Illumina -PU unit1 -SM SRR7287381 STAR_mapped_SRR7287381_withoutRG_Aligned.out.sorted.RG.bam STAR_mapped_SRR7287381_withoutRG_Aligned.out.sorted.bam

- -I: Input file (BAM or SAM or a GA4GH url)
- -LB: Read-Group library (BAM or SAM)
- -PL: Read-Group platform
- -PU: Read-Group platform unit
- -SM: Read-Group sample name

If this does not work we can also use samtools to remove read group from each sequence and we do that as following:

4. $ samtools view -h STAR_mapped_SRR7287361.Aligned.out.sorted.bam | grep - v “^@RG” | sed "s/\tRG:Z:[^\t]*//" | STAR_mapped_SRR7287381_Aligned.out.sorted_withoutRG.bam -
5. $ ./gatk AddOrReplaceReadGroups -I alignment/STAR_mapped_SRR7287381_withoutRG_Aligned.out.sorted_RGadded.bam -LB Arabidopsis_roots -PL Illumina -PU unit1 -SM SRR7287381
6. $ samtools view -h STAR_mapped_SRR7287381_withoutRG_Aligned.out.sorted_RGadded.bam | less

When we take a look at the file now, after finishing the above we can see that RG created with ID: 1, LB (library): Illumina, SM: SRR7287381 and PU: unit1.

7. $ ./gatk MarkDuplicates -I alignment/STAR_mapped_SRR7287381_withoutRG_Aligned.out.sorted_RGadded.bam -M alignment/STAR_mapped_SRR7287381_withoutRG_Aligned.out.sorted_RGadded.MD.txt -O alignment/STAR_mapped_SRR7287381_withoutRG_Aligned.out.sorted_RGadded.MD.bam --READ_NAME_REGEX null

- -M: Specifying the output file as text file
- --READ_NAME_REGEX: MarkDuplicates can use the title and cluster positions to estimate the rate of optical duplication in addition to the dominant source of duplication, PCR, to provide a more accurate estimation of library size. By default it will attempt to extract coordinates using split on. Set READ_NAME_REGEX to null to disable optical duplicate detection.

In addition, we know that STAR is a splicing aware alignment tool. That means that it can align one read on two regions of the genome. Therefore it can recognize where the intron is located and based on that perform alignment.

Now we can come to the CIGAR alignment. The sequence being aligned to a reference may have additional bases that are not in the reference or may be missing bases that are in the reference. The CIGAR string is a sequence of base lengths and the associated operation. They are used to indicate things like which bases align (either a match/mismatch) with the reference, are deleted from the reference, and are insertions that are not in the reference.

8. $ ./gatk SplitNCigarReads -I alignment/STAR_mapped_SRR7287381_withoutRG_Aligned.out.sorted_RGadded.MD.bam -O alignment/STAR_mapped_SRR7287381_withoutRG_Aligned.out.sorted_RGadded.MD.split.bam -R genome/Arabidopsis_thaliana.TAIR10.dna.toplevel.fa -L 1

- -R: Reference sequence file
- -L: Specified location of the analysis
  By specifying L to 1, we decided to perform this only to the first chromosome.

Then we continue to the Base Recalibration.
Each base has its own base quality value. This step is performed per-sample and consists of applying machine learning to detect and correct for patterns of systematic errors in the base quality scores, which are confidence scores emitted by the sequencer for each base. Base quality scores play an important role in weighing the evidence for or against possible variant alleles during the variant discovery process, so it's important to correct any systematic bias observed in the data.

First we need to create a sequence dictionary:

9. $ ./gatk CreateSequenceDictionary -R genome/Arabidopsis_thaliana.TAIR10.dna.toplevel.fa -O genome/Arabidopsis_thaliana.TAIR10.dna.toplevel.dic t

10. $ cat arabidopsis_thaliana.vcf | sed ‘s/The 1001 Genomes Project_2016/The_1001_Genomes_Project_2016/g’ > arabidopsis_thaliana_corrected_spaces.vcf

11. awk ‘{if(NR!=1687355){print $0}}’ arabidopsis_thaliana_corrected_spaces.vcf > arabidopsis_thaliana_corrected_spaces_rmW.vcf

12. $ ./gatk IndexFeatureFile -I genome/Arabidopsis_thaliana_rmW.vcf

13. $ samtools index gatk/STAR_mapped_SRR7287381_Aligned.out.sorted.RG.rmdup.split.BQSR.bam

The next step is to do the variant calling, which we do using the HaplotypeCaller. HaplotypeCaller doesn't need any specific changes to run with RNA once the bam has been run through SplitNCigarReads. We do adjust the minimum phred-scaled confidence threshold depending on the specific use case. When the dataset shows regions with variation, the function discards the emapping information and reassembles the reads in that region. So the program takes bam file, and based on it checks on which site there are different nucleotides, so polymorphisms.

14. $ ./gatk HaplotypeCaller -R genome/Arabidopsis_thaliana_TAIR10.dna.toplevel.fa -I alignment/STAR_mapped_SRR7287381_Aligned.out.sorted.RG.rmup.split.BQSR.bam -O alignment/STAR_mapped_SRR7287381_Aligned.out.sorted.RG.rmdup.split.BQSR.g.vcf.gz -ERC GVCF -L 1

- -ERC: Mode for emitting reference confidence scores (For Mutect2, this is a BETA feature)

The following function that we will use is GenotypeGVCFs. It outputs a .vcf file with all the samples having been jointly genotyped. It is a very useful tool for joint genotyping on a single input. The input file can contain one or more samples.

15. $ ./gatk GenotypeGVCFs -R genome/Arabidopsis_thaliana.TAIR10.dna.toplevel.fa -V alignment/STAR_mapped_SRR7287381_Aligned.out.sorted.RG.rmdup.split.BQSR.g.vcf.gz -O output.vcf.gz -L 1

Then we installed IGV. The Integrative Genomics Viewer (IGV) is a high-performance, easy-to-use, interactive tool for the visual exploration of genomic data. We can start it by opening a file in the terminal and then writing "./igv.sh".

In order to view the alignment we need a reference genome, and the mapped alignment of chromosome one from the previous class, as well as a new file that we created today in order to compare the alignment.

Below there is an image representing this comparison. In it we can see that there were actually changes in the alignment.

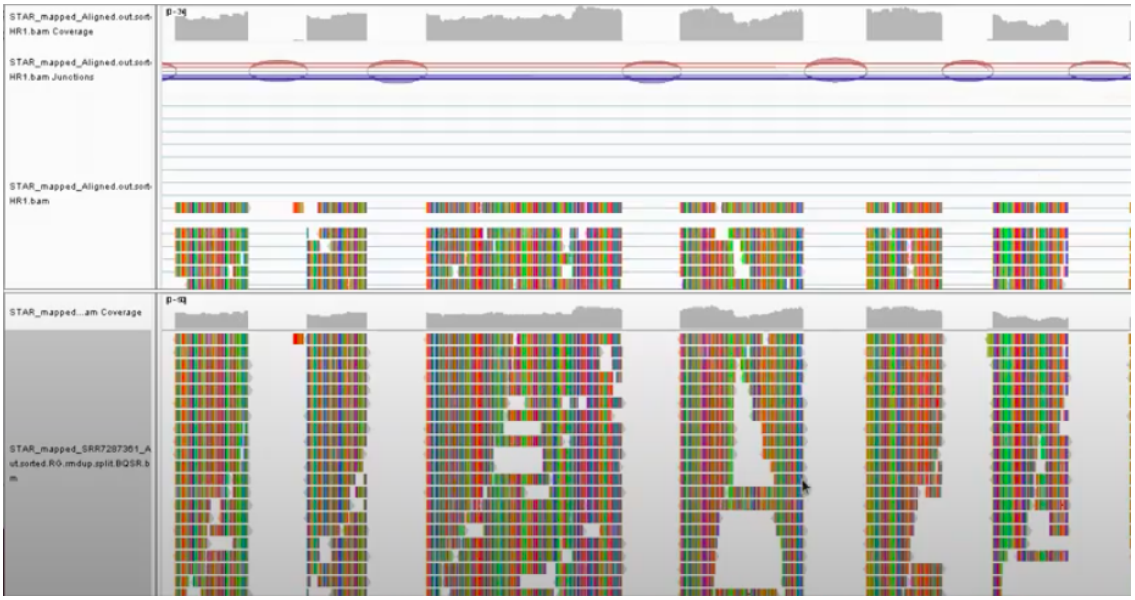


Figure 2. - Comparing two alignments

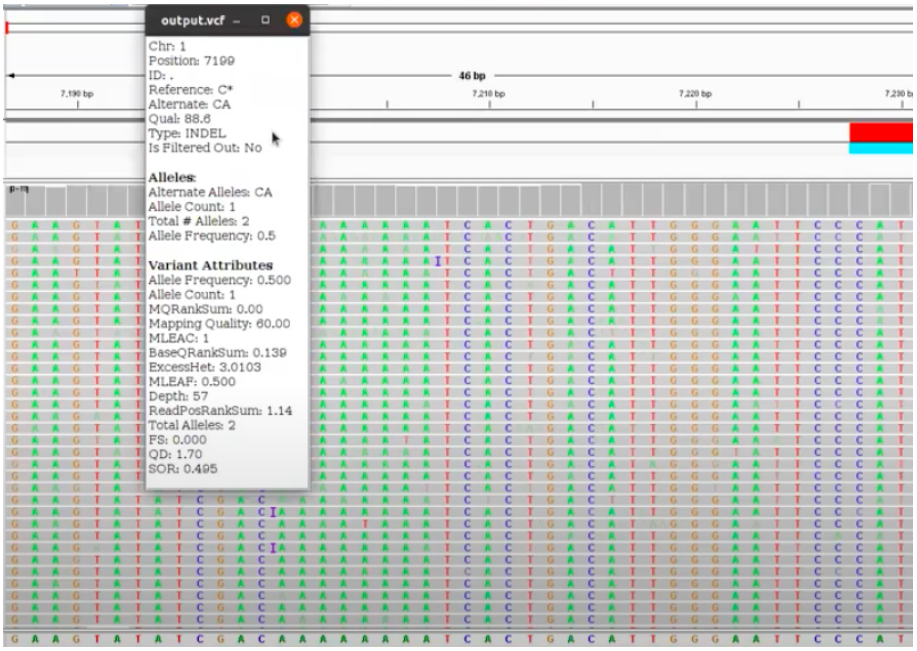


Figure 3. - Information about the mutation

In the third figure, we have the detailed information of the mutation present on the first chromosome, which is located at position 7199. There are also more information about alleles and variant attributes.

After this we move to the selection of the variants, which is a part of variant filtering.We do that using the function SelectVariants. It selects a subset of variants with given criteria.

16. $ ./gatk SelectVariants -V alignment/output.vcf.gz -select-type SNP -O alignment/output_SNP.vcf.gz

Then we filter the most trustable variants with the function VariantFiltration. The function filters the dataset with given criteria.

17. $ ./gatk VariantFiltration -V alignment/output_SNP.vcf.gz -filter “OD < 2.0” --filter-name “QD2” -filter “QUAL < 30.0” --filter-name “QUAL30” -filter “SOR > 3.0” --filter-name “SOR3” -filter “FS > 60.0” --filter-name “FS60” -filter “MQ < 40.0” --filter-name “MQ40” -filter “MQRankSum < -12.5” --filter-name “MQRankSum-12.5” -filter “ReadPosRankSum < -8.0” --filter-name “ReadPosRankSum-8” -O alignment/output_SNP_filter ed.vcf.gz

With SnpEff, we are checking if the database is available. It is a genetic variant annotation and functional effect prediction toolbox. It annotates and predicts the effects of genetic variants on genes and proteins (such as amino acid changes). Therefore it helps to filter and manipulate .vcf (genomic annotated files) and annotates the effects of variants on genes. The end file stores a big text file with our dataset gene sequence variations.

18. $ snpEff databases | grap -i arabidopsis

- grap: When we do not want to print all, but something specific

19. $ snpEff Arabidopsis_thaliana

20. $ snpEff Arabidopsis_thaliana output_SNP_filtered.vcf.gz > snps_filtered.ann.vcf

After this we have created a html file with the summary of snpEff report.

PART 6

EDirect

EDirect is a software package, developed by NCBI, which allows you to use the E-utilities API in a Unix environment. The aim of EDirect is to access Entrez, which is a search engine of the NCBI website.

First we needed to install EDirect, which we do using the following function:

1. $ sh -c "$(wget -q ftp://ftp.ncbi.nlm.nih.gov/entrez/entrezdirect/install-edirect.sh -O -)"

   It is important to mention that we will get the question regarding the editing PATH, which must be done in order to complete the configuration process, and we have to answer that question with yes.

The next thing we had to do is to install BLAST. First we downloaded the file of the software needed for linux and then we finish the installation with the following function, which extracts the downloaded package:

2. $ tar zxvpf ncbi-blast-2.11.0+-x64-linux.tar.gz

After this, we can check which databases are available and can be accessed with EDirect.

3. $ einfo -dbs

   Nucleotide database is also called nuccore, and it is available for analysis.

4. $ esearch -db nuccore -query "lycopene cyclase AND lanum lycopersicum[ORGN]"

- esearch: Search NCBI database
- ORGN: Short for organism

   This function provided us even the number of entries, which in our case is fifty.

5. $ esearch -db nuccore -query "(lycopene[All Fields] AND cyclase[All Fields]) AND Solanum lycopersicum[Organism] AND (100[SLEN] : 5000 [SLEN])"

   This function gave the result that the count number of entries now is 44.

Select a gene from an organism according to your preferences and write a command to get protein sequence from RefSeq database with EDirect tools in fasta format (using esearch tool).

6. $ esearch -db nuccore -query "(lycopene[All Fields] AND cyclase[All Fields]) AND Solanum lycopersicum[Organism] AND (100[SLEN] : 5000 [SLEN])" | efetch -format fasta > solanum_lycopersicum.fasta

- efetch: Downloads records from NCBI database
- -format: Specifies the format of the output file

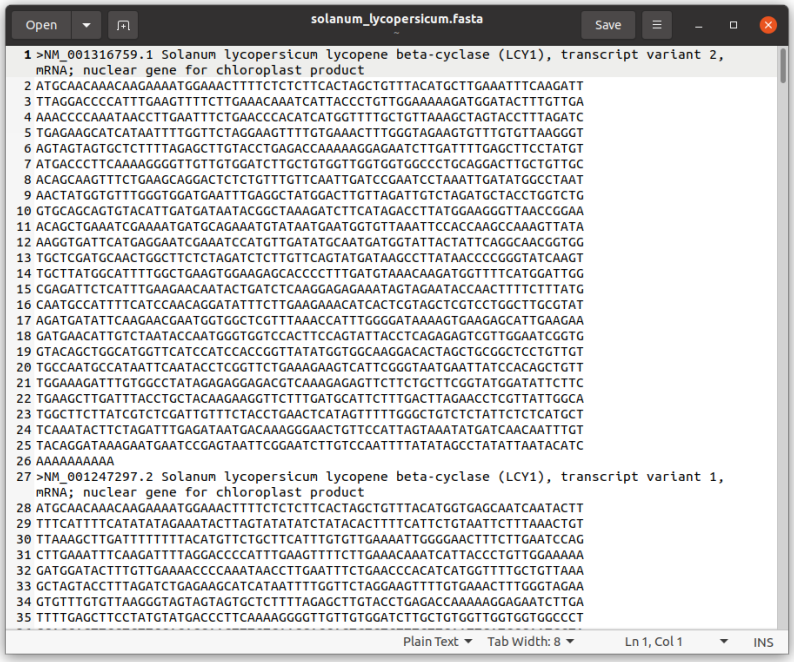


Figure 1. - Opened created fasta file

7. $ esearch -db nuccore -query "(lycopene[All Fields] AND cyclase[All Fields]) AND Solanum lycopersicum[Organism] AND (100[SLEN] : 5000 [SLEN])" | efetch -format gbc | head -n 100

   This function creates a xml file, and we used the “head -n” function to specify that for an output we want to view just the first 100 lines.

```
(base) ivana@ivana-folio:~$ esearch -db nuccore -query "(lycopene[All Fields] AND cyclase[All Fields]) AND Solanum ly
copersicum[Organism] AND (100[SLEN] : 5000 [SLEN])" | efetch -format gbc | head -n 100
<?xml version="1.0" encoding="UTF-8"  ?>
<!DOCTYPE INSDSet PUBLIC "-//NCBI//INSD INSDSeq/EN" "https://www.ncbi.nlm.nih.gov/dtd/INSD_INSDSeq.dtd">
<INSDSet>
  <INSDSeq>

    <INSDSeq_locus>NM_001316759</INSDSeq_locus>
    <INSDSeq_length>1690</INSDSeq_length>
    <INSDSeq_strandedness>single</INSDSeq_strandedness>
    <INSDSeq_moltype>mRNA</INSDSeq_moltype>
    <INSDSeq_topology>linear</INSDSeq_topology>
    <INSDSeq_division>PLN</INSDSeq_division>
    <INSDSeq_update-date>12-APR-2021</INSDSeq_update-date>
    <INSDSeq_create-date>24-OCT-2015</INSDSeq_create-date>
    <INSDSeq_definition>Solanum lycopersicum lycopene beta-cyclase (LCY1), transcript variant 2, mRNA; nuclear gene f
or chloroplast product</INSDSeq_definition>
    <INSDSeq_primary-accession>NM_001316759</INSDSeq_primary-accession>
    <INSDSeq_accession-version>NM_001316759.1</INSDSeq_accession-version>
    <INSDSeq_other-seqids>
```

Figure 2. - Xml file as an output

Create a new txt file and write the NCBI accession number of protein sequence in it and repeat the previous exercise with epost tool.

8. $ esearch -db nuccore -query "(lycopene[All Fields] AND cyclase[All Fields]) AND Solanum lycopersicum[Organism] AND (100[SLEN] : 5000 [SLEN])" | efetch -format gbc | xtract -pattern INSDSeq -element INSDSeq_primary-accession > acc.numbers.txt

- xtract: Extracts specific data from XML and creates tabular output
- -pattern: Defining rows
- -element: Inserts identified patterns as different columns

9. $ cat acc.numbers.txt | epost -db nuccore -format acc | efetch -format fasta > solanum_lycopersicum_epost.fasta

- epost: Posts a list to the History server (e. g. a list of gene IDs can be uploaded, instead of using esearch)

Use elink to get corresponding mRNA from RefSeq and download records in GenBank format.

10. $ esearch -db nuccore -query NM_001316759.1 | elink -target protein -name nuccore_protein

- elink: Looks up similar, related or otherwise connected records in the same database, or linked records in a different database
- -target: Specifying target database

  This function shows us that there are three query keys, with one entry.

11. $ esearch -db nuccore -query NM_001316759.1 | elink -target protein -name nuccore_protein | efetch -format gp

Check how many sequences are linked to this in the nuccore database. With xtract tool extract accession number, length and definition (hint: download the results in gbc format).

12. $ esearch -db nuccore -query NM_001316759.1 | elink -target protein -name nuccore_protein | efetch -format gbc | xtract -pattern INSDSeq -element INSDSeq_primary-accession INSDSeq_length INSDSeq_definition

```
(base) ivana@ivana-folio:~$ esearch -db nuccore -query NM_001316759.1 | elink -target protein -name nuccore_protein |
 efetch -format gbc | xtract -pattern INSDSeq -element INSDSeq_primary-accession INSDSeq_length INSDSeq_definition
NP_001303688    500     lycopene beta cyclase, chloroplastic [Solanum lycopersicum]
```

Figure 3. - The above twelfth function performed and the result

STANDALONE BLAST

We can check which databases are available and can be accessed with BLAST.

13. $ update_blastdb.pl --showall

Database can be downloaded or updated with the following function:

14. $ update_blastdb.pl nr --decompress

Use a fasta sequence of a protein, which you have obtained in the previous task and write a command to search this sequence against the RNA RefSeq database and limit the search space to some other species and check if the transcript for this protein is available.

We use the blastx to perform a blast of nucleotide query against the protein database.

15. $ blastx -db refseq_protein -query solanum_lycopersicum_epost.fasta -out results.txt -outfmt 0 -entrez_query "solanum lycopersicum[ORGN]" -remote

- -db: BLAST database name.
- -query: Query file name.
- -out: Output file name
- -outfmt: Alignment view options, 0 is for pairwise
- -entrez_query: Restrict search with the given Entrez query. Remote searches only.
- -remote: Execute search on NCBI servers

Download all mRNA sequences of previously used organisms and create a local blast database. Repeat the analysis with the created database.

We can also create our local nucleotide database via following:

16. $ makeblastdb -in arabidopsis_mRNA_RefSeq.fasta -input_type -dbtype “nucl” -parse_seqids

- -in: Input file that contains a list of scoremat files (delimited by space, tab, or newline)
- -input_type: Input file type
- -dbtype: Molecule type of input, values can be nucl or prot
- -parse_seqids: Parse bar delimited sequence identifiers in FASTA input.

17. $ blastn -db Arabidopsis_mRNA_RefSeq.fasta -query solanum_lycopersicum_epost.fasta -out results.txt -outfmt 0

Figure 4. - Result of performed BLASTN

As we can see from the results, there were no hits found.