**Tutorial preprocessing RNASeq data with R-ODAF:**

**Omics Data Analysis Framework for Regulatory application**

# The R-ODAF pipeline:

|  |  |
| --- | --- |
| The omics data analysis framework (ODAF) for regulatory application (R-ODAF) was designed to address issues from regulatory agencies that prevent routine application of omics technology in regulatory toxicology.  One of the reasons for this has been a trepidation about relying on the produced data. It has been argued that omics outputs might not be sufficiently reliable for regulatory application because the techniques, bioinformatics and interpretation can vary. For these reasons the robustness of the obtained results is questioned. This reticence to trust omics data is further magnified by the lack of internationally agreed upon guidelines and protocols for both the generation and processing of omics data. One way forward would be to reach a consensus. Therefore, the Long-Range Research Initiative (LRI) project that has developed R-ODAFs for transcriptomics data obtained with microarray, RNA-sequencing or TempO-seq sequencing technologies. In the current tutorial covers the R-ODAF for RNASeq. Scripts and tools for running this R-ODAF can be found here: <https://github.com/R-ODAF/Main/tree/main>  For more information on the R-ODAF(s) please read our publication: <https://www.sciencedirect.com/science/article/pii/S0273230022000307> | Afbeelding met tekst, schermopname, Lettertype, ontwerp  Automatisch gegenereerde beschrijving  Figure 1: Schematic overview of the R-ODAF. |

# Setup the R-ODAF pipeline:

The R-ODAF pipeline can be easily set up using the supplied Dockerfile. Using this approach will create a miniconda environment that contains the necessary software with versions that were used for creation of the R-ODAF pipeline (fastp v0.20.0, star v2.7.1a, rsem v1.3.1, multiqc v1.7, and several R packages).

* Go to <https://github.com/R-ODAF/Main/tree/main/R-ODAF_docker>
* Obtain the Dockerfile
* Follow the steps in R-ODAF\_ReadMe.txt for installation

|  |
| --- |
| For this training, the docker has already been installed in the training environment |

If you want to use the latest software versions, please retrieve the software yourself (but be aware that some codes might need adapting to the new version).

# Obtain training data & reference genome:

For this tutorial, we will be analyzing a dataset obtained from the H2020 HeCaToS project, more specifically of Acetaminophen exposed 3D hepatic microtissues (<https://www.ebi.ac.uk/biostudies/hecatos/studies/S-HECA143>) and the corresponding DMSO controls (<https://www.ebi.ac.uk/biostudies/hecatos/studies/S-HECA158?query=s-heca158>).

Because preprocessing full datasets with the R-ODAF does take some time, we have created shortened fastq files (top 400.000 lines) of these samples for training purposes.

|  |
| --- |
| For this tutorial, the training data has already been loaded in the training environment |

Next to the sample data, the R-ODAF pipeline also requires a reference genome. These can be obtained from multiple sources, but for this tutorial, we use the human reference from ensemble: <https://www.ensembl.org/Homo_sapiens/Info/Index>.

In the field “gene annotation” on the right side of the page, you can download the latest version of the Human files. After clicking Download GTF, download the “Homo\_sapiens.GRCh38.110**.gtf.gz**” file. Thereafter, return to the previous page, click Download FASTA, select “DNA” and download the “Homo.sapiens.GRCh38**.dna.primary\_assembly.fa.gz**”. Before use, the genomes should be gunzipped.

|  |
| --- |
| For this training, the reference genome has already been loaded into the training environment. Furthermore, for this training only Chromosome 1 will be used as a reference to reduce the amount of hard disk space needed. |

# Running the R-ODAF:

The complete R-ODAF script can be found here: <https://github.com/R-ODAF/Main/blob/main/scripts/>. In this tutorial, we will go though it step by step. The tutorial will contain textual explanations and lines of code to be run. For easy recognition, lines of code are contained in grey boxes, made **bold** and start with > (this sign should not be pasted into the terminal).

#### #### Settings which need to be adapted by user ##### (lines 7-56)

In this first section, you will have to specify information related to the data that you are preprocessing. Every time you use the R-ODAF pipeline, this part of the script needs to be adapted to the files that you want to process. But before we do this, we will first make sure the miniconda environment containing the software is loaded into the session.

|  |
| --- |
| **> source ${HOME}/miniconda3/etc/profile.d/conda.sh** |

Next, we set a project name. This name is used in creating folders where the output is stored. If you want your output in a different directory, the OUTPUT\_DIR parameter can be adapted, but this is not necessary for the pipeline to run.

|  |
| --- |
| **> project="R-ODAFtutorial\_Chr1"**  **> OUTPUT\_DIR="/home/streamline/Downloads/${project}/output/"** |

For the pipeline to run, it must be able to locate the files to be processed. Therefore, it is very important to specify the RAW\_SAMPLE\_DIR (and check that it does not contain typing errors). Important to know: the pipeline will process all the files that are in the specified folder.

|  |
| --- |
| **> RAW\_SAMPLE\_DIR="/home/streamline/Downloads/${project}/ShortTrainingData/"** |

Next, we specify some information related to your data. First is the extension of input files. The R-ODAF needs fastq files as input, though they can be either gzipped or unpacked. The extension can therefor vary (".fastq", ".fq", ".fastq.gz" or ".fq.gz")

|  |
| --- |
| **> SUFFIX\_INPUTFILES='.fastq'** |

The R-ODAF was made for RNASeq data, but can also be used for data obtained with the TempOSeq technology. Because parameters differ between these methods, the sequencing type needs to be specified ("RNASeq" or "TempOSeq")

|  |
| --- |
| **> SEQTYPE="RNASeq"** |

Furthermore, sequencing can be done in single end mode or in paired end mode. This decision was made before the data was generated. In single end mode, each read is sequenced from one side, resulting in a READ1 fastq file, while in paired end mode, the read is sequenced from both sides resulting in a READ1 and READ2 fastq files.

|  |
| --- |
| Afbeelding met tekst, schermopname, lijn, Parallel  Automatisch gegenereerde beschrijving |
| Figure 2: Schematic representation of paired-end sequencing, depicting the difference between read 1 and read 2 |

The SEQMODE parameter should therefore specify either “single” or “paired”, according to your data. (In some cases of paired end sequencing, the quality of READ2 may be lacking. In these cases, it can be decided to run the R-ODAF in single end mode.)

|  |
| --- |
| **> SEQMODE="single"** |

The extension or suffix of the reads can be different based on the conventions the data generating lab is using (e.g. "\_R1\_001"). Therefore, you should have a look at the extensions that your data has and specify them here. When SEQMODE = " paired", both forward and reverse need to be specified.

|  |
| --- |
| **> PAIRED\_END\_SUFFIX\_FORWARD="\_R1"**  **> PAIRED\_END\_SUFFIX\_REVERSE="\_R2"** |

When SEQMODE = " single", only the forward read needs to be specified (the reverse read will not be called in this mode and can therefore be anything left from a previous run. We advise not to remove the parameter)

Now that we have specified information relating to the files themselves, it is important to choose the organism for genome alignment.

The ORGANISM\_GENOME\_ID parameter is made for the user to identify which genome version (e.g " **Human\_** **GRCh38**") was used. It can contain any text as it is never called in the pipeline itself. It is however nice to record this in the script for FAIR purposes when sharing output and results.

|  |
| --- |
| **> ORGANISM\_GENOME\_ID="Human\_** **GRCh38\_r110"** |

For the pipeline, it is important to specify where the genome files (fasta and GTF) are located and how there files are called.

|  |
| --- |
| **> GENOME\_FILES\_DIR="/home/streamline/Downloads/${project}/RefHuman\_GRCh38\_r110\_Chr1/"**  **> GENOME\_FILE\_NAME="Homo\_sapiens.GRCh38.dna.chromosome.1.fa"**  **> GTF\_FILE\_NAME="Homo\_sapiens.GRCh38.110.gtf"** |

Furthermore, the script needs to know whether this is the first time that these files are used. If this is the case, the genome files need to be specifically prepared (indexed) for use in the STAR and RSEM software. Once this has been done, this process can be skipped (which saves a lot of time and working memory). When "Yes" is specified, the indexing will be skipped. If "No" the index will be made.

|  |
| --- |
| **> GENOME\_INDEX\_ALREADY\_AVAILABLE="Yes"**  **> RSEM\_INDEX\_ALREADY\_AVAILABLE="Yes"** |

The last parameter relating to genome files defines whether a large genome is used. This parameter is set to "Yes" when working with a human genome and "No" for other species. This parameter reduces the RAM usage during indexing of the human genome, making sure that the R-ODAF pipeline can be used on many systems.

|  |
| --- |
| **> LARGE\_GENOME="Yes"** |

Finally, we set some system parameters, more specifically the amount of CPU the program is allowed to use. It is important to know that a higher number of CPUs makes the pipeline run faster. Furthermore, the alignment step is the most time-consuming part of the script. If you are working on a stand-alone system or a private virtual machine, the CPUs can be set to the max of your environment. However, if you are working on a system that is also used by others (e.g. a department server) the parameters should be set so that there are CPUs left for other users, e.g.:

|  |
| --- |
| **> CPU\_FOR\_ALIGNMENT=20**  **> CPU\_FOR\_OTHER=6** |

When all these parameters are set, you are ready to run the pipeline. In the future, you will probably not even look at the code parts that are coming up next, because they do not need any additional user input. However, in this tutorial, we will go over these parts and explain what the pipeline is doing.

#### # Running the sequencing R-ODAF # (lines 59-91)

In the previous section, parameters for the R-ODAF pipeline were specified. In that section the parameter names were made so that the user can easily understand what information is required. However, in the script itself shorter parameter names are used. In this section the transformation from long to short name is done.

|  |
| --- |
| [**https://github.com/R-ODAF/Main/blob/main/scripts/R-ODAF\_sequencing\_DataPreprocessing.sh**](https://github.com/R-ODAF/Main/blob/main/scripts/R-ODAF_sequencing_DataPreprocessing.sh)  **> Run lines 59-91** |

#### ### Defining parameters for script to run (no user input necessary) ### (lines 94-139)

This section of the script makes the directories and subdirectories that are needed for the pipeline. It also starts a logging process that saves all text printed in the terminal to a text file. This text file can be important for identifying whether the process has finished without errors and for adhering to FAIR principles when sharing output and results. (We will skip the logging part for now)

|  |
| --- |
| [**https://github.com/R-ODAF/Main/blob/main/scripts/R-ODAF\_sequencing\_DataPreprocessing.sh**](https://github.com/R-ODAF/Main/blob/main/scripts/R-ODAF_sequencing_DataPreprocessing.sh)  **> Run lines 94-119**  **> Do NOT run lines 120-139 (during this tutorial)**  **> Run line 135** |

*### Trimming raw reads: Fastp ### (lines 142-212)*

The first step of preprocessing RNAseq data is called trimming. The main goal of trimming is the removal of artificial adapter sequences. These are sequences that have been added during the library prep (figure 3) in order to for the technique to work. Because these sequences do not map to regions in the genome, if not removed they can interfere with downstream analysis and lead to inaccurate results. After trimming, the output fastq files contain only the sequence of the input RNA (green in figure 3)

|  |  |
| --- | --- |
| A diagram of a dna sequence  Description automatically generated | Figure3: Library preparation workflow for RNA-seq.  The Illumina workflow ligates a 3´ adenylated adapter (purple) to the 3´ end of the RNA (green). Then, a 5’adapter (red) is ligated to the 5´ end of the RNA.  The doubled-ligated products are RT-PCR amplified to introduce barcodes (orange) for multiplex applications and generate sequencing libraries and sequences for attachment to the flowcell (blue).  Source [1] |

In the R-ODAF pipeline, trimming of the reads is done with Fastp. Fastp uses an algorithm to detect the adapter sequences in the data. In single end mode, this algorithm uses 2 assumptions to detect the adapter sequences: 1) the adapter sequence is identical in all reads of a single sample, and 2) adapters are always located at the tail of the read. In paired end mode, the sequence of the sample itself is detected through complementary base pairing. Consequently, the adapters are identified as all bases in the ends that do not overlap between the reads [2]

Next to adapter sequences, trimming also removes low-quality bases that are caused by sequencing errors or other factors such as poor sample quality. Finally, reads that have been cut too short (due to many low quality bases) are also removed because their quality is questionable. Furthermore, shorter reads are less specific and may have the ability to map to multiple places in the genome, reducing the confidence in obtained results from these reads.

Explanation of Fastp code.

**Code Trimm&QC:** fastp --in1 input\_forward.fastq.gz --in2 input\_reverse.fastq.gz --out1 output\_forward.fastq.gz --out2 output\_reverse.fastq.gz --json output\_filename.json --html output\_filename.html --cut\_front --cut\_front\_window\_size 1 --cut\_front\_mean\_quality 3 --cut\_tail --cut\_tail\_window\_size 1 --cut\_tail\_mean\_quality 3 --cut\_right --cut\_right\_window\_size 4 --cut\_right\_mean\_quality 15

* --in1; inputfile for single end mode or the forward\_input\_file for paired end mode.
* --in2; reverse\_input\_file for paired end mode.
* --out1; output\_file for single end mode or the forward\_output\_file for paired end mode.
* --out2; reverse\_output\_file for paired end mode.
* --cut\_front --cut\_tail\_window\_size 1 --cut\_tail\_mean\_quality 3; remove low quality bases (quality below 3; specified as N) at the read start. When the window\_size is set to 1, functionalities are equal to the LEADING option of trimmomatic.
* --cut\_tail --cut\_tail\_window\_size 1 --cut\_tail\_mean\_quality 3; remove low quality bases (quality below 3; specified as N) at the read end. When the window\_size is set to 1, functionalities are equal to the TRAILING option of trimmomatic.
* --cut\_right --cut\_right\_window\_size 4 --cut\_right\_mean\_quality 15; scan the read with a 4-base wide sliding window, cutting when the average quality per base drops below 15. Functionalities are equal to the SLIDINGWINDOW option of trimmomatic.
* --length\_required 36; remove reads when they are shorter as 36 bases after trimming.

|  |
| --- |
| [**https://github.com/R-ODAF/Main/blob/main/scripts/R-ODAF\_sequencing\_DataPreprocessing.sh**](https://github.com/R-ODAF/Main/blob/main/scripts/R-ODAF_sequencing_DataPreprocessing.sh)  **> Run lines 142-212** |

In the R-ODAF workflow (figure 1) a quality control step is included after the reads are trimmed. However, this quality step is not in the script. Well, it actually is, but it’s not located in between the trimming and alignment step. To promote automatization of the pipeline, all preprocessing steps are executed one after another, each saving their information in specified subfolders. At the end of the pipeline, the quality control is run for all intermediate steps at once. We will get to this topic at the end of the tutorial.

*### Alignment of reads (paired end & single end) ### (lines 215-309)*

After trimming, we end up with clean and reliable sequencing reads. Buit we still don’t know which read belongs to what gene. To retrieve this information, the sequencing reads are aligned to a reference genome.

The first time a reference genome is used, the FASTA sequence needs to be indexed. Each genome has a corresponding GTF file that is used for this purpose. The GTF contains information were each gene is located (chromosome number, gene start (bp), gene end (bp)) and the identity (gene ID, transcript ID, gene name, … ). Indexing is the process in which the FASTA sequences are linked to the identity contained in the GTF.

For this tutorial, the genome is already indexed. Therefore, this step is skipped (because we set ${GenomeIndexDone} == "Yes"). If you want to index a genome in the future, setting ${GenomeIndexDone} == "No" will execute the indexing in lines **220-277** of <https://github.com/R-ODAF/Main/blob/main/scripts/R-ODAF_sequencing_DataPreprocessing.sh>

Now that the sequence of the reference genome is linked to gene identities, we can take our own data and map this to the reference. Mapping means finding out where on the genome our sample reads originate from. The sequence of the sample read is searched in the reference genome. When the sequences are identical (or highly similar), the gene ID of this location is assigned to the sample read. A visual representation of this is included in Figure 4.

|  |
| --- |
| A blue line in a black background  Description automatically generated |
| Figure 4: schematic overview of read alignment. (Red) sequenced reads of the sample, (Black/Blue) reference genome, (Blue) genes & (Black) non-coding DNA.  Sequence similarity between the sample reads and the reference genome determines which gene the read belongs to. Source: <https://training.galaxyproject.org/archive/2022-06-01/topics/proteomics/tutorials/proteogenomics-dbcreation/tutorial.html> |

In the R-ODAF pipeline, read alignment is done with STAR. STAR is an aligner designed to specifically address many of the challenges of RNA-seq data mapping using a strategy to account for spliced alignments. As you probably know, splicing is the process of removing introns (non-coding regions) from a pre-messenger RNA (pre-mRNA) and connecting the protein-coding exons back together to form a mature mRNA molecule. Through this process a single gene can give rise to multiple mRNA variants (called transcripts), and consequently, multiple protein isoforms. Some transcripts may contain all exons of the gene, some may contain only subset and even the order of the exons may differ between transcripts. Furthermore, some transcripts may still contain one or more introns.

As can be seen in figure 5, a distinguishment can be made between reads mapping completely to exomes, reads mapping completely to introns and reads that partly map to two parts of the genome. The latter ones are known as junction reads (sepia). These reflect which parts (e.g. exons) were spliced together. To oversimplify the process, STAR looks at the ratio of junction reads, reads mapping to an intron, and reads mapping to an exon to determine which transcripts were present in the sample.

|  |
| --- |
| A diagram of a dna sequence  Description automatically generated |
| Figure 5: Mapping spliced reads to a reference genome. RNA-seq experiments produce short reads sequenced from processed mRNAs. When a reference genome is available the reads can be mapped on it using efficient alignment software. Due to splicing, some reads (known as junction reads) will partially map to the end of one exon and partially to the beginning of another (displayed in sepia). Furthermore, some reads will map to introns, indicating a spliced variant with a retained intron. This information is used by aligners to determine the identity of the RNA’s in the sample. Figure adapted from: <https://www.hindawi.com/journals/bmri/2010/853916/> |

Explanation of STAR code.

**Code:** <PathToProgram>/STAR --genomeDir <PathToGenomeDirectory> --readFilesIn <input1> <input2> --outFileNamePrefix <PathToOutputdir + output\_prefix> –-runThreadN 20

* <PathToProgram>; specifies the directory in which STAR is installed. This directory (usually ending with /source/) contains the STAR.cpp file.
* --genomeDir <PathToGenomeDirectory>; specifies the directory in which the genome indexes from step1 were saved.
* --readFilesIn <input1> <input2>; specifies the FASTA file(s) of the samples, with <input1> being the forward reads and <input2> the reverse. These files are not allowed to be compressed (eg. .gz/.bz2/.gzip ; if they are compressed, decompression before running is necessary).
* --outFileNamePrefix <PathToOutputdir><output\_prefix>; specifies where to store the output (<PathToOutputdir>) and what name the output should have (<output\_prefix>).
* --runThreadN 20; specifies the number of threads to be used (for many systems this equals the number of CPUs to be used). A higher number decreases the runtime. The actual setting depends on the operating hardware and the number of users on the system. Setting this value to 20 results in decent runtimes, but it can be adapted based on the used system capacity.

|  |
| --- |
| [**https://github.com/R-ODAF/Main/blob/main/scripts/R-ODAF\_sequencing\_DataPreprocessing.sh**](https://github.com/R-ODAF/Main/blob/main/scripts/R-ODAF_sequencing_DataPreprocessing.sh)  **> Run lines 248-309** |

*# QUANTIFICATION RSEM # (lines 312-380)*

After the reads have been mapped to the genome, the expression of each gene (or transcript) needs to be quantified. Basically, the number of reads assigned to each gene (or transcript) are counted. All this information is recorded in a big table which is often referred to the counts table.

In the R-ODAF pipeline, the quantification is done with RSEM. The major benefit of RSEM is that it returns 2 counts tables, one at gene level and one at transcript level. The researcher can decide which file suits best with their research question, without the hazard of having to switch to another preprocessing pipeline.

During the RSEM quantification, information from the reference genome is used. As was the case for STAR, the reference genome needs to be prepared, though for RSEM the required format different. For this tutorial, the genome is already prepared and this step is skipped (because we set ${ GenomeIndexRSEMDone} == "Yes"). If you want to prepare a genome in the future, setting ${ GenomeIndexRSEMDone} == "No" will execute the indexing in lines 315-321 of <https://github.com/R-ODAF/Main/blob/main/scripts/R-ODAF_sequencing_DataPreprocessing.sh>

Explanation of RSEM code.

**Code run STAR:** <PathToProgram>/STAR --genomeDir <PathToGenomeDirectory> --readFilesIn <input1> <input2> --outFileNamePrefix <PathToOutputdir + output\_prefix> –-runThreadN 20

**Necessary for compatibility with Cufflinks:** --outSAMstr andField intronMotif --outFilterIntronMotifs RemoveNoncanonical

**Necessary for compatibility with RSEM:** --quantMode TranscriptomeSAM

* --outSAMstr andField intronMotif; cufflinks requires alignments with XS strand attribute, which is added by this code (for unstranded RNA-Seq data).
* --outFilterIntronMotifs RemoveNoncanonical; for cufflinks it is recommended to remove non-canonical junctions, which is accomplished by including this code.
* --quantMode TranscriptomeSAM; outputs alignments to transcriptome (necessary for RSEM) into a separate file.

|  |
| --- |
| [**https://github.com/R-ODAF/Main/blob/main/scripts/R-ODAF\_sequencing\_DataPreprocessing.sh**](https://github.com/R-ODAF/Main/blob/main/scripts/R-ODAF_sequencing_DataPreprocessing.sh)  **> Run lines 312-380** |

The RSEM quantification is the last preprocessing step of the R-ODAF pipeline. The output of this tool, genes.data.tsv (=gene-level expression values) or isoforms.data.tsv (=transcript-level expression values) can be used for data analysis (e.g. differential gene expression, correlation analysis, …). However, as mentioned before, all quality control steps were left until the end of the script.

*### Quality control raw reads: Fastp + RSEM + STAR MultiQC report ### (lines 383-388)*

To obtain quality control reports, the R-ODAF uses the MultiQC tool. This tool is capable of handling QC data from all tools that are used in the R-ODAF (and many more) and is run by a single line of code.

|  |
| --- |
| [**https://github.com/R-ODAF/Main/blob/main/scripts/R-ODAF\_sequencing\_DataPreprocessing.sh**](https://github.com/R-ODAF/Main/blob/main/scripts/R-ODAF_sequencing_DataPreprocessing.sh)  **> Run lines 383-388** |

After running these lines, the QC report can be found in the folder “…/R-ODAFtutorial/output/Trimmed\_reads/MultiQC/. Open the html file to see the QC result.

The General statistics gives a summary of the quality of your data. Here you can see different versions of the same sample. Rows of samples that still display the readsuffix (e.g. \_R1, \_R2) contain information from the trimming of these files. Rows of samples without the readsuffix contain information from the alignment (= mapping) and quantification.

1. **General statistics table – Trimming:**

Before we asses the information, we customize the general table to depict the information we want. This is done by clicking the “Configure Columns” button (underneath the title “General Statistics”. We will start with the QC of the trimming process: Therefore, we select: &>Q30, Mb Q30 bases, GC content and %PF.

We now assess whether the samples pass the thresholds as specified by the R-ODAF.

**Sequencing quality:** For this purpose, the %>Q30 value of the MultiQC general statistic table is used. When the %>Q30 value is > 70%, obtained sequences are of sufficient quality and can be mapped with confidence to the genome.

**Sequencing depth:** The sequencing depth itself determines the ability to detect gene expression changes during downstream analysis. For analysis of highly expressed genes, 5-25 million reads per sample is advised. For a more global analysis, also including lower expressed RNAs, 30-60 million reads per sample are needed. The sequencing depth can be obtained from the Mb Q30 bases by calculating: (value/read length)\*%>Q30. However, it is easier to use the Mb Q30 bases column to assess if the dataset contains outliers and check the actual sequencing depth in a plot later in the document.

**GC content:** The genome and transcriptome contain an (more or less) equal distribution of bases. Therefore, the GC content should be around 50%. Values that are much lower or higher can point to technical issues related to amplification (PCR steps in the library prep) and may introduce a bias in the data.

**%PF:** The percentage of reads passing the filter indicates the quality of the sample. Here it is of importance to assess if the dataset contains outliers that should be assessed in more detail in upcoming plots.

**Tutorial data – Conclusions from the General Statistics table:**

All samples have good sequencing quality (%Q30 >90% & GC content ≈ 50%). There are 2 samples that depict lower amount of Mb Q30 bases and % PF: APAP\_The\_024\_2\_R1 & APAP\_Tox\_002\_1\_R1. These need to be assessed in more detail.

1. **Fastp QC plots:**

Next, we go to the detailed information from the fastp trimming (use the navigation on the left or scroll down).

**Filtered reads:** This barplot shows the amount of reads that pass the filter in blue and the once that failed in other colors, depending on the reason for failing.

The sequencing depth can be assessed by looking at the full length of the bar. The sequencing depth (as described above) should be at least 5 million. For this tutorial run, all samples fail this because we work with shortened fastq files (100k), so we ignore this threshold now. What we do see from this plot is that sample APAP\_The\_024\_2\_R1 has significantly lower sequencing depth, only 25k reads, and should therefore be excluded from the analysis. Furthermore, sample APAP\_Tox\_002\_1\_R1 shows that more reads were removed because they were too short after trimming. This could indicate that the sample RNA was of lower quality. We do not discard this sample at this point but keep it under investigation.

**Sequencing quality**: This plot shows the average sequencing quality of the reads per sample, from the 3’end to the 5’end. From the general statistics we already know that the overall sequencing quality is good. In this plot, we focus on the end of the read. It is common for reads to have a lower quality at the end of a read (for paired end, also lower quality in read 2 compared to read 1 is normal). The quality should not drop below Q20 and the drop should be comparable between samples. In this plot we again see APAP\_The\_024\_2\_R1 & APAP\_Tox\_002\_1\_R1 with lower quality then the rest. However, when we move to the tab “After filtering”, only APAP\_The\_024\_2\_R1 stands out.

**GC content:** Here we select the tab “After filtering” (and we ignore the fluctuating line from APAP\_The\_024\_2\_R1). Here we see nice GC content around 50% for most of the read. Only in the beginning of the read we see that all samples depict the same drop and rise in percentage. This identifies that the sequenced read still contains some artificial bases at the start of the read. This is a common phenomenon for many library prep procedures.

The current data was obtained using a Lexogen kit, for which the first 13 bases are artificial. To remove these bases, the R-ODAF should be run again, though this time the fastp code should include:

|  |
| --- |
| **> --trim\_front1 13 \ (for single & paired end runs)**  **> --trim\_front2 13 \ (only for paired end runs)** |

1. **General statistics table – Alignment:**

For assessment of the alignment, we return to the general statistics table. We now select: STAR %Aligned, STAR M Aligned.

**STAR %Aligned:** This tells you how many of the trimmed & filtered reads are able to align to the reference genome. The recommendations from the R-ODAF pipeline, describe that at least 70% of the reads should map. A lower percentage could indicate sequencing errors or a high amount of mutations (for some studies, this threshold may be adapted to fit the research question).

**STAR M Aligned:** This parameter informs about the amount of reads that were aligned. The R-ODAF recommendations do not specify a threshold for this parameter because other parameters should identify “bad” samples. However, because analysis of sequencing data often makes use of CPM (counts per million) in thresholds or data reporting, it is important that at least 1 million reads have been aligned (otherwise, conversion to CPM will give a skewed image of the reality).

**Tutorial data – Conclusions from the General Statistics table:**

The information from the general statistics table again identifies APAP\_The\_024\_2 as a sample that should be discarded. Furthermore, sample APAP\_Tox\_024\_3 reveals a %Aligned just below the advised threshold. Finally, the sample APAP\_Tox\_002\_1\_R1 (which we were monitoring more closely due to maybe lower quality) the % was good, but the M Aligned was lower.

1. **Alignment plots:**

For more detailed assessment, we turn to the barplot from STAR (the barplot from RSEM contains similar though less information and is therefore not assessed). By now it is obvious that APAP\_The\_024\_2 is unsuited for analysis. APAP\_Tox\_002\_1\_R1 has low amount of mapped reads, though in the full sample it might be decent for analysis. Finally, APAP\_Tox\_024\_3 does not show obvious issues in this plot, however, when we switch to Percentages we see that this sample has more reads mapping to multiple locations of the genome. This has the potential to reduce confidence in the obtained results.

1. **Overall conclusion**

In reality, the data should be rerun with the additional trimming parameter and QC needs to be assessed again. For this tutorial, we have not included this because we think that the process and the way to handle the data is clear. From the currently assessed QC results, the following conclusions would be drawn:

* APAP\_The\_024\_2 🡪 exclude from downstream analysis due to quality
* APAP\_Tox\_002\_1\_R1 🡪 Keep for now. Might have quality issues. Should be assessed by PCA in downstream analysis. Remove when it’s an outlier (>20% variance)
* APAP\_Tox\_024\_3🡪 Keep for now. Might have multimapper issues. Should be assessed by PCA in downstream analysis. Remove when it’s an outlier (>20% variance)

*### End of script ### (lines 390-395)*

The end of the script contains some final code to log and deactivate the conda environment

|  |
| --- |
| [**https://github.com/R-ODAF/Main/blob/main/scripts/R-ODAF\_sequencing\_DataPreprocessing.sh**](https://github.com/R-ODAF/Main/blob/main/scripts/R-ODAF_sequencing_DataPreprocessing.sh)  **> Run lines 390-395** |

References

1. Head, S.R., et al., *Library construction for next-generation sequencing: overviews and challenges.* Biotechniques, 2014. **56**(2): p. 61-77.

2. Chen, S., et al., *fastp: an ultra-fast all-in-one FASTQ preprocessor.* Bioinformatics, 2018. **34**(17): p. i884-i890.