Pipeline - Preprocessing of RNA-seq data

Jean Silva de Souza[1] and Mauro A. A. Castro[1]

[1]Bioinformatics and Systems Biology Lab, Federal University of Paraná, Curitiba, 81520-260, Brazil.

E-mails: jean.souza@edu.unipar.br

Contents

Abstract

The increasing use of RNA-seq technology has led to the development of new tools for each stage of pre-processing of this type of data, requiring a sequence of steps suitable for a better result. In this pipeline we present an RNA-seq preprocessing workflow. We use several tools in order to check the quality control (QC) of the FASTQ files, cleaning and filtering the reads and obtaining the gene expression matrix.

1. Introduction

The RNA-seq sequencing technique reveals the presence and amount of RNA in a biological sample, proposing several applications, ranging from the identification of differentially expressed genes and transcripts, alternative splicing and polyadenylation analyzes, to the detection of gene fusion and post-transcriptional events (Wang *et al.*, 2009). Consequently, RNA-seq has gained prominence in the analysis of expression of genes and transcripts, and has become the main object for this type of analysis.

Although the standardization of some stages of the pre-processing of RNA-seq are in progress, the number of tool options for use in each stage is huge. Furthermore, it is a challenging task to test each tool in order to choose the one that best fits each stage, especially for non-specialists in the field (Cornwell *et al.*, 2018). We propose in this pipeline a sequence of steps for analyzing RNA-seq data. The steps covered involve quality control, cleaning and filtering the reads, quantifying the transcripts and obtaining the gene expression matrix.

The pipeline steps were generated based on a prostate cancer cohort, which is currently not available in a database. However, we used public data from the Gene Expression Omnibus (GEO) repository to validate the pipeline. Regarding the tools, we present several ways on how to treat the data and which tools can be used at each stage.

1.1. Experimental data

The samples used in this pipeline are from a study of gene expression profile, based on the identification of different subtypes in diffuse gastric cancer available at GEO. Data were generated from transcriptome based on RNA-seq from 140 fresh frozen tissues, divided into gastric cancer tissues of the diffuse type (n = 107), intestinal type (n = 23) and normal gastric tissues (n = 10). We downloaded a sample of each tissue analyzed: intestinal gastric cancer tissue (SRR7014291), diffuse gastric cancer tissue (SRR7014430) and normal gastric tissue (SRR7014308).

1.2. Tools used

To check the quality control we use the FastQC and MultiQC tools. FastQC checks quality control on raw sequencing data, generating a set of statistics on the quality of FASTQ files (Andrews *et al.*, 2012). MultiQC (Ewels *et al.*, 2016) was used to group the data generated by FastQC, presenting the result of all samples in a single file.

The cleaning and filtering of the reads were carried out with the fastp and Cutadapt tools. Fastp is a tool that checks the quality control and filters the reads, and according to the parameters specified to it, we can check the quality control before and after cleaning, filter strings with specific sizes, as well as bases with low indexes of qualities and identify and cut strings related to adapters (Chen *et al.*, 2018). Cutadapt finds and removes strings from adapters and other types of unwanted strings (Martin, 2011). The advantage of Cutadapt is that it tolerates an error rate and accepts IUPAC wildcards, which can be useful for removing illumina adapters with different indexes (Martin, 2011).

Quantification of transcript expression was done with Salmon, a tool that quantifies transcript expression using RNA-seq data (Patro *et al.*, 2017). Its approach involves the use of new algorithms providing more accurate expression estimates, it is faster and uses little machine memory when compared to other tools and finally, it considers experimental attributes and biases observed in RNA-seq data (Patro *et al.*, 2017).

For the construction of the gene expression matrix we used the R/Bioconductor tximeta package, but we also provided a tutorial on how to build the matrix with the R/Bioconductor tximport package. Tximport imports abundance at the level of transcription, estimated counts and transcription lengths and summarizes this data in matrices (Soneson *et al.*, 2015). The package, on the other hand, imports the transcription quantification obtained by Salmon and automatically appends transcription intervals and metadata information (Love *et al.*, 2020).

2. Workplace

Download the pipeline in the personal folder of your Linux computer (~/) so that the generated scripts can access the necessary folder structure in each step. If you prefer, replace the "~/" with the path you downloaded the pipeline.

3. Samples

We downloaded the GEO samples with the following commands on the Linux terminal:

```
# -- LINUX TERMINAL --
# Download tool installation
sudo apt install curl

cd ~/Pipeline/Preproc_RNAseq/Tools
curl https://ftp-trace.ncbi.nlm.nih.gov/sra/sdk/2.9.6/sratoolkit.2.9.6-ubuntu64.tar.gz \
-o sratoolkit.2.9.6-ubuntu64.tar.gz

tar -xvzf sratoolkit.2.9.6-ubuntu64.tar.gz

# Downloading samples
cd ~/Pipeline/Preproc_RNAseq/Samples
~/Pipeline/Preproc_RNAseq/Tools/sratoolkit.2.9.6-ubuntu64/bin/fastq-dump \
--split-files --gzip SRR7014291
~/Pipeline/Preproc_RNAseq/Tools/sratoolkit.2.9.6-ubuntu64/bin/fastq-dump --split-files \
--gzip SRR7014430
~/Pipeline/Preproc_RNAseq/Tools/sratoolkit.2.9.6-ubuntu64/bin/fastq-dump --split-files \
--gzip SRR7014308
```

4. Installing the tools

The following describes how we install each tool, some of which can be installed in another way - check the documentation for the tools and choose your preferred installation mode.

4.1 System Libraries

```
sudo apt-get install libssl-dev openssl libcurl4 libcurl4-openssl-dev \
default-jdk gdebi libgl1-mesa-glx libegl1-mesa libxrandr2 \
libxrandr2 libxss1 libxcursor1 libxcomposite1 libasound2 libxi6 \
libxtst6 xml2 libxml2-dev libtiff-dev libfftw3-dev
```

4.2. FastQC

FastQC can be installed from the precompiled binary.

```
# -- LINUX TERMINAL --
cd ~/Pipeline/Preproc_RNAseq/Tools
# FastQC v0.11.9
wget https://www.bioinformatics.babraham.ac.uk/projects/fastqc/fastqc_v0.11.9.zip \
--no-check-certificate
unzip fastqc_v0.11.9
cd FastQC/
chmod 755 fastqc
./fastqc --help
```

4.3. MultiQC

The installation of the **MultiQC** package can be done with the python package manager "pip".

```
# -- LINUX TERMINAL --
sudo apt install python3-pip
sudo pip3 install multiqc
```

4.4. fastp

Download the compiled binary for CentOS/Ubuntu

```
# -- LINUX TERMINAL --
cd ~/Pipeline/Preproc_RNAseq/Tools
wget http://opengene.org/fastp/fastp
chmod a+x ./fastp
```

4.5. Cutadapt

Install **Cutadapt** using the Apt tool.

```
# -- LINUX TERMINAL --
sudo apt install cutadapt
```

4.6. Salmon

```
# -- LINUX TERMINAL --
# miniconda library
cd ~/Pipeline/Preproc_RNAseq/Tools
wget https://repo.anaconda.com/miniconda/Miniconda3-py37_4.8.3-Linux-x86_64.sh
bash Miniconda3-py37_4.8.3-Linux-x86_64.sh

# Restart the terminal
# Salmon
conda config --add channels conda-forge
conda config --add channels bioconda
conda create -n salmon salmon
conda config --set auto_activate_base False
```

4.7. R Libraries

The R libraries that we will use: **tximport** (Soneson *et al.*, 2015), **tximeta** (Love *et al.*, 2020), **GenomicFeatures** (Lawrence *et al.*, 2013), **ensembldb** (Rainer *et al.*, 2019), **AnnotationDbi** (Pagès *et*

al., 2019), **magrittr** (Bache and Wickham, 2014), **tibble** (Müller and Wickham, 2020), **knitr** (Xie, 2020), **kableExtra** (Zhu, 2019), **readr** (Wickham *et al.*, 2018) e **AnnotationHub** (Morgan and Shepherd, 2020).

Within RStudio or in the R environment, run the commands below to install the libraries that we will be using.

```
# -- ENVIRONMENT R --
# CRAN repository libraries
install.packages(c("magrittr", "tibble", "knitr", "kableExtra", "readr", "BiocManager"))

# Bioconductor repository libraries
BiocManager::install(c("rnaseqGene", "tximport", "tximeta","GenomicFeatures", "ensembldb",
                       "AnnotationDbi","AnnotationHub"))
```

5. Quality control before cleaning and filtering RNA-seq data

We check the quality of FASTQ files using the **FastQC** program(Andrews *et al.*, 2012). Through it we obtain quality statistics by file (and not by sample, as FastQC does not consider pairend).

To generate the files regarding the quality of the sequencing, we generate a script (script_quality_gross_seq_fastqc.sh) in Bash.

```
# -- LINUX TERMINAL --
# - script_quality_gross_seq_fastqc.sh -
## Generating the script
cd ~/Pipeline/Preproc_RNAseq/1_Qual_control_2-1/FastQC
cat > script_quality_gross_seq_fastqc.sh
#!/bin/bash
cd ~/Pipeline/Preproc_RNAseq/Samples
~/Pipeline/Preproc_RNAseq/Tools/FastQC/./fastqc *.gz
mv *zip* ~/Pipeline/Preproc_RNAseq/1_Qual_control_2-1/FastQC/Rep_zip_FastQC
mv *html* ~/Pipeline/Preproc_RNAseq/1_Qual_control_2-1/FastQC/Rep_html_FastQC

## -- Hit "Ctrl" plus "z" to save the script
```

OBS: If you do not want to generate the script, download the scripts and place each script in its respective working directory (the scripts are in the Scripts directory).

We run the script: script_quality_gross_seq_fastqc.sh via Linux terminal.

```
# -- LINUX TERMINAL --
# Running the script via terminal
bash ~/Pipeline/Preproc_RNAseq/1_Qual_control_2-1/FastQC/script_quality_gross_seq_fastqc.sh
```

In the ~/Preproc_RNAseq/1_Contr_qual_2-1/FastQC/Rel_zip_FastQC directory are the ".zip" files for the analyzed files, they contain the graphs of each step of the analysis. In the directory ~/Preproc_RNAseq/1_Contr_qual_2-1/FastQC/Rel_html_FastQC are the ".html" files for the same analyzed files, they present the graphs of each step of the analysis on a single HTML page per file, which can be viewed by the browser.

In the ".html" file generated by **FastQC** with RNA-seq data, the interesting thing is to look at the average of the quality indexes per position, distribution of the quality index in all sequences, content of N and the content of adapters. Figure 1 shows the most important graphics of a FASTQ file generated by the RNA-seq technique of the SRR7014308_2 file (directory of the HTML files: ~/Preproc_RNAseq/1_Contr_qual_2-1/FastQC/Rel_html_FastQC).

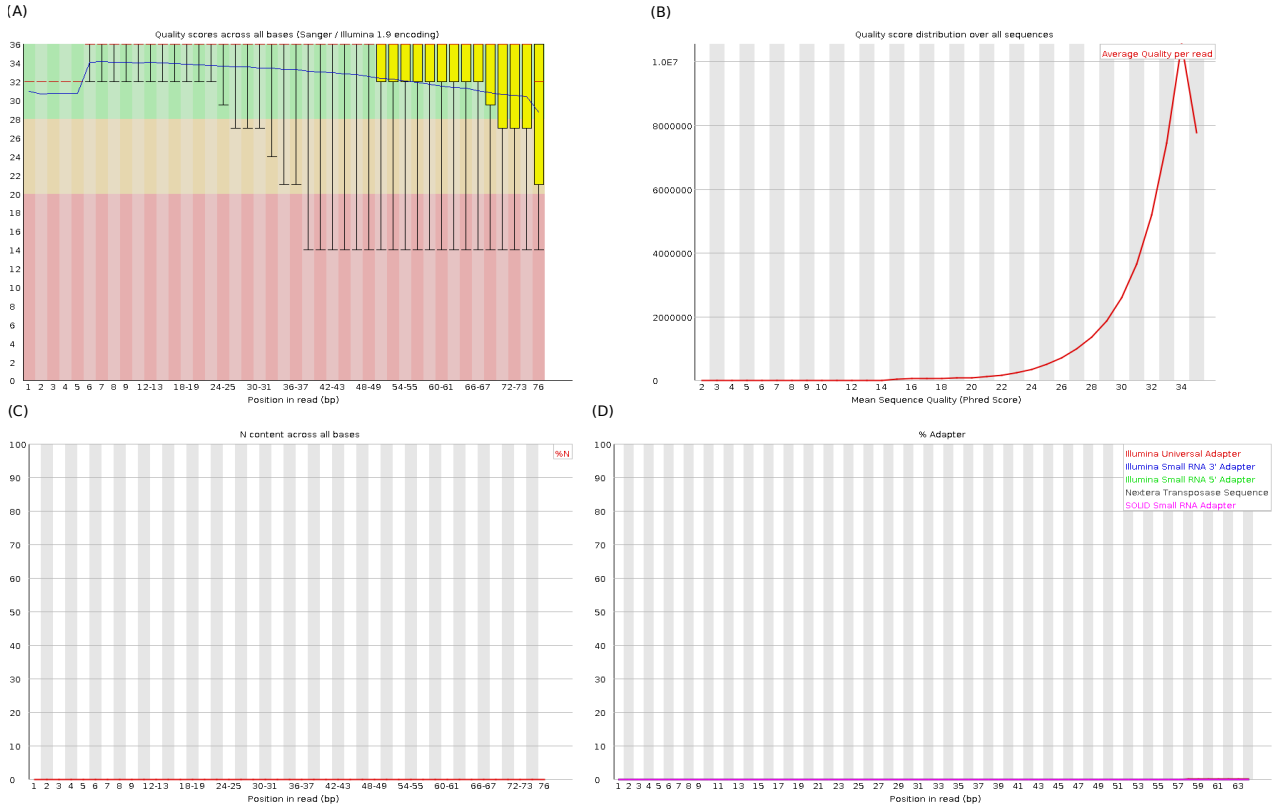

Figure 1: Quality control before cleaning and filtering reads from file SRR7014308_2. (A) Average quality indexes by position before cleaning and filtering RNA-seq data. (B) Distribution of quality indexes across all reads. (C) N content in all reads. (D) Adapter content.

The average quality indexes of the SRR7014308_2 file (Figure 1.A) were good, however some bases at the end of the sequences had low quality indexes making the average of these positions drop. Most of the reads had quality indexes> 20 on the Phred scale (Figure 1.B), indicating the success of the sequencing. Few Ns were identified (Figure 1.C), reinforcing the presentation of good sequencing. As for adapters, a low content of Illumina adapters was identified at the end of some reads (Figure 1.D). Such results show that the SRR7014308_2 file needs cleaning and filtering the reads.

To combine **FastQC** statistics in a single ".html" report we use the **MultiQC** tool (Ewels *et al.*, 2016). We generated a script (script_quality_gross_seq_multiqc.sh) in Bash:

```
# -- LINUX TERMINAL --
# - script_quality_gross_seq_multiqc.sh -
# Generating the script
cd ~/Pipeline/Preproc_RNAseq/1_Qual_control_2-1/FastQC
cat > script_quality_gross_seq_multiqc.sh
#!/bin/bash
cd ~/Pipeline/Preproc_RNAseq/1_Qual_control_2-1/FastQC/Rep_zip_FastQC
multiqc .
mv *_data* ~/Pipeline/Preproc_RNAseq/1_Qual_control_2-1/FastQC/Rep_MultiQC_FastQC
mv *html* ~/Pipeline/Preproc_RNAseq/1_Qual_control_2-1/FastQC/Rep_MultiQC_FastQC

## Hit "Ctrl" plus "z" to save the script
```

We run the script_quality_gross_seq_multiqc.sh via the Linux terminal:

```
# -- LINUX TERMINAL --
# Running the script via terminal
bash ~/Pipeline/Preproc_RNAseq/1_Qual_control_2-1/FastQC/script_quality_gross_seq_multiqc.sh
```

We see in figure 2 that **MultiQC** concatenated the graphics of the files and presented them in an HTML page (directory of the HTML file: ~/Preproc_RNAseq/1_Contr_qual_2-1/FastQC/Rel_MultiQC_FastQC).

Figure 2. presents the quality control before cleaning and filtering the FASTQ files. Some bases at the end of some reads have low quality scores (Figure 2.A), few reads have very low quality scores showing averages less than 5 and it is indicated that the Phred score is greater than 20. The Ns Content (Figure 2.C) was below 0.5%, as well as the content of adapters (Figura 2.D). We saw that the sequencing of the samples was good, but it was necessary to carry out the cleaning and filtering of the reads.

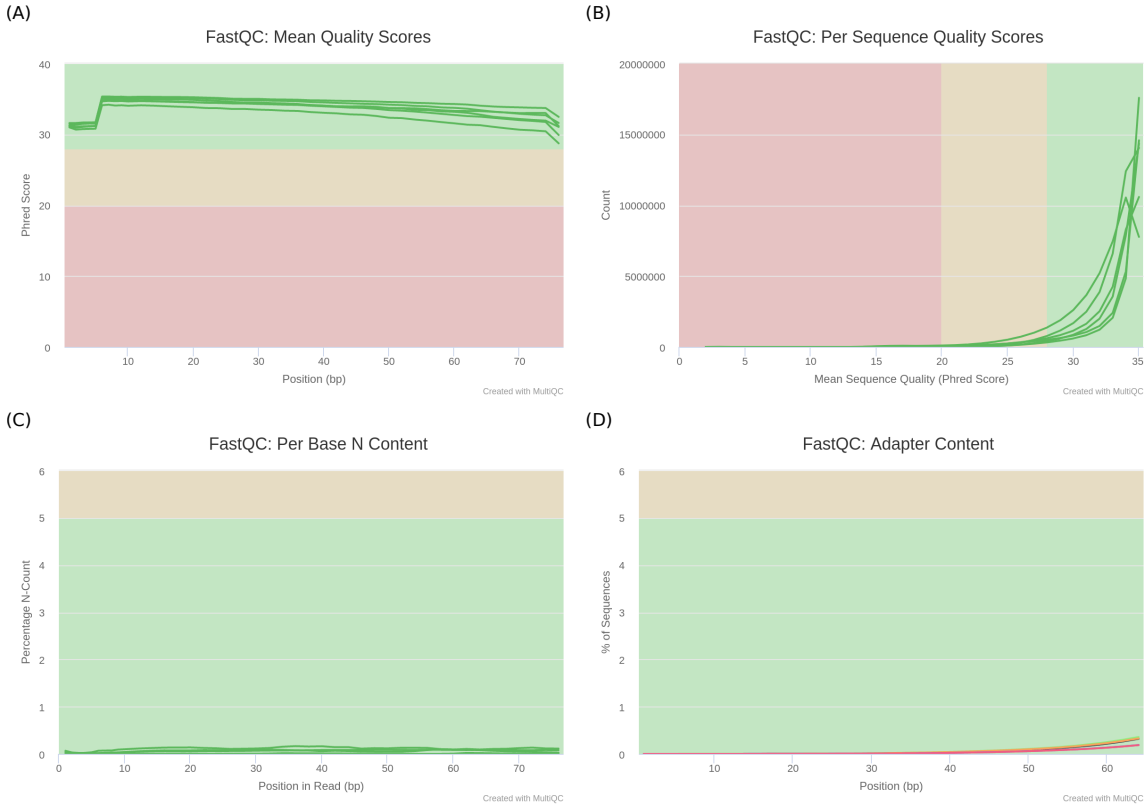

Figure 2: Quality control before cleaning and filtering FASTQ files. (A) Average quality indexes by position before cleaning and filtering RNA-seq data. (B) Distribution of the quality index in all reads. (C) N content in all reads. (D) Adapter content.

6. Cleaning and filtering of RNA-seq data

Before assembling the script for cleaning and filtering the reads, we generated a ".txt" file (samples_names.txt) via the R environment with the names of the FASTQ files without the extension "_1.fastq.gz" or "_2.fastq.gz" so that the filter script could read files R1 and R2 for the same sample.

```
# -- ENVIRONMENT R --
# - samples_names.txt -
# Defining analysis directory
setwd("~/Pipeline/Preproc_RNAseq/2_Filt_cleaning")

# Informing the directory path with the samples
```

```
dir1 <- "~/Pipeline/Preproc_RNAseq/Samples"
list.files(dir1)

# Informing the path of the analysis directory
dir2 <- "~/Pipeline/Preproc_RNAseq/2_Filt_cleaning"
list.files(dir2)

# Generating the .txt file
files <- list.files(paste0(dir1))
samples_names <- unique(gsub("_..fastq.gz", "", files))
write.table(samples_names, "samples_names.txt",
            quote = FALSE,
            row.names = FALSE,
            col.names = FALSE
            )
```

6.1. Unknown sequence of adapters

As the adapters were identified in the FASTQ files, and there were some reads that needed to be filtered for the quality of the nucleotides and the N content, we used the **fastp** tool (Chen *et al.*, 2018) to remove the adapters and clean the reads with low quality indexes. The **fastp** tool filters bad readings, that is, with very low quality, very short or with many Ns. It uses the sliding window method such as Trimmomatic (but faster) when cutting the adapter. Another advantage of this tool is that it is not necessary to enter the adapter string, that is, it automatically detects (Chen *et al.*, 2018) adapters and removes them.

We generated a Bash script using the **fastp** tool to filter and clean the reads. The file "nome_amotras.txt" was used as a reference indicating to the tool that the files R1 and R2 with the same name refer to the same sample.

```
# -- LINUX TERMINAL --
# - script_trimming_fastp.sh -
# Generating the script
cd ~/Pipeline/Preproc_RNAseq/2_Filt_cleaning
cat > script_trimming_fastp.sh
#!/bin/bash
while read SAMP \n
    do \n
        echo "Preprocessing sample ${SAMP}"
        ~/Pipeline/Preproc_RNAseq/Tools/./fastp -i ~/Pipeline/Preproc_RNAseq/Samples/${SAMP}_1.fastq.gz
        -I ~/Pipeline/Preproc_RNAseq/Samples/${SAMP}_2.fastq.gz \
        -o ~/Pipeline/Preproc_RNAseq/Samples_proc_fastp/${SAMP}_1.fastp.fastq.gz \
        -O ~/Pipeline/Preproc_RNAseq/Samples_proc_fastp/${SAMP}_2.fastp.fastq.gz \
        -j ${SAMP}.json -h ${SAMP}.html
    done < ~/Pipeline/Preproc_RNAseq/2_Filt_cleaning/samples_names.txt
mv *.html* ~/Pipeline/Preproc_RNAseq/3_Qual_control_2-2/Fastp/Rep_html_fastp
mv *.json* ~/Pipeline/Preproc_RNAseq/3_Qual_control_2-2/Fastp/Rep_json_fastp

## Hit "Ctrl" plus "z" to save the script
```

We run the script_trimming_fastp.sh script on the Linux terminal.

```
# -- LINUX TERMINAL --
# Running the script
bash ~/Pipeline/Preproc_RNAseq/2_Filt_cleaning/script_trimming_fastp.sh
```

6.2. Known sequence of adapters

If we knew what the adapter strings contained in the files were, we would use the **Cutadapt** tool to remove them. Another time that we would also use this tool is if **fastp** had not detected or removed all the adapters identified by FastQC.

We generated, a Bash script indicating for **Cutadapt** to remove the Illumina universal adapter string.

```
# -- LINUX TERMINAL --
# - script_trimming_cutadapt.sh -
# Generating the script
cd ~/Pipeline/Preproc_RNAseq/2_Filt_cleaning
cat > script_trimming_cutadapt.sh
#!/bin/bash
while read SAMP \n
    do \n
        echo "Processing sample ${SAMP}"
        cutadapt -g GATCGGAAGAGCACACGTCTGAACTCCAGTCACNNNNNNNNNTCTCGTATGCCGTCTTCTGCTTG \
        -m 20 \
        -o ~/Pipeline/Preproc_RNAseq/Samples_proc_Cutadapt/${SAMP}_1.cutadapt.fastq.gz \
        -p ~/Pipeline/Preproc_RNAseq/Samples_proc_Cutadapt/${SAMP}_2.cutadapt.fastq.gz \
        ~/Pipeline/Preproc_RNAseq/Samples/${SAMP}_1.fastq.gz \
        ~/Pipeline/Preproc_RNAseq/Samples/${SAMP}_2.fastq.gz
        done < ~/Pipeline/Preproc_RNAseq/2_Filt_cleaning/samples_names.txt

## Hit "Ctrl" plus "z" to save the script
```

We run script_trimming_cutadapt.sh on the Linux terminal.

```
# -- LINUX TERMINAL --
# Running the script
bash ~/Pipeline/Preproc_RNAseq/2_Filt_cleaning/script_trimming_cutadapt.sh
```

If we had a list of adapters, or if steps 5.1 and 5.2 had not removed all adapters found by FastQC, we would use **Cutadapt** with a list of adapters. We generated a FASTA file with a list of Illumina adapters (can be changed):

```
# -- LINUX TERMINAL --
# - adapters.fasta
# Generating the file
cd ~/Pipeline/Preproc_RNAseq/2_Filt_cleaning
cat > adapters.fasta
>TruSeq Adapter
GATCGGAAGAGCACACGTCTGAACTCCAGTCACNNNNNNNNNTCTCGTATGCCGTCTTCTGCTTG

>TruSeq Universal Adapter
AATGATACGGCGACCACCGAGATCTACACTCTTTCCCTACACGACGCTCTTCCGATCT

>RNA PCR Primer
CAAGCAGAAGACGGCATACGAGATNNNNNNGTGACTGGAGTTCCTTGGCACCCGAGAATTCCA

>Illumina PCR Primer
CAAGCAGAAGACGGCATACGAGATNNNNNNGTGACTGGAGTTC

>Illumina Multiplexing Adapter 1
GATCGGAAGAGCACACGTCT
>Illumina Multiplexing Adapter 2
```

```
ACACTCTTTCCCTACACGACGCTCTTCCGATCT
>Illumina Multiplexing PCR Primer 1.01
AATGATACGGCGACCACCGAGATCTACACTCTTTCCCTACACGACGCTCTTCCGATCT
>Illumina Multiplexing PCR Primer 2.01
GTGACTGGAGTTCAGACGTGTGCTCTTCCGATCT
>Illumina Multiplexing Read1 Sequencing Primer
ACACTCTTTCCCTACACGACGCTCTTCCGATCT
>Illumina Multiplexing Index Sequencing Primer
GATCGGAAGAGCACACGTCTGAACTCCAGTCAC
>Illumina Multiplexing Read2 Sequencing Primer
GTGACTGGAGTTCAGACGTGTGCTCTTCCGATCT

>Illumina Small RNA Adapter 1
GTTCAGAGTTCTACAGTCCGACGATC
>Illumina Small RNA Adapter 2
TCGTATGCCGTCTTCTGCTTGT
>Illumina Small RNA RT Primer
CAAGCAGAAGACGGCATACGA
>Illumina Small RNA PCR Primer 1
CAAGCAGAAGACGGCATACGA
>Illumina Small RNA PCR Primer 2
AATGATACGGCGACCACCGACAGGTTCAGAGTTCTACAGTCCGA
>Illumina Small RNA Sequencing Primer
CGACAGGTTCAGAGTTCTACAGTCCGACGATC

>Illumina Small RNA RT Primer
CAAGCAGAAGACGGCATACGA
>Illumina 5p RNA Adapter
GTTCAGAGTTCTACAGTCCGACGATC
>Illumina RNA Adapter1
TCGTATGCCGTCTTCTGCTTGT

>Illumina Small RNA 3p Adapter 1
ATCTCGTATGCCGTCTTCTGCTTG
>Illumina Small RNA PCR Primer 1
CAAGCAGAAGACGGCATACGA
>Illumina Small RNA PCR Primer 2
AATGATACGGCGACCACCGACAGGTTCAGAGTTCTACAGTCCGA
>Illumina Small RNA Sequencing Primer
CGACAGGTTCAGAGTTCTACAGTCCGACGATC

# Hit "Ctrl" plus "z" to save the script
```

Finally, we generate the script to read the list of adapters and remove those present in the samples.

```
# -- LINUX TERMINAL --
# - script_trimming_cutadapt_list.sh -
# Generating the script
cd ~/Pipeline/Preproc_RNAseq/2_Filt_cleaning
cat > script_trimming_cutadapt_list.sh
#!/bin/bash
while read SAMP \n
    do \n
        echo "Processing sample ${SAMP}"
```

```
        cutadapt -g file:adapters.fasta -m 20 \
        -o ~/Pipeline/Preproc_RNAseq/Samples_proc_Cutadapt/${SAMP}_1.cutadapt.fastq.gz \
        -p ~/Pipeline/Preproc_RNAseq/Samples_proc_Cutadapt/${SAMP}_2.cutadapt.fastq.gz \
        ~/Pipeline/Preproc_RNAseq/Samples/${SAMP}_1.fastq.gz \
        ~/Pipeline/Preproc_RNAseq/Samples/${SAMP}_2.fastq.gz
        done < ~/Pipeline/Preproc_RNAseq/2_Filt_cleaning/samples_names.txt

# Hit "Ctrl" plus "z" to save the script
```

We run script_trimming_cutadapt_list.sh on the Linux terminal.

```
# -- LINUX TERMINAL --
# Running the script
bash ~/Pipeline/Preproc_RNAseq/2_Filt_cleaning/script_trimming_cutadapt_list.sh
```

7. Quality control after cleaning and filtering RNA-seq data

After cleaning and filtering the FASTQ files, it was necessary to check the quality of the files, making sure to check the removal of adapters and bases with low quality indexes.

To generate the files regarding the quality of the sequencing, we built a script (script_quality_fil_seq_fastqc.sh) in Bash.

```
# -- LINUX TERMINAL --
# - script_quality_fil_seq_fastqc.sh -
# Generating the script
cd ~/Pipeline/Preproc_RNAseq/3_Qual_control_2-2/FastQC
cat > script_quality_fil_seq_fastqc.sh
#!/bin/bash
cd ~/Pipeline/Preproc_RNAseq/Samples_proc_fastp
~/Pipeline/Preproc_RNAseq/Tools/FastQC/./fastqc *.gz
mv *zip* ~/Pipeline/Preproc_RNAseq/3_Qual_control_2-2/FastQC/Rep_zip_FastQC_fastp
mv *html* ~/Pipeline/Preproc_RNAseq/3_Qual_control_2-2/FastQC/Rep_html_FastQC_fastp

# Hit "Ctrl" plus "z" to save the script
```

We run script_quality_fil_seq_fastqc.sh on the Linux terminal.

```
# -- LINUX TERMINAL --
# Running the script
bash ~/Pipeline/Preproc_RNAseq/3_Qual_control_2-2/FastQC/script_quality_fil_seq_fastqc.sh
```

As mentioned in topic 4, the directory ~/Preproc_RNAseq/3_Contr_qual_2-2/FastQC/Rel_zip_FastQC contains the file ".zip" referring to the analyzed samples, in it we will have the graphs of each step of the analysis. In the ~/Preproc_RNAseq/3_Contr_qual_2-2/FastQC/Rel_html_FastQC directory, we will have the ".html" file referring to the same samples.

Figure 3 shows the same graphics as the file in Figure 1, but with the reads filtered. We noticed that the average of the quality indexes increased (Figure 3.A), the distribution of the quality indexes started from 18 concentrating the data with indexes greater than 24 (Figure 3.B). Figure 3.C shows that there was no change in the content of Ns and the adapter that had been identified was removed (Figure 1.D).

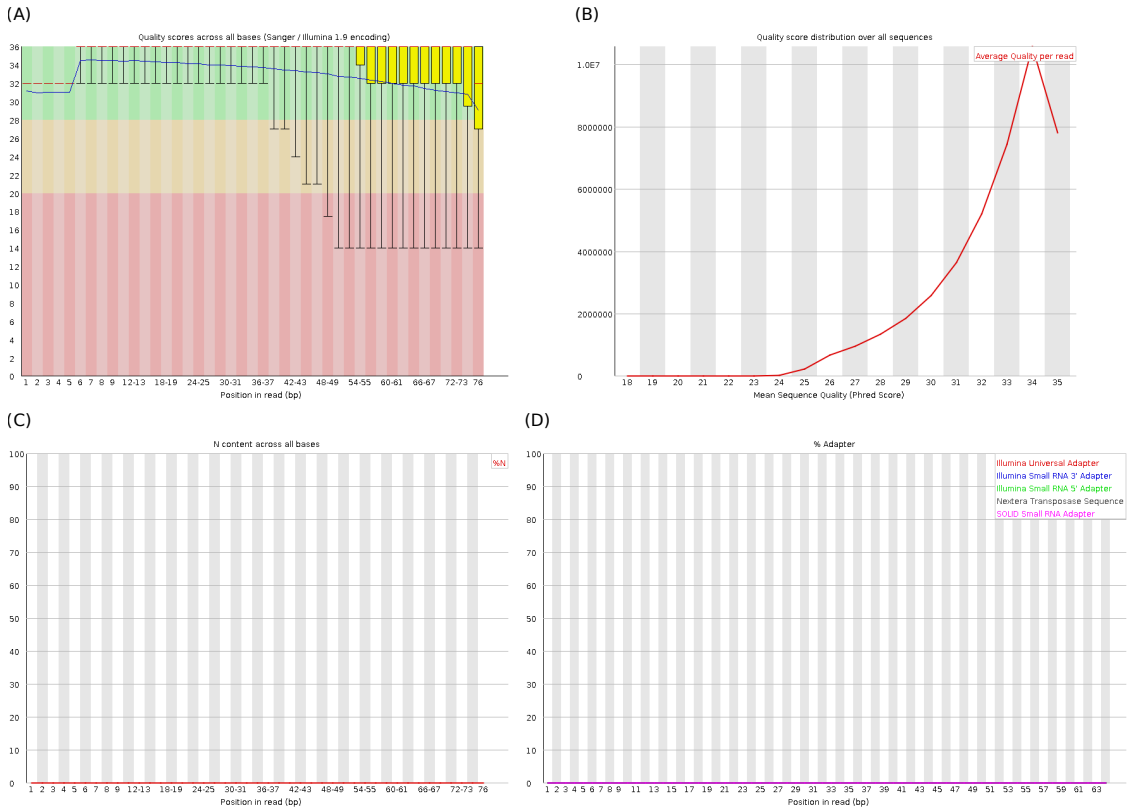

Figure 3: Quality control after cleaning and filtering the sample reads SRR7014308_2. (A) Average quality indexes per position in the sample before cleaning and filtering the RNA-seq data. (B) Distribution of the quality index in all reads. (C) N content in all reads. (D) Adapter content.

We ran the **MultiQC** tool to combine the **FastQC** statistics again. We generated a script (script_quality_fil_seq_multiqc.sh) in Bash.

```
# -- LINUX TERMINAL --
# - script_quality_fil_seq_multiqc.sh -
# Generating the script
cd ~/Pipeline/Preproc_RNAseq/3_Qual_control_2-2/FastQC
cat > script_quality_fil_seq_multiqc.sh
#!/bin/bash
cd ~/Pipeline/Preproc_RNAseq/3_Qual_control_2-2/FastQC/Rep_zip_FastQC_fastp
multiqc .
mv *_data* ~/Pipeline/Preproc_RNAseq/3_Qual_control_2-2/FastQC/Rep_MultiQC_FastQC_fastp
mv *html* ~/Pipeline/Preproc_RNAseq/3_Qual_control_2-2/FastQC/Rep_MultiQC_FastQC_fastp

## Hit "Ctrl" plus "z" to save the script
```

We run the script_quality_fil_seq_multiqc.sh on the Linux terminal:

```
# -- LINUX TERMINAL --
# Running the script
bash ~/Pipeline/Preproc_RNAseq/3_Qual_control_2-2/FastQC/script_quality_fil_seq_multiqc.sh
```

Figure 4 shows the same statistics as the FASTQ files in Figure 2, however, with the FASTQ files cleaned and filtered with **fastp**. It is noticed that the cleaning and filtering with **fastp** was successful, he managed to remove the adapters (Figure 4.D), reduced the Ns content (Figure 4.C) and thus increased the quality indexes of the reads (Figure 4.A and 4.B).

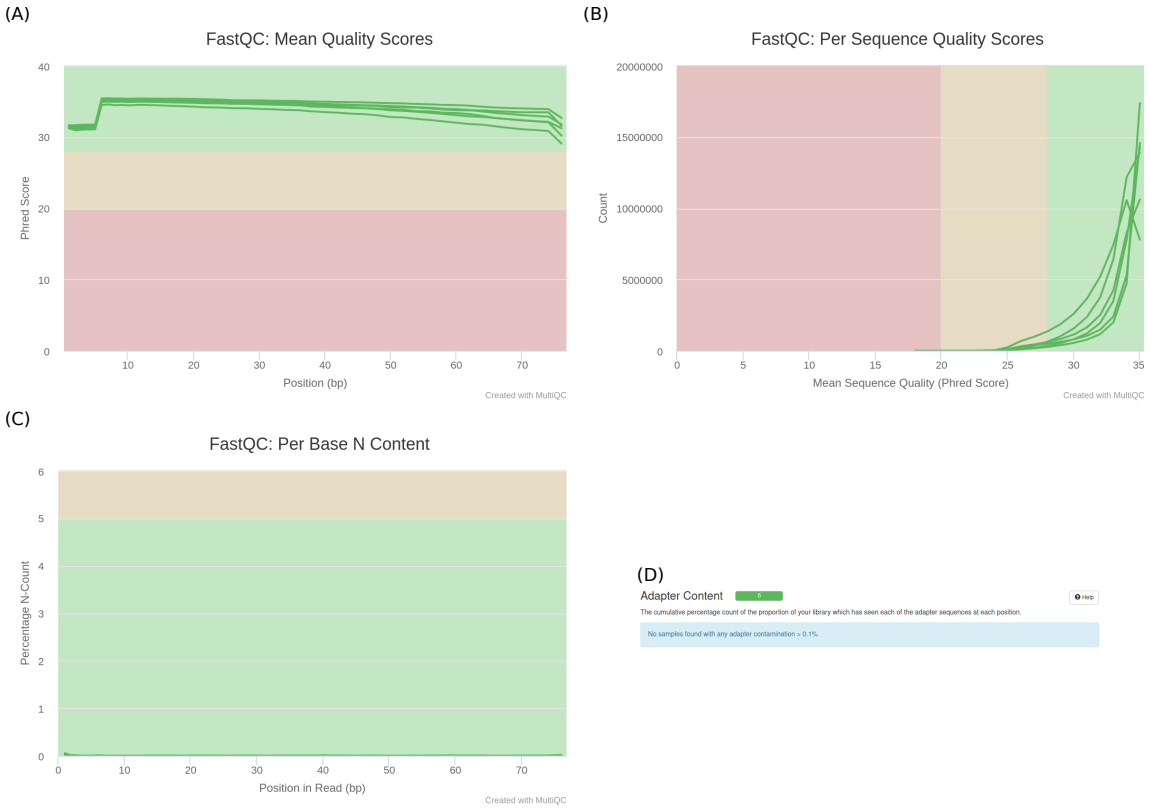

Figure 4: Quality control after cleaning and filtering FASTQ files. (A) Average quality indexes per position in the sample before cleaning and filtering RNA-seq data. (B) Distribution of the quality index in all reads. (C) N content in all reads. (D) Adapter content.

Note: If you followed topic 5.2, change the name of the directories in the scripts so that **FastQC** and **MultiQC** read the correct files.

8. Quantification of transcripts

For the quantification of RNA-seq readings, we used the **Salmon** program - a free software tool that estimates the abundance at the level of transcription from RNA-seq readings (Patro *et al.*, 2017). To map the transcripts, we used the GENCODE reference transcriptome (release 34, GRCh38.13) (Gencode, 2020).

At first we will download the transcriptome mapping for GENCODE release 34 (GRCh38.p13) - you can change the mapping version by changing version 34 to another version - via the R environment.

```
# -- ENVIRONMENT R --
# Defining working directory
setwd("~/Pipeline/Preproc_RNAseq/4_Quant_transc")

# Analysis directory
dir_quant <- "~/Pipeline/Preproc_RNAseq/4_Quant_transc"
list.files(dir_quant)

# Searching the mapping in the repository
reposit <- "ftp://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_human/release_34/"
transcriptome_file <- paste0(reposit, "gencode.v34.transcripts.fa.gz")
```

```
# Mapping Download
download.file(transcriptome_file, "gencode.v34.transcripts.fa.gz")
```

Then, we generate the transcriptome index - it is generated only once and we can use it to quantify another round of samples - for **Salmon**, as it needs a reference transcriptome in multi-FASTA format where each entry provides the sequence of a transcript. **Salmon** uses the transcriptome index to almost map RNA-seq readings during quantification.

```
# -- LINUX TERMINAL --
conda activate salmon
cd ~/Pipeline/Preproc_RNAseq/4_Quant_transc
salmon index --gencode -t gencode.v34.transcripts.fa.gz -i gencode.v34_salmon_1.3.0
conda deactivate
```

To run each sample on **Salmon** it is necessary to create a Bash script with some recommendations provided by the RNA-seq analysis workflow from the Bioconductor (Love, 2019) repository. In order for the **Salmon** to be run for each sample, the file "nome_amostras.txt" generated in the step of cleaning and filtering the RNA-seq data was used.

Script_quantificacao_salmon.sh generated in Bash assuming that the processed readings are in the ~/Preproc_RNAseq/Amostras_proc_fastp directory if you have done the cleaning process with **Cutadapt** join the processed files in a single directory and enter in the argument -1 and -2 of the script.

```
# -- LINUX TERMINAL --
# - script_quantification_salmon.sh -
# Generating the script
cd ~/Pipeline/Preproc_RNAseq/4_Quant_transc
cat > script_quantification_salmon.sh
#!/bin/bash
while read SAMP \n
    do \n
        echo "Processing sample ${SAMP}"
        salmon quant -i gencode.v34_salmon_1.3.0 -l A \
            -1 ~/Pipeline/Preproc_RNAseq/Samples_proc_fastp/${SAMP}_1.fastp.fastq.gz \
            -2 ~/Pipeline/Preproc_RNAseq/Samples_proc_fastp/${SAMP}_2.fastp.fastq.gz \
            -p 4 --gcBias --validateMappings -o Quant_Salmon/${SAMP}_quant
    done < ~/Pipeline/Preproc_RNAseq/2_Filt_cleaning/samples_names.txt

## Hit "Ctrl" plus "z" to save the script
```

We run script_quantification_salmon.sh on the Linux terminal:

```
# -- LINUX TERMINAL --
# Running the script
conda activate salmon
bash ~/Pipeline/Preproc_RNAseq/4_Quant_transc/script_quantification_salmon.sh
conda deactivate
```

9. Construction of the gene expression matrix

For the construction of gene expression matrices, it is necessary to import the quantification generated by **Salmon** into the R environment. In this Pipeline we demonstrate two R/Bioconductor packages for importing the quantification of transcripts: **tximport** (Soneson *et al.*, 2015) and **tximeta** (Love *et al.*, 2019).

9.1. Construction of the gene matrix with the package `tximport`

The R / Bioconductor **tximport** package imports plenty of transcripts and summarizes in matrices (Soneson *et al.*, 2015). We mapped the transcripts to genes using the GENCODE release 34 (GRCh38.p13) (Gencode, 2020) gene annotation, downloaded the GENCODE release 34 (GRCh38.p13) annotation via the R environment and selected the GENEID of the transcripts of the annotation:

```
# -- ENVIRONMENT R --
library(GenomicFeatures)
library(AnnotationDbi)
library(magrittr)

# Defining working directory
setwd("~/Pipeline/Preproc_RNAseq/5_Gen_mat")

# Download annotation
reposit <- "ftp://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_human/release_34/"
annot_file <- paste0(reposit, "gencode.v34.annotation.gff3.gz")
download.file(annot_file, "gencode.v34.annotation.gff3.gz")

# TxDb from the GENCODE annotation
txdb <- makeTxDbFromGFF("gencode.v34.annotation.gff3.gz")

# Get transcript for genetic mapping
k <- keys(txdb, keytype = "TXNAME")
tx2gene <- ensembldb::select(txdb, k, "GENEID", "TXNAME")

# Preview
head(tx2gene) %>%
  tibble::as_tibble() %>%
  knitr::kable("html", booktabs = T) %>%
  kableExtra::kable_styling("striped")
```

To import the quantification of the transcripts into the R environment, we use a vector with the names of the files to be imported by executing the following commands in the R environment:

```
# -- ENVIRONMENT R --
dirquant <- "~/Pipeline/Preproc_RNAseq/4_Quant_transc/Quant_Salmon/"
files <- list.files(dirquant)
files_import <- paste0(dirquant, files, "/quant.sf")
all(file.exists(files_import))
```

Then we import the gene-mapped transcripts into the R environment, transforming them into an R object:

```
# -- ENVIRONMENT R --
library(tximport)
mat_gse <- tximport(files_import,
                    type = "salmon",
                    tx2gene = tx2gene,
                    ignoreAfterBar = TRUE)
```

We named the matrix columns of the `mat_gse` object.

```
# -- ENVIRONMENT R --
names(mat_gse)
samples_names <- gsub("_quant" , "", files)
colnames(mat_gse$counts) <- samples_names
```

```
colnames(mat_gse$abundance) <- samples_names
colnames(mat_gse$length) <- samples_names
```

Finally, we save the `mat_gse` object:

```
# -- ENVIRONMENT R --
save(mat_gse, file = "~/Pipeline/Preproc_RNAseq/5_Gen_mat/mat_gse_tximport.RData")
```

9.2. Construction of the gene matrix with the package `tximeta`

The import of the quantification of the transcripts can also be done using the package R / Bioconductor **tximeta** (Love *et al.*, 2019). The **tximeta** package extends the **tximport** package by automatically adding annotation metadata for transcriptomes (Love, 2019). Below we describe the tutorial on how to import the quantification of transcripts using the R/Bioconductor **tximeta** package.

We downloaded the GENCODE release 34 (GRCh38.p.13) gene annotation via R environment:

```
# -- ENVIRONMENT R --
# Defining working directory
setwd("~/Pipeline/Preproc_RNAseq/5_Gen_mat")

# Download annotation
reposit <- "ftp://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_human/release_34/"
annot_file <- paste0(reposit, "gencode.v34.annotation.gtf.gz")
download.file(annot_file, "gencode.v34.annotation.gtf.gz")
```

We created vectors with the transcript path, gene annotation and the index used by the **Salmon** program via the R environment.

```
# -- ENVIRONMENT R --
dirquants <- "~/Pipeline/Preproc_RNAseq/4_Quant_transc/Quant_Salmon/"
dirannotation <- "~/Pipeline/Preproc_RNAseq/5_Gen_mat/"
dirindex <- "~/Pipeline/Preproc_RNAseq/4_Quant_transc/"
```

We inform the reference files for the **tximeta** to map the transcripts and download the annotation metadata for the transcript used.

```
# -- ENVIRONMENT R --
library(tximeta)
# Defining the paths of GENCODE references
indexDir <- file.path(dirindex, "gencode.v34_salmon_1.3.0")
gtfPath <- file.path(dirannotation, "gencode.v34.annotation.gtf.gz")
fastaFTP <-
  "ftp://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_human/release_34/gencode.v34.
transcripts.fa.gz"

makeLinkedTxome(indexDir=indexDir, source="GENCODE", organism="Homo sapiens",
                release="34", genome="GRCh38", fasta=fastaFTP, gtf=gtfPath, write=FALSE)
```

We imported the sample metadata table into the R environment (download the table "metadata.csv" in the directory: ~/Preproc_RNAseq/Metadata/).

```
# -- ENVIRONMENT R --
library(readr)
coldata <- read_csv("~/Pipeline/Preproc_RNAseq/Metadata/metadata.csv")
```

We incorporated two columns with the names of the samples and the path of the `quant.sf` files respectively, to the set of metadata for the samples.

```r
# -- ENVIRONMENT R --
# Adding file names
coldata$names <- coldata$sample

# Adding quantifications path
coldata$files <- paste0(dirquants, coldata$names, "_quant/", "quant.sf")
all(file.exists(coldata$files))
```

We import the quantification of the transcripts into the R environment.

```r
# -- ENVIRONMENT R --
se <- tximeta(coldata)
```

We summarize the transcription level quantifications for the gene level.

```r
# -- ENVIRONMENT R --
gse <- summarizeToGene(se)
```

Finally, we save the `se` and`gse` objects.

```r
# -- ENVIRONMENT R --
save(se, file = "~/Pipeline/Preproc_RNAseq/5_Gen_mat/mat_se_tximeta.RData")
save(gse, file = "~/Pipeline/Preproc_RNAseq/5_Gen_mat/mat_gse_tximeta.RData")
```

10. Annotation of transcripts

We used the R / Bioconductor **AnnotationHub** (Morgan and Shepherd, 2020) package to annotate the transcripts. Below we present a way to annotate the matrices obtained by **tximeta**, if you are working with the object generated by **tximport** replace the `gse` object with`mat_gse`.

```r
# -- ENVIRONMENT R --
library(AnnotationHub)

# Get row annotation (biotype, etc)
ah <- AnnotationHub()
# query(ah, "EnsDb")
edb <- query(ah, pattern = c("Homo sapiens", "EnsDb", 100))[[1]]
gns <- genes(edb)
EnsDbAnnotation <- as.data.frame(gns)
EnsDbAnnotation <- EnsDbAnnotation[,c("gene_id","symbol","gene_biotype","entrezid")]
dim(EnsDbAnnotation)
colnames(EnsDbAnnotation) <- c("ensemblid","symbol","gene_biotype","entrezid")

# Add row annotation
load("~/Pipeline/Preproc_RNAseq/")
nrow(gse)
gseAnnotation <- rowData(gse)

# Remove ENSEMBL version para tx2gene
rownames(gseAnnotation) <- stringr::str_replace(rownames(gseAnnotation), "\\...$", "")
rownames(gseAnnotation) <- stringr::str_replace(rownames(gseAnnotation), "\\..$", "")

# Match
all(rownames(gseAnnotation)%in%rownames(EnsDbAnnotation))
# [1] TRUE
rowAnnotation <- EnsDbAnnotation[rownames(gseAnnotation),]
```

```
rowAnnotation <- data.frame(gseAnnotation, rowAnnotation, stringsAsFactors = F)
rownames(rowAnnotation) <- rowAnnotation$gene_id
rowData(gse) <- rowAnnotation

# Saving the object
save(gse, file = "~/Pipeline/Preproc_RNAseq/5_Gen_mat/gse_noted.RData")
```

Session information

```
## R version 4.0.2 (2020-06-22)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 20.04.1 LTS
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.9.0
## LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.9.0
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## loaded via a namespace (and not attached):
##  [1] compiler_4.0.2  magrittr_1.5    tools_4.0.2     htmltools_0.5.0
##  [5] yaml_2.2.1      stringi_1.4.6   rmarkdown_2.3   knitr_1.29
##  [9] stringr_1.4.0   xfun_0.15       digest_0.6.25   rlang_0.4.7
## [13] evaluate_0.14
```

References

Andrews,S. *et al.* (2012) FastQC: A quality control tool for high throughput sequence data. *Babraham Institute, version 0.11.5*.

Bache,M.,Stefan and Wickham,H. (2014) Magrittr: A forward-pipe operator for r. *R package version 1.5*.

Chen,S. *et al.* (2018) Fastp: An ultra-fast all-in-one fastq preprocessor. *Bioinformatics*, **34**, i884–i890.

Cornwell,M. *et al.* (2018) VIPER: Visualization pipeline for rna-seq, a snakemake workflow for efficient and complete rna-seq analysis. *BMC Bioinformatics*, **19**, 1–14.

Ewels,P. *et al.* (2016) MultiQC summarize analysis results for multiple tools and samples in a single report. *Bioinformatics*, **32**, 3047–3048.

Gencode (2020) The gencode project: Encyclopedia of genes and gene variants. *Human*, **34**, GRCh38.

Lawrence,M. *et al.* (2013) Software for computing and annotating genomic ranges. *PLoS Computational Biology*, **9**.

Love,M. (2019) RnaseqGene: RNA-seq workflow: Gene-level exploratory analysis and differential expression. *R package version 1.10.0*.

Love,M.I. *et al.* (2019) Tximeta: Reference sequence checksums for provenance identification in rna-seq. *bioRxiv*.

Love,M. *et al.* (2020) Tximeta: Reference sequence checksums for provenance identification in rna-seq. *PLOS Computational Biology*, **16**, 1–13.

Martin,M. (2011) Cutadapt removes adapter sequences from high-throughput sequencing reads. *EMBnet.journal*, **17**, 10–12.

Morgan,M. and Shepherd,L. (2020) AnnotationHub: Client to access annotationhub resources. *R package version 2.20.0*.

Müller,K. and Wickham,H. (2020) Tibble: Simple data frames. *R package version 3.0.1*.

Pagès,H. *et al.* (2019) AnnotationDbi: Manipulation of sqlite-based annotations in bioconductor. *R package version 1.48.0*.

Patro,R. *et al.* (2017) Salmon provides fast and bias-aware quantification of transcript expression. *Nature Methods*, **14**, 417–419.

Rainer,J. *et al.* (2019) Ensembldb: An r package to create and use ensembl-based annotation resources. *Bioinformatics*.

Soneson,C. *et al.* (2015) Differential analyses for rna-seq: Transcript-level estimates improve gene-level inferences [version 1; peer review: 2 approved]. *F1000Research*, **4**, 1–18.

Wang,Z. *et al.* (2009) RNA-seq: A revolutionary tool for transcriptomics. *Nature Reviews Genetics*, **10**, 57–63.

Wickham,H. *et al.* (2018) Readr: Read rectangular text data. *R package version 1.3.1*.

Xie,Y. (2020) Knitr: A general-purpose package for dynamic report generation in r. *R package version 1.29*.

Zhu,H. (2019) KableExtra: Construct complex table with 'kable' and pipe syntax. *R package version 1.1.0*.