

# Complete bulk-RNA-seq pipeline. Part 1: fastq to count matrix

Author: Tomàs Montserrat Ayuso

April 10, 2023

## Introduction

In the next two posts we will look at how to perform a complete bulk RNA-seq pipeline, from *fastq* files to statistical analysis. In this post we will see all the necessary steps to get the count matrix from the *fastq* files of each sample. In a simplified form, we saw how to do this for a single sample in [this previous post](#). Now, however, we will look at a more realistic scenario where we do not have just one *fastq* file, but 98, counting technical and biological replicates that are paired-end. In the following post we will study how to perform a differential gene expression analysis with DESeq2 of these data.

We will use the data from GEO entry [GSE124975](#), which correspond to the study by Marjanovic et al. (preprint: [Salinomycin disturbs Golgi apparatus function and specifically affects cells in epithelial-to-mesenchymal transition](#)).

The authors of this study were interested in studying the effect of the drug salinomycin on the HMLE cell line, used to study breast cancer. They had HMLE-Twist and HMLE-pBp cells, which differ in their ability to carry out epithelial-mesenchymal transition (EMT), which is very important in cancer progression. While HMLE-Twist cells express EMT markers, HMLE-pBp cells do not.

The design of this research has 4 different sample groups: HMLE-Twist and HMLE-pBp cells grown in the presence or absence of salinomycin. For each experimental condition we have 3 biological replicates, and for each sample, sequencing has been carried out in 4 different runs (which can be considered technical replicates). In total we have approximately 690 million reads. In addition, according to the authors' article, the two experimental conditions are paired (cells treated and not treated with salinomycin come from the same cell culture). We will have to keep this in mind when performing the statistical tests for differential gene expression.

## Pipeline

The pipeline we will follow to carry out the processing and analysis of the data is as follows:

- Download data from ENA
- Quality control using *fastqc*
- Alignment of the sequences against the reference genome with *hisat2*
- Generation of the count matrix with *featureCounts*
- Combination of technical replicates by adding the counts with *DESeq2*

- Statistical analysis of the results with *DESeq2*

Since we will be generating quite a few intermediate files, it is necessary to be tidy throughout the analysis. So, our directory should have a structure similar to this:

```


.
├── alignment
├── data
├── quality
├── quantification
└── scripts

```

In each directory we will save the different outputs that we will get in each step.

## Data download from ENA

To download the raw data, that is, the *fastq* files with the reads of each sample, we will first access [Gene Expression Omnibus](#) (GEO) and look for the repository with the accession number provided by the authors in the article: GSE124975.



The screenshot shows the NCBI GEO Accession Display page for GSE124975. The page header includes the NCBI logo and the GEO logo (Gene Expression Omnibus). Navigation links include HOME, SEARCH, SITE MAP, GEO Publications, FAQ, MIAME, and Email GEO. The breadcrumb trail is NCBI > GEO > Accession Display. The user is not logged in, with a login link provided. The search criteria are: Scope: Self, Format: HTML, Amount: Quick, GEO accession: GSE124975, and a GO button. The series information is displayed below the search criteria.

Series GSE124975		<a href="#">Query DataSets for GSE124975</a>
Status	Public on Jan 11, 2023	
Title	Salinomycin induces expression of ER and Golgi apparatus related genes in EMT cell model	
Organism	<a href="#">Homo sapiens</a>	
Experiment type	Expression profiling by high throughput sequencing	
Summary	Epithelial-to-mesenchymal transition (EMT) gives rise to cells with properties similar to cancer stem cells (CSCs) that drive tumor metastasis. Recently, a screening of a large compound library on a breast EMT model has identified salinomycin, a K+/H+ ionophore, as a highly selective drug towards CSCs. We used the same EMT model to show that salinomycin targets Golgi apparatus. We have performed RNA-seq analysis on HMLE-Twist and HMLE-pBp cells (EMT and non-EMT) that were either mock treated or treated for 24h with micro molar concentration (0.2uM) of salinomycin. Salinomycin induced expression of genes enriched by known ER and Golgi stressors.	

Once there, look for the *BioProject* code, which is PRJNA514730. With this information we can now go to the [European Nucleotide Archive](#) (ENA) and enter the accession number.

## Project: PRJNA514730

Epithelial-to-mesenchymal transition (EMT) gives rise to cells with properties similar to cancer stem cells (CSCs) that drive tumor metastasis. Recently, a screening of a large compound library on a breast EMT model has identified salinomycin, a K<sup>+</sup>/H<sup>+</sup> ionophore, as a highly selective drug towards CSCs. We used the same EMT model to show that salinomycin targets Golgi apparatus. We have performed RNA-seq analysis on HMLE-Twist and HMLE-pBp cells (EMT and non-EMT) that were either mock treated or treated for 24h with micro molar concentration (0.2uM) of salinomycin. Salinomycin induced expression of genes enriched by known ER and Golgi stressors. Overall design: Total RNA from biological triplicates from mock and salinomycin treated HMLE-Twist and HMLE-pBp cells was extracted (12 biological samples in total). Library was prepared with TrueSeq Stranded mRNA kit (Illumina) from 750 ng of RNA. Library was sequenced on a NextSeq500 using High output flow-cell in pair-end mode (75+75 cycles). Raw data yielded total over 920 million high-quality reads (Q30 > 75%).

Show Less

View:

XML  
XML (STUDY)

Download:

XML  
XML (STUDY)

Navigation:

Show

Read Files:

Hide

Parent Projects:

Show

We can now download all the *fastq* files. The reads we will work with are paired-end (we have both ends of the fragments sequenced), so for each sample there are two *fastq* files.

Read Files

Show Column Selection

Download report:
JSON
TSV
Download Files as ZIP
Download selected files

Study Accession	Sample Accession	Experiment Accession	Run Accession	Tax Id	Scientific Name	Generated FASTQ files: FTP	Sub
PRJNA514730	SAMN10734055	SRX5242845	SRR8435265	9606	Homo sapiens	<input type="checkbox"/> SRR8435265.fastq.gz <input type="checkbox"/> SRR8435265_1.fastq.gz <input type="checkbox"/> SRR8435265_2.fastq.gz	
PRJNA514730	SAMN10734055	SRX5242845	SRR8435266	9606	Homo sapiens	<input type="checkbox"/> SRR8435266_1.fastq.gz <input type="checkbox"/> SRR8435266_2.fastq.gz	
PRJNA514730	SAMN10734055	SRX5242845	SRR8435267	9606	Homo sapiens	<input type="checkbox"/> SRR8435267_1.fastq.gz <input type="checkbox"/> SRR8435267_2.fastq.gz	
PRJNA514730	SAMN10734055	SRX5242845	SRR8435268	9606	Homo sapiens	<input type="checkbox"/> SRR8435268_1.fastq.gz <input type="checkbox"/> SRR8435268_2.fastq.gz	

We will save all these files in the data folder.

## Quality control of the reads

The next step in our analysis is to perform the quality control of the reads. We will do it, as we have done in other posts, with the *fastqc* program. This program works at sample level and generates a report in *html* format for each of them with the results of several quality metrics of the reads:

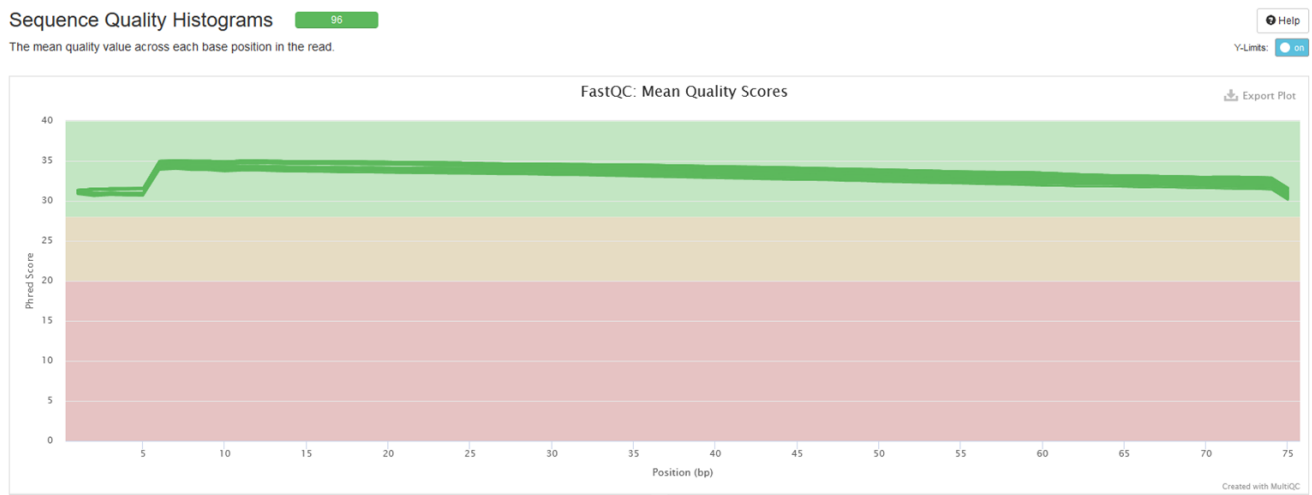
```
for i in SRR8435265 SRR8435266 SRR8435267 (...);  
do  
    fastqc data/${i}/${i}_1.fastq.gz data/${i}/${i}_2.fastq.gz -o quality/  
done
```

*fastqc* accepts as many arguments as *fastq* files we have, no for loop is needed. Here we have only used it for a matter of space.

Since we have many samples (between biological and technical replicates we have a total of 98 *fastq* files, and thus 98 *fastqc* reports), we will use *multiqc* that automatically groups all reports into one so that we can study them together:

```
multiqc quality/fastqc/ -o multiqc_quality/
```

This program needs only two arguments: the directory where the reports are located and the directory where we want the report. If we open the generated *html* file we can study the quality of our short sequences. The most interesting characteristic of these data is the high quality of the reads, a fact that coincides with what the authors state in the article. There are also no obvious problems in either sample or the presence of adapter sequences:



The *Per Base Sequence Content* section is the worst performing. However, this is normal for RNA-seq data, as explained in the [fastqc](#) manual.

We will not go into details of the rest of the sections. You can consult the *fastqc* manual for more information.

The question of what to do next arises from the read quality reports: should we remove duplicate sequences and poor-quality bases (those with a relatively high probability of being missequenced)?

Regarding duplicated sequences, the current consensus is that they should not be removed for differential expression studies with RNA-seq. As demonstrated in [this study](#), removing duplicate sequences does not improve results, but would only worsen statistical power. On the other hand, removing bad quality bases by trimming (cutting them at the ends until the desired quality is reached) is currently not necessary because aligners such as *hisat2* do soft-clipping of the bases that do not align with the reference genome, as explained in the [Harvard Chan Bioinformatics Core](#) training materials. The conclusion of this other [article](#) also advises against trimming in RNA-seq studies.

## Alignment with the reference genome

Once we are satisfied with the quality of the data, we can now align the reads with the reference genome to know which part of it they come from. To do this, we need to use a splice-aware aligner to take into account those sequences that have an unsequenced intron in the middle. The one we will use is *hisat2*. If we have not downloaded the indexed reference genome yet, we can do so with this command:

```
wget https://genome-id3.s3.amazonaws.com/hisat/grch38_genome.tar.gz \
-P hisat2/
tar -xvzf grch38_genome.tar.gz
```

Now we have everything to perform the alignment of the sequences:

```
for i in SRR8435265 SRR8435266 SRR8435267 (...);
do
    echo "Aligning ${i}."
    hisat2 -x ../hisat2/grch38/genome \
        -1 data/${i}/${i}_1.fastq.gz -2 data/${i}/${i}_2.fastq.gz \
        -t \
        -p 6 \
        --rna-strandness RF \
        | samtools sort -o alignment/${i}.bam
done
```

The `-t` option is used to display the time required for the operations, while the `-p` option specifies the number of processors that the program can use. The `-1` and `-2` options are for specifying the two reads of the fragment.

Finally, the `--rna-strandness` option allows us to specify the orientation of each read in the fragment if the protocol used maintains this information. The cDNA libraries for obtaining the data were prepared with the Illumina TrueSeq Stranded mRNA kit, in which the second read has the same orientation as the transcript and the first read is the reverse complement. As we see in the code, we pass the alignment result directly to *samtools sort* to sort and convert it into bam format.

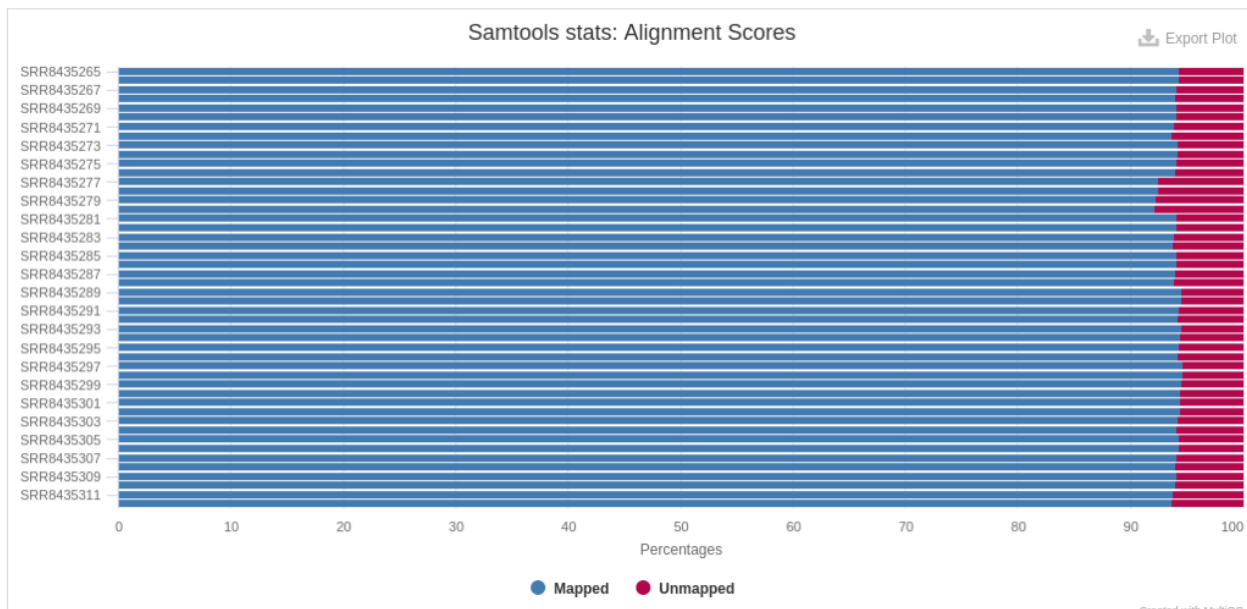
Once we have the bam files of the alignment, it is convenient to run *samtools stats*. The generated files (one for each sample) can also be processed by *multiqc*, adding them to the quality report:

```
for i in SRR8435265 SRR8435266 SRR8435267 (...);
do
    samtools stats alignment/${i}.bam > quality/alignment/${i}.txt
done
```

If we now use *multiqc* again, we can see the data from these new files in the quality report:

```
multiqc quality/fastqc/ quality/alignment/ -o multiqc_quality/
```

Perhaps the most important part of *bam* file statistics is the percentage of mapped reads for each sample. Ideally, we hope to obtain a percentage of mapped reads greater than 90% for each sample. The percentages of our data are around 95%, so this is very good news.



## Quantification of gene expression

Now we have everything ready to obtain a matrix with the gene expression of each sample. The *bam* files contain the reads with the coordinates of the reference genome they come from. We first download the *gtf* file with the annotations:

```
wget http://ftp.ensembl.org/pub/release-
107/gtf/homo_sapiens/Homo_sapiens.GRCh38.107.gtf.gz \ -P data/

gunzip data/Homo_sapiens.GRCh38.107.gtf.gz
```

It is very important that the version of the reference genome in this file and the version of the reference genome used in the alignment are the same, otherwise the coordinates will not match.

It is also important to check that the names of the chromosomes are the same in the *gtf* file and the *bam* files. We can quickly check this with the following commands in the terminal:

```
head ../gtf/Homo_sapiens.GRCh38.107.gtf | grep -v "#" | cut -f1,3,4,5

1      gene      1471765      1497848

1      transcript 1471765      1497848

1      exon      1471765      1472089

1      CDS       1471885      1472089

1      start_codon 1471885      1471887


samtools index alignment/SRR8435265.bam

samtools idxstats alignment/SRR8435265.bam | cut -f1-3 | head -n5

1      248956422    908435

10     133797422     462794

11     135086622     554481

12     133275309     764466

13     114364328     139946
```

The first column of the above outputs is the name of the chromosomes. So, the name encoding is the same.

We will use *featureCounts* to quantify gene expression. This program basically needs two inputs: the *gtf* file with the genome annotations (the genes and the coordinates of each gene) and the *bam* files. The program will count, for each sample, how many reads overlap each gene and return an array with the genes in the rows and the samples in the columns:

```

featureCounts \

-T 6 \

-p \

-a ../gtf/Homo_sapiens.GRCh38.107.gtf \

-o quantification/count_matrix.txt \

-t exon \

-s 2 \

alignment/*.bam

```

The `-t` option specifies that we are interested in counting reads that fall into exons, while the `-s` option informs the program of the orientation of the reads.

The program generates two text files: the count matrix (*txt*) and the summary. We can take a look at the header of the count matrix:

```

Geneid Chr    Start  End    Strand Length alignment/SRR8435265.bam
alignment/SRR8435266.bam alignment/SRR8435267.bam alignment/SRR8435268.bam
alignment/SRR8435269.bam alignment/SRR8435270.bam alignment/SRR8435271.bam
alignment/SRR8435272.bam alignment/SRR8435273.bam alignment/SRR8435274.bam
alignment/SRR8435275.bam alignment/SRR8435276.bam alignment/SRR8435277.bam
alignment/SRR8435278.bam alignment/SRR8435279.bam alignment/SRR8435280.bam
alignment/SRR8435281.bam alignment/SRR8435282.bam alignment/SRR8435283.bam
alignment/SRR8435284.bam alignment/SRR8435285.bam alignment/SRR8435286.bam
alignment/SRR8435287.bam alignment/SRR8435288.bam alignment/SRR8435289.bam
alignment/SRR8435290.bam alignment/SRR8435291.bam alignment/SRR8435292.bam
alignment/SRR8435293.bam alignment/SRR8435294.bam alignment/SRR8435295.bam
alignment/SRR8435296.bam alignment/SRR8435297.bam alignment/SRR8435298.bam
alignment/SRR8435299.bam alignment/SRR8435300.bam alignment/SRR8435301.bam
alignment/SRR8435302.bam alignment/SRR8435303.bam alignment/SRR8435304.bam
alignment/SRR8435305.bam alignment/SRR8435306.bam alignment/SRR8435307.bam
alignment/SRR8435308.bam alignment/SRR8435309.bam alignment/SRR8435310.bam
alignment/SRR8435311.bam alignment/SRR8435312.bam

```

Columns 2 to 6 are not of interest to us for differential gene expression analysis, so we will eliminate them:

```

cat quantification/count_matrix.txt | cut -f1,7-55 | \

grep -v "#" > quantification/genes_count_matrix.txt

```

Now we can look at the count matrix and see that it has the format we expect (gens in the rows and samples in the columns):



```
head genes_count_matrix.txt | cut -f1-4
```

```
Geneid alignment/SRR8435265.bam alignment/SRR8435266.bam
alignment/SRR8435267.bam
```

ENSG00000160072	120	196	206
ENSG00000234396	2	1	0
ENSG00000225972	0	0	0
ENSG00000224315	0	0	0
ENSG00000198744	2	8	4
ENSG00000279928	1	0	2
ENSG00000228037	0	0	0
ENSG00000142611	0	0	0
ENSG00000225630	385	413	427

Although the samples name contains the full path to the file, this is a minor problem that we will easily fix when we have the data in R.

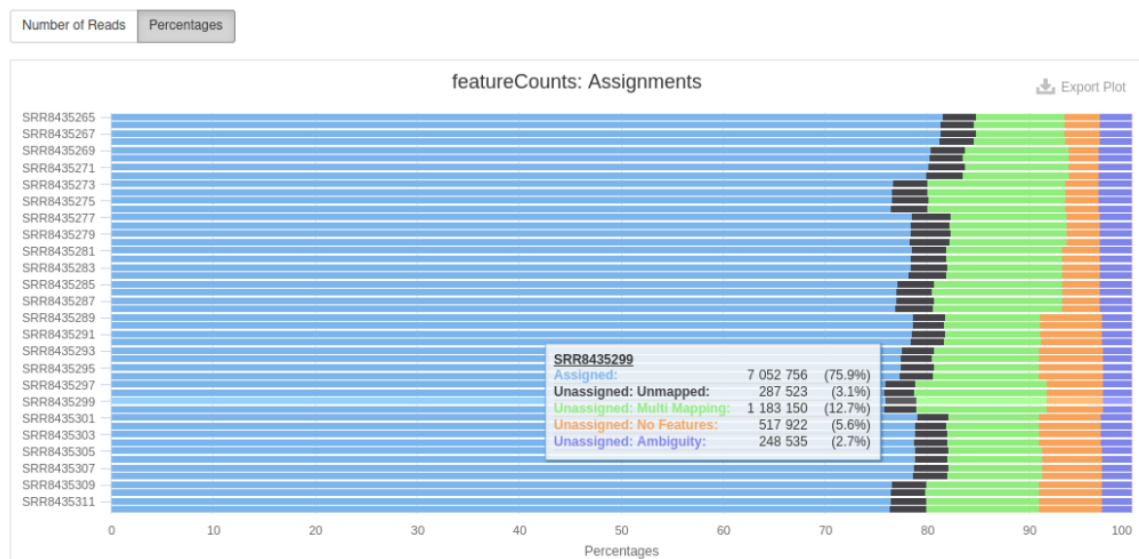
Now, the last thing left for us to do is to incorporate the information from the summary file into the report of *multiqc*:

```
multiqc quality/fastqc/ quality/alignment/ quantification/ -o multiqc_quality/
```

Between 75% and 80% of the reads have been assigned to a gene, this numbers seem high enough to proceed with the statistical analysis of the data:

## featureCounts

**Subread featureCounts** is a highly efficient general-purpose read summarization program that counts mapped reads for genomic features such as genes, exons, promoter, gene bodies, genomic bins and chromosomal locations. DOI: 10.1093/bioinformatics/btt656.



## Conclusion

In this first part of the complete RNA-seq bulk data analysis pipeline we have seen how to do the processing of all *fastq* files at once. First, we did the quality control of the reads, where we verified that the data is of very good quality. We then aligned the reads against the reference genome to determine the origin of each of them. With this information in the *bam* files, we were able to quantify gene expression by generating the count matrix that we will use in the next post to perform the differential gene expression analysis. The *multiqc* program has helped us to visualize, jointly for all the samples and in a single report, the statistics of the quality measures of the reads, the alignments and the quantification of gene expression.

All the code used in this analysis is on [GitHub](#).