RNAseq tutorial

by
Roseric Azondekon, PhD
University of Wisconsin Milwaukee

May 6, 2019

Background

In this tutorial, we show you how to download raw sequence data from the European instance of the SRA, which can be accessed via `https://www.ebi.ac.uk/ena`. At ENA, the sequencing reads are directly available in FASTQ or SRA formats, which will be explained below.

For this tutorial, it is required that `FastQC`, `multiQC`, the `SRA toolkit`, the `subread` package, a powerful suite of tools designed to interact with SAM and BAM files called `samtools`, `salmon`, and `STAR` are installed and referenced in the environment variable `PATH`. Let's first check if this requirement is met:

```
In [ ]: fastqc --version
```

```
In [ ]: multiqc --version
```

```
In [ ]: fastq-dump --version
```

```
In [ ]: samtools --version
```

```
In [ ]: salmon -v
```

```
In [ ]: STAR --version
```

```
In [ ]: # Checking if the package is properly installed
        featureCounts -v
```

If at least one of the above commands produces an error, please, check your installation of the tool and try again.

Now let's create a working directory for our RNA-seq project.

```
In [ ]: mkdir -p tuto && cd tuto
```

1. Data Download

To download a set of SRA files: 1. Go to https://www.ebi.ac.uk/ena. 2. Search for the accession number of the project, e.g., SRP053046 (should be indicated in the published paper). 3. There are several ways to start the download, here we show you how to do it through the command line interface on GNU/Linux: - copy the link's address of the "SRA files" column (right mouse click), go to the command line, move to the target directory, type: `wget <link copied from the ENA website>` - If there are many samples as it is the case for the project referenced here (accession number: SRP053046), you can download the summary of the sample information from ENA by right-clicking on "TEXT" and copying the link location.

Now let's download the file from the link copied earlier.

```
In [ ]: wget -O all_samples.txt "https://www.ebi.ac.uk/ena/data/warehouse/filereport?\
        accession=PRJNA274258&result=read_run&fields=study_accession,\
        sample_accession,secondary_sample_accession,experiment_accession,\
        run_accession,tax_id,scientific_name,instrument_model,library_layout,fastq_ftp,\
        fastq_galaxy,submitted_ftp,submitted_galaxy,sra_ftp,sra_galaxy,cram_index_ftp,\
        cram_index_galaxy&download=txt"
```

You may try to open the `all_samples.txt` file with LibreOffice or Excel to view it. For this project, we are only interested in the paired-end first 9 RNA-seq samples. Since the first line in `all_samples.txt` contains the header, we will generate another file containing only the first 9 lines of `all_samples.txt` with the following command:

```
In [ ]: sed '1d' all_samples.txt > all_samples.txt2
        head -9 all_samples.txt2 > samples.txt
        rm all_samples.txt2
```

Now, let's create a new folder for our SRA files.

```
In [ ]: mkdir -p sra_files
```

According to https://www.ncbi.nlm.nih.gov/books/NBK158899/, the FTP root to download files from NCBI is ftp://ftp-trace.ncbi.nih.gov/ and the remainder path follows the specific pattern `/sra/sra-instant/reads/ByRun/sra/{SRR|ERR|DRR}/<first 6 characters of accession>/<accession>/<accession>.sra`.

Notice that the accession number for the SRA files are located in the 5th column "Run accession" in `all_samples.txt`. We proceed to the download of the SRA files of the samples listed in `samples.txt` with the following code: (**Attention: The download may take a long time!**)

```
In [ ]: #The command below may take too long to download.
        cut -f5 samples.txt | xargs -i sh -c \
        'v={}; FTPROOT=ftp://ftp-trace.ncbi.nih.gov/;\
               REM=sra/sra-instant/reads/ByRun/sra/;\
               url=${FTPROOT}${REM}${v:0:3}/${v:0:6}/${v}/${v}.sra;\
               wget $url -P sra_files'
```

2. Converting SRA files to FASTQ files

Now that the download is complete, let's convert the SRA files into FASTQ files with the following command: (**Attention: This may take a long time!**)

```
In [ ]: cut -f5 samples.txt | xargs -i sh -c \
        'v={}; fastq-dump --outdir fastq/${v} --gzip \
                          --skip-technical --split-3 sra_files/${v}.sra'
```

This should be the file structure of your working directory up to this point:

```
.
|-- all_samples.txt
|-- samples.txt
|-- fastq
|   |-- SRR1784137
|   |   -- SRR1784137_1.fastq.gz
|   |   -- SRR1784137_2.fastq.gz
|   |-- SRR1784138
|   |   -- SRR1784138_1.fastq.gz
|   |   -- SRR1784138_2.fastq.gz
|   |-- SRR1784139
|   |   -- SRR1784139_1.fastq.gz
|   |   -- SRR1784139_2.fastq.gz
|   |-- SRR1784140
|   |   -- SRR1784140_1.fastq.gz
|   |   -- SRR1784140_2.fastq.gz
|   |-- SRR1784141
|   |   -- SRR1784141_1.fastq.gz
|   |   -- SRR1784141_2.fastq.gz
|   |-- SRR1784142
|   |   -- SRR1784142_1.fastq.gz
|   |   -- SRR1784142_2.fastq.gz
|   |-- SRR1784143
|   |   -- SRR1784143_1.fastq.gz
|   |   -- SRR1784143_2.fastq.gz
|   |-- SRR1784144
|   |   -- SRR1784144_1.fastq.gz
|   |   -- SRR1784144_2.fastq.gz
|   |-- SRR1784145
|       -- SRR1784145_1.fastq.gz
|       -- SRR1784145_2.fastq.gz
|-- sra_files
    -- SRR1784137
    -- SRR1784137.sra
    -- SRR1784138
    -- SRR1784138.1
    -- SRR1784138.sra
    -- SRR1784139
    -- SRR1784139.sra
    -- SRR1784140.sra
    -- SRR1784141.sra
    -- SRR1784142.sra
    -- SRR1784143.sra
    -- SRR1784144.sra
    -- SRR1784145.sra
```

3. Quality Control of the FASTQ files

Up to this point, we have all our RNA-seq FASTQ files ready for Quality Control (QC) check. This is done with the `fastqc` tools developed by the Babraham Institute. Run the following command to perform QC check for all the samples: (**This may take some time!**)

```
In [ ]: cut -f5 samples.txt | xargs -i sh -c \
        'v={}; mkdir -p fastqc_reports/${v};\
              fastqc fastq/${v}/*fastq.gz -o fastqc_reports/${v}'
```

Next, let's summarize the QC reports (for all the samples) into one unique report using `multiqc`:

```
In [ ]: multiqc fastqc_reports --dirs -o multiQC_report/
```

Let's examine the summary `multiqc` report either by double-clicking on `multiQC_report/multiqc_report.html` or by executing the following code:

```
In [ ]: xdg-open multiQC_report/multiqc_report.html
```

4. Read Alignment

In order to identify the transcripts that are present in a specific sample, the genomic origin of the sequenced cDNA fragments must be determined. The assignment of sequencing reads to the most likely locus of origin is called read alignment or mapping and it is a crucial step in most types of high-throughput sequencing experiments.

The general challenge of short read alignment is to map millions of reads accurately and in a reasonable time, despite the presence of sequencing errors, genomic variation and repetitive elements. The different alignment programs employ various strategies that are meant to speed up the process (e.g., by indexing the reference genome) and find a balance between mapping fidelity and error tolerance.

4.1. Reference genomes and annotation

Genome sequences and annotation are often generated by consortia such as (mod)ENCODE, The Mouse Genome Project, The Berkeley Drosophila Genome Project, and many more. The results of these efforts can either be downloaded from individual websites set up by the respective consortia or from more comprehensive data bases such as the one hosted by the University of California, Santa Cruz (UCSC) or the European genome resource (Ensembl).

Reference sequences are usually stored in plain text FASTA files that can either be compressed with the generic gzip command. The annotation file is often stored as a GTF (Gene Transfer Format) despite the availability of several other file formats.

Both reference sequences and GTF annotation file can be obtained either from NCBI, ENSEMBL or UCSC Genome Browser.

While it is usually faster to align against a cDNA reference sequence (cDNA), in this tutorial, we will align the reads against the genome (DNA) reference sequences. We obtain both the genome reference sequences and our gene annotation files from ENSEMBL.

```
In [ ]: # Download the latest human genome
        wget -P reference \
        ftp://ftp.ensembl.org/pub/release-93/fasta/homo_sapiens\
        /dna/Homo_sapiens.GRCh38.dna.primary_assembly.fa.gz
```

```
In [ ]: # Decompress genome sequence file and rename it
        gzip -dk < reference/Homo_sapiens.GRCh38.dna.\
        primary_assembly.fa.gz > reference/human_genome.fa
```

```
In [ ]: # Download GTF annotation file
        wget -P annotation ftp://ftp.ensembl.org/pub/release-95\
        /gtf/homo_sapiens/Homo_sapiens.GRCh38.95.gtf.gz
```

```
In [ ]: # Decompress gene annotation file and rename it
        gzip -dk < annotation/Homo_sapiens.GRCh38.95.gtf.gz\
        > annotation/gene_annotation.gtf
```

4.2. Aligning reads using STAR

4.2.1. Generate genome index

This step has to be done only once per genome type (and alignment program). The index files will comprise the genome sequence, suffix arrays (i.e., tables of k-mers), chromosome names and lengths, splice junctions coordinates, and information about the genes (e.g. the strand). Therefore, the main input for this step encompasses the reference genome and an annotation file.

The index creation may take a long time!

```
In [ ]: # create a directory to store the index in
        mkdir -p STARindex
```

```
In [ ]: # Run STAR in "genomeGenerate" mode with 10 threads
        # Feel free to change the 'runThreadN' parameter
        # depending on the number of cores available on your computer
        STAR --runThreadN 10 \
             --runMode genomeGenerate --genomeDir STARindex \
             --genomeFastaFiles reference/human_genome.fa \
             --genomeChrBinNbits 15 \
             --sjdbGTFfile annotation/gene_annotation.gtf \
             --sjdbOverhang 49
```

4.2.2. Alignment

This step has to be done for each individual FASTQ file.

This step may take a long time!

```
In [ ]: # create a directory to store the alignment files
        mkdir -p alignment_STAR
```

```
In [ ]: # execute STAR in the runMode "alignReads"
        cut -f5 samples.txt | xargs -i sh -c \
        'v={}; STAR --genomeDir STARindex \
                    --readFilesIn fastq/${v}/*fastq.gz \
                    --readFilesCommand zcat \
                    --outFileNamePrefix alignment_STAR/${v} \
                    --outFilterMultimapNmax 1 \
                    --outReadsUnmapped Fastx \
                    --outSAMtype BAM SortedByCoordinate \
                    --twopassMode Basic \
                    --runThreadN 10'
```

4.2.3. BAM file indexing

Most downstream applications like the Integrative Genomics Viewer (IGV) will require a .BAM.BAI file together with every BAM file to quickly access the BAM files without having to load them into memory.

To generate these index files, let's now index all the BAM files within the 'alignment_STAR' folder using `samtools`.

```
In [ ]: for i in alignment_STAR/*; do
            if [ "${i}" != "${i%.bam}" ];then
                samtools index ${i}
            fi
        done
```

4.3. Read alignment assessment

There are numerous ways to do basic checks of the alignment success. An alignment of RNA-seq reads is usually considered to have succeeded if the mapping rate is >70%.

The very first QC of aligned reads should be to generally check the aligner's output. We can perform a basic assessment of the read alignment for all our samples with using the `samtools flagstat` command:

```
In [ ]: for i in alignment_STAR/*; do
            if [ "${i}" != "${i%.bam}" ];then
                echo ${i:15:10}
                samtools flagstat ${i}
                echo '############################################################'
            fi
        done
```

We can also inspect the final log file generated by STAR with the following command:

```
In [ ]: for i in alignment_STAR/*; do
            if [ "${i}" != "${i%.final.out}" ];then
                echo ${i:15:10}
                cat ${i}
                echo
                echo '############################################################'
                echo
            fi
        done
```

5. Read Quantification

5.1. Using STAR output

To compare the expression of single genes between different conditions, an essential step is the quantification of reads per gene. The most popular tools for gene quantification are `htseq-count` and `featureCounts`. `featureCounts` is provided by the `subread` package.

The `featureCounts` tool calls a hit if any overlap (1 bp or more) is found between the read and a feature and provides the option to either exclude multi-overlap reads or to count them for each feature that is overlapped.

The nature and lengths of the reads, gene expression quantification will be strongly affected by the underlying gene annotation file that is supplied to the quantification programs.

In the following command, `featureCounts` is used to count the number of reads overlapping with genes.

```
In [ ]: # Create a folder for read counts
        mkdir -p read_counts

In [ ]: # Count features using 10 threads ('-T 10')
        featureCounts -a annotation/gene_annotation.gtf \
                      -o read_counts/featureCounts_results.txt \
                         alignment_STAR/*bam -T 10
```

5.2. Using the `salmon` pseudoaligner

`salmon` is a probabilistic RNA-seq quantification program. It pseudoailigns reads to a reference sequence, producing a list of transcripts that are compatible with each read while avoiding alignment of individual bases.

The program circumvents the need for large alignment files, thus reducing storage needs while increasing speed and enabling the processing of large numbers of samples on modest computational resources. Pseudoaligners like `salmon` estimate **transcript-level counts (not gene-level counts)**.

5.2.1. Transcriptome Indexing

Let's now download the Homo sapiens transcriptome reference from NCBI. Transcriptome reference sequence file can also be downloaded from ENSEMBL.

```
In [ ]: wget -P reference ftp://ftp.ncbi.nlm.nih.gov/refseq/H_sapiens/annotation\
        /GRCh38_latest/refseq_identifiers/GRCh38_latest_rna.fna.gz
```

```
In [ ]: # Decompress transcriptome sequence file and rename it
        gzip -dk < reference/GRCh38_latest_rna.fna.gz > reference/human_transcriptome.fna
```

Next, let's build an index on our transcriptome (**this may take some time!**).

```
In [ ]: if [ ! -d salmon_index ] ; then
            salmon index -t reference/human_transcriptome.fna -i salmon_index
        fi
```

5.2.2. Quantifying reads using `salmon`

Now that we have our index built, we are ready to quantify our samples using the following command (**this may take some time!**):

```
In [ ]: # Read quantification using 8 threads (-p 8)
        cut -f5 samples.txt | xargs -i sh -c \
        'v={}; salmon quant -i salmon_index -l A \
                 -1 fastq/${v}/${v}_1.fastq.gz \
                 -2 fastq/${v}/${v}_2.fastq.gz \
                 -p 8 -o quants/${v}_quant'
```

The read counts file `featureCounts_results.txt` can be imported in `R` for downstream data analysis. Similarly, we can import transcript-level estimates into `R` from the quantification files generated by `salmon`.

In another tutorial, we will show how to process `featureCounts_results.txt` and the transcript level files from `salmon` for Differential Gene Expression Analysis using `DESeq2`, `edgeR`, or `limma-voom` in `R`.