**STEP:1**

**(Provided excel sheet & details)**

* We were provided with an excel sheet named HNSCC\_normals\_20230529.xlsx by Dr. Pratik Chandrani, containing data of 65 studies of HNSCC normal samples. The sheet contained 4 columns (study\_id, sample\_id, raw\_SRA, study\_title).
* We found more 6 samples (now the total number of samples is 71), made a new excel sheet (named: HNSCC\_*date*).

**STEP:2**

**(SRA, Ids and representation)**

SRA (Sequence Read Archive) is a publicly accessible repository under NCBI that stores raw high-throughput sequencing data from various genomic studies by providing a vast collection of genetic information that can be used for a wide range of analyses and investigations. It enables the retrieval and analysis of sequencing data, allowing researchers to study genetic variations, gene expression patterns, and other genomic features, aiding in the advancement of fields like genomics, genetics, and molecular biology.

SRP, SRX, SRS, and SRR are identifiers used within the Sequence Read Archive (SRA) database for different components of a sequencing study.

1. **SRP** (Project/Study): SRP is the Study identifier in SRA, representing a sequencing project as a whole. It encompasses multiple samples, experiments, and datasets associated with a specific research study. For example, SRP123456 could represent a study investigating gene expression changes in cancer cells.

2. **SRX** (Experiment): SRX is the Experiment identifier in SRA, representing a specific sequencing experiment within a study. It corresponds to a particular sample or library preparation and is associated with one or more sequencing runs. For instance, SRX789012 could represent an experiment studying the transcriptome of liver tissue under different treatment conditions.

3. **SRS** (Sample): SRS is the Sample identifier in SRA, representing an individual biological sample used in a sequencing study. It describes the characteristics of the sample, such as its organism, tissue type, and experimental conditions. For example, SRS345678 could represent a sample of human blood collected from healthy individuals.

4. **SRR** (Run): SRR is the Run identifier in SRA, representing a specific sequencing run or set of raw sequencing data files associated with an experiment. It contains the actual sequence reads and quality information generated from a sample. For instance, SRR9012345 could represent a single sequencing run producing raw data files for a specific experiment.

Together, these identifiers (SRP, SRX, SRS, and SRR) help organize and link different components of a sequencing study within the SRA database, enabling to access and analyze specific studies, experiments, samples, and sequencing runs of interest.

**STEP:3**

**(SRA toolkit: downloading tool)**

SRA-toolkit was used to download SRA files via MobaXterm (toolbox for remote computing).

SRA Toolkit is a collection of command-line tools used in bioinformatics for working with sequencing data from the Sequence Read Archive (SRA). It allows users to retrieve, convert, and manipulate raw sequencing files, perform quality control checks, and integrate with other bioinformatics tools. SRA Toolkit provides an essential resource for processing and analyzing SRA data efficiently.

**STEP:4**

**(Script files and details)**

The main excel shit was split into 4 sub sheets according to the different SRP ids and put to download the SRR separately.

Path for accessing the SRP: cd /home/pclab2/Elveera/SRA\_Download/

cd to the corresponding SRP

vi SRA\_download.py

|  |
| --- |
| **The script for downloading SRR ids:**  import pandas as pd  import subprocess  SRS\_list = pd.read\_csv("SRP078156\_sra\_result.csv", sep = ",")  #print(SRS\_list.head(10))  for idx in SRS\_list.index:  #print(idx)  SRR\_Value = SRS\_list['Sample Accession'][idx]  subprocess.call(['prefetch', str(SRR\_Value)]) |

The script was run using the command - nohup python3 script file > log file &

The download status was checked using: ls -lrth (list the files with size)

To view log file, we used "less" command and the files which were completely downloaded, showed “successfully downloaded” status

Once download was done, "done" was entered in the adjacent column of the Read\_id in the file.

**STEP:5**

**(Updating the excel sheet post download)**

Once, all the SRR were downloaded and the corresponding files checked, the excel sheet was updated. The sheet contained 71 rows of samples and 25 columns (Study\_Acc, Experiment\_Acc, Sample\_Acc, Read\_Acc , PM\_ID, Download\_Status, Experiment\_Title, Organism\_Name, Instrument, Submitter, Study\_Title, Sample\_Title, Sex, Total\_Size, Total\_RUNs, Total\_Spots, Total\_Bases, Library\_Name, Library\_Strategy, Library\_Source, Library\_Selection, Disease\_remark, Ethnic\_origin\_of\_individual(literature based), Read\_number, fastq\_quality).

**STEP:6**

**(Conversion of sralite files to fastq files)**

A new directory (ALL\_SRR\_20230619) was made in SRA\_download and all the previously downloaded SRR files were copied to this file and fastq script was written to run in the particular directory

vi Fasterq-dump\_20230621.sh was scripted

|  |
| --- |
| **The script for fastq:**  for file in SRR\*/\*.sralite  do  echo "$file"  fasterq-dump --split3 "${file}"  done |

nohup bash Fasterq-dump\_20230621.sh > Fasterq-dump\_20230621.log &

All the sralite files were converted into fastq files. For each SRR id, there was two fastq files, consisting of read1(forward read) as SRR\*\*\*\*\*\*\_1.fastq & read2(reverse read) as SRR\*\*\*\*\*\*\*\_2.fastq

**STEP:7**

**(fastq to fastqc)**

A new directory (FastQC\_res\_20230626) was made in ALL\_SRR\_20230619 for saving the fastqc files and fastqc script was written to run in the particular directory

|  |
| --- |
| for R1 in \*\_1.fastq  do  # Extract the SRR ID from the filename  name=$(basename ${R1} \_1.fastq)  R2=${name}\_2.fastq  #echo "$srr\_id"  #echo $R1  # Run FastQC on the R1 and R2 FastQ files  fastqc -t 8 ${R1} ${R2} -o FastQC\_res\_20230626  done |

nohup bash fastqc.sh > FastQC\_res\_20230626.log &

All the fastq files were converted into fastqc files and each saved in html and zip format. Each html formatted files were opened using chrome and looked for quality check results.

Some files had poor QC values, they were further put for trimming.

**STEP:8**

**(Trimming of bad QC using TRIMMOMATIC and then performing fastqc on those files)**

Trimmomatic tool was downloaded for trimming off the parts of bad quality read and then fastqc was run on the trimmed files and was saved in Trimmomatic\_res\_20230628/FastQCTrim

|  |
| --- |
| for R1 in \*\_1.fastq  do  # Extract the SRR ID from the filename  name=$(basename ${R1} \_1.fastq)  R2=${name}\_2.fastq  #echo "$srr\_id"  #echo $R1  # Run FastQC on the R1 and R2 FastQ files  # fastqc -t 8 ${R1} ${R2} -o FastQC\_res\_20230626  java -jar /home/pclab2/tools/Trimmomatic-0.39/trimmomatic-0.39.jar PE -threads 8 -phred33 ${R1} ${R2} Trimmomatic\_res\_20230628/${name}\_1\_paired.fq Trimmomatic\_res\_20230628/${name}\_1\_unpaired.fq Trimmomatic\_res\_20230628/${name}\_2\_paired.fq Trimmomatic\_res\_20230628/${name}\_2\_unpaired.fq ILLUMINACLIP:/home/pclab2/tools/Trimmomatic-0.39/adapters/TruSeq3-PE-2.fa:2:30:10:2:True SLIDINGWINDOW:4:20 MINLEN:50  fastqc Trimmomatic\_res\_20230628/${name}\_1\_paired.fq Trimmomatic\_res\_20230628/${name}\_2\_paired.fq -o Trimmomatic\_res\_20230628/FastQCTrim  done |

**STEP:9**

**(STAR Alignment)**

STAR(Spliced Transcripts Alignment to a Reference) is used to align RNA-Seq reads to a reference genome, allowing to accurately map and quantify gene expression levels in different tissues. General overview:

1. **Preprocessing**: Before aligning the RNA-Seq reads, certain preprocessing steps are typically performed, such as quality control, adapter trimming, and filtering of low-quality reads. These steps ensure that only high-quality reads are used for alignment.

2. **Genome** **Indexing**: STAR requires a pre-built index of the reference genome. This index is generated using the reference genome sequence and annotation files, which provide information about gene locations, splice sites, and other genomic features.

3. **Alignment**: The RNA-Seq reads are aligned to the reference genome using the STAR algorithm. STAR uses a two-step process that *involves aligning reads to potential exon-exon junctions and then extending alignments across the entire transcript sequence*. This approach enables accurate alignment, especially for spliced transcripts.

4. **Quantification**: Once the reads are aligned, STAR assigns them to specific genes and transcripts, allowing for the quantification of gene expression levels. This information can be used to analyze gene expression patterns across different tissues in GTEx studies.

5. **Post-processing**: After alignment and quantification, additional quality control steps may be performed to filter out potential artifacts or low-quality data. These steps help ensure the reliability of the gene expression measurements.

|  |
| --- |
| #!/bin/bash  ##run by bsub -Is -n 26 -m cn1 -J rna rna\_processing.sh -o batch4b.log -e batch4b.err  NR\_CPUS=3  job\_count=0  for R1 in \*\_1\_paired.fq.gz  do  R2=${R1//\_1\_paired.fastq.gz/\_2\_paired.fastq.gz}  echo "Star alignment for $R1 $R2"  sample="${R1%%\_1\*}"  echo ${sample}  /scratch/pratik/tools/STAR-2.7.9a/bin/Linux\_x86\_64\_static/STAR --runThreadN 8 --genomeDir /scratch/pratik/ref/GRCh38\_gencode\_v37\_CTAT\_lib\_Mar012021.plug-n-play/ctat\_genome\_lib\_build\_dir/ref\_genome.fa.star.idx/ --readFilesIn ${R1} ${R2} --readFilesCommand gunzip -c --outReadsUnmapped None --quantMode GeneCounts --twopassMode Basic --outSAMstrandField intronMotif --outSAMtype BAM SortedByCoordinate --outSAMunmapped Within --chimSegmentMin 12 --chimJunctionOverhangMin 8 --chimOutJunctionFormat 1 --alignSJDBoverhangMin 10 --alignMatesGapMax 100000 --alignIntronMax 100000 --alignSJstitchMismatchNmax 5 -1 5 5 --outSAMattrRGline ID:"$sample" SM:"$sample" CN:ILLUMINA PG:STAR --chimMultimapScoreRange 3 --chimScoreJunctionNonGTAG -4 --chimMultimapNmax 20 --chimNonchimScoreDropMin 10 --peOverlapNbasesMin 12 --peOverlapMMp 0.1 --alignInsertionFlush Right --alignSplicedMateMapLminOverLmate 0 --alignSplicedMateMapLmin 30 --outFileNamePrefix result\_files/${sample}  job\_count=$((job\_count+1))  if [ "$job\_count" -eq $NR\_CPUS ]; then  wait  job\_count=0  fi  done |

STAR alignment was used to align the downloaded data. For this, we connected with HPC and downloaded the six samples alternatively in two batches.

**STEP:10**

**(Fastp)**

It was observed that **trimmomatic** was not efficient in removing the *PolyA and PolyG* adapters, so Fastp was further used to remove the residual adapters, i.e., mainly *PolyA and PolyG.*

Fastp was run of all the fastq files which were not trimmed properly and a new directory named Fastp\_Trimmed was made in ALL\_SRR\_20230619 folder for saving the trimmed files. The trimmed files were further checked for quality and saved in directory named FastQCTrim and multiqc was run on all the fastqc files to get a compiled result of all the fastqc samples.

|  |
| --- |
| for R1 in \*\_1.fastq  do  # Extract the SRR ID from the filename  name=$(basename ${R1} \_1.fastq)  R2=${name}\_2.fastq  #echo "$srr\_id"  #echo $R1  # Run FastQC on the R1 and R2 FastQ files  # fastqc -t 8 ${R1} ${R2} -o FastQC\_res\_20230626  /home/pclab2/tools/fastp -i ${R1} -I ${R2} --thread 16 -q 20 --length\_required 50 --cut\_right --cut\_right\_window\_size 4 --cut\_right\_mean\_quality 20 --detect\_adapter\_for\_pe --low\_complexity\_filter --complexity\_threshold 60 --trim\_poly\_g --trim\_poly\_x -o Fastp\_Trimmed/${name}\_1\_paired.fastq -O Fastp\_Trimmed/${name}\_2\_paired.fastq --unpaired1 Fastp\_Trimmed/${name}\_1\_unpaired.fastq --unpaired2 Fastp\_Trimmed/${name}\_2\_unpaired.fastq --failed\_out Fastp\_Trimmed/${name}\_failed.txt -h Fastp\_Trimmed/${name}\_fastp.html  fastqc Fastp\_Trimmed/${name}\_1\_paired.fastq Fastp\_Trimmed/${name}\_2\_paired.fastq -o Fastp\_Trimmed/FastQCTrim  done |

**STEP:11**

**(SALMON)**

The trimmed files were further **aligned** and **quantified** using SALMON tool and the output was saved in /home/pclab2/Elveera/SRA\_Download/ALL\_SRR\_20230619/SALMON\_20230724. The script used for the alignment was:

|  |
| --- |
| #!/bin/bash  NR\_CPUS=3  job\_count=0  for f in /home/pclab2/Elveera/SRA\_Download/ALL\_SRR\_20230619/Fastp\_Trimmed/\*\_1\_paired.fastq.gz  do  echo "Start the process"  fq=${f##\*/}  sample=${fq//\_1\_paired.fastq.gz/}  echo ${sample}  /home/pclab2/ref/ref\_encode104\_salmon\_20210716/index\_new/index\_new/salmon-1.5.1\_linux\_x86\_64/bin/salmon quant -i /home/pclab2/ref/ref\_encode104\_salmon\_20210716/index\_new/index\_new/ \  -l A \  -1 ${f} -2 ${f//\_1\_paired.fastq.gz/\_2\_paired.fastq.gz} \  -o SALMON\_20230724/${sample}\_quant \  -p 24 \  --seqBias --gcBias --validateMappings > SALMON\_20230724/${sample}.log &  job\_count=$((job\_count+1))  if [ "$job\_count" -eq $NR\_CPUS ]; then  wait  job\_count=0  fi  echo "Process completed"  done |

Five files were created in the output, named: aux\_info, cmd\_info.json, lib\_format\_counts.json, libParams logs and quant.sf.

Amongst these, the quant.sf file contained the quantifies data, which is further used for creating the matrix.