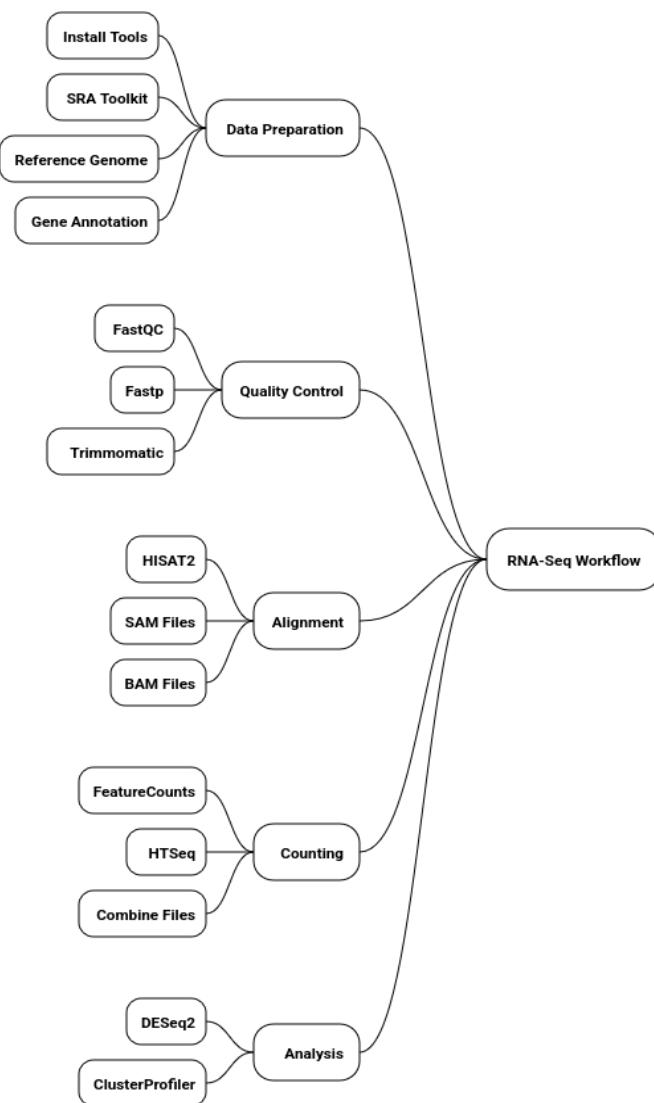


RNASeq Analysis – From Raw Data to Read Counts

RNA sequencing (RNA-Seq) is a powerful and widely used technique for studying the transcriptome, the complete set of RNA transcripts produced by the genome under specific conditions or in a specific cell type. Unlike traditional microarrays, RNA-Seq provides a high resolution, quantitative, and unbiased view of gene expression, enabling researchers to identify differentially expressed genes, discover novel transcripts, detect alternative splicing events, and quantify gene expression across the entire genome.

RNA-Seq Workflow: From Raw Reads to Functional Analysis



The RNA-Seq data analysis process can be broadly divided into two main stages: the transformation of raw sequencing reads into a count matrix, and the downstream statistical and functional interpretation of gene expression data.

It begins with quality control of raw sequencing reads in FASTQ format. Tools like FastQC are commonly used to evaluate sequencing quality, identify adapter contamination, and detect technical biases. If necessary, low-quality reads and adapter sequences are trimmed using programs such as Trimmomatic or Cutadapt. The cleaned reads are then aligned to a reference genome or transcriptome using aligners like STAR or HISAT2, or alternatively pseudoaligned using tools such as Salmon or Kallisto. Following alignment, gene- or transcript-level quantification is performed with tools like featureCounts or HTSeq, resulting in a count matrix that represents the number of reads mapped to each gene across all samples.

The downstream analysis of this count data to derive meaningful biological insights. Differential gene expression (DGE) analysis is typically performed using statistical frameworks such as DESeq2, edgeR, or limma to identify genes that show significant changes in expression between conditions or experimental groups. The results are often visualized using principal component analysis (PCA), heatmaps, and volcano plots to explore sample relationships and highlight key regulatory genes. To interpret the biological relevance of differentially expressed genes, functional enrichment analysis is carried out using gene ontology (GO) and pathway analysis tools such as clusterProfiler or GSEA. This integrative workflow allows researchers to move from raw sequencing data to a deeper understanding of cellular states, molecular functions, and regulatory mechanisms underlying the biological system under study.

Altogether, the RNA-Seq analysis workflow is a multi-step process that transforms raw sequencing data into biologically meaningful insights, enabling researchers to understand gene regulation, identify biomarkers, and study disease mechanisms at the transcriptomic level.

Setting Up

PART A: Creating conda Environment and Installation of tools

Before beginning RNA-seq data analysis, it's essential to establish a reproducible and isolated environment. This ensures that all dependencies are correctly installed and consistent across runs. The following steps set up a conda environment tailored for both preprocessing and downstream differential expression analysis.

Create Conda Environment (with R – required for DEseq2)

```
conda create -n bulk_env -c conda-forge -c bioconda r-base=4.3.3 r  
essentials
```

- `bulk_env` is the name of the conda environment.
- The environment includes R version 4.3.3, compatible with the DESeq2 package used later for differential expression analysis.
- The `r-essentials` package includes core R packages (e.g., `tidyverse`, `IRkernel`) commonly used in data analysis workflows.

Install tools for RNA-Seq Analysis

```
conda install -y -c bioconda fastqc multiqc trimmomatic hisat2 samtools  
subread
```

This command installs the core command-line tools necessary for RNA-seq data processing:

- **FastQC**: Quality control tool for raw sequencing reads.
- **Trimmomatic**: Adapter and quality trimming tool for FASTQ files.
- **HISAT2**: Fast and sensitive alignment tool for mapping reads to a reference genome.
- **SAMtools**: Suite of tools for manipulating alignments in the SAM/BAM format.
- **Subread**: Provides the `featureCounts` tool used for gene-level quantification.

PART B: Create Directories

To maintain a clean and organized workspace throughout the RNA-seq analysis, it's best practice to create a structured directory layout. This ensures that files generated at different stages of the pipeline are stored in dedicated folders, making the workflow easier to manage and debug.

```
mkdir BulkTranscriptomics  
cd BulkTranscriptomics  
mkdir FASTQ FASTQC_Results TRIMMED ALIGN BAM REFERENCE COUNTS
```

OR

```
mkdir -p  
BulkTranscriptomics/{FASTQ,FASTQC_Results,TRIMMED,ALIGN,BAM,REFERE  
NC E,COUNTS}
```

These commands create subdirectories within your project folder (`BulkTranscriptomics`) for different stages of the analysis:

- **FASTQ**: Raw sequencing data (downloaded FASTQ files).
- **FASTQC_Results**: Output from FastQC quality control checks.
- **TRIMMED**: Adapter- and quality-trimmed FASTQ files.

- ALIGN: Intermediate alignment files
- BAM: Final sorted BAM files after alignment.
- REFERENCE: Reference genome files and annotation files
- COUNTS: Read count files generated by featureCounts.

Export Directory Paths as Environment Variables

Path exporting improves efficiency, clarity, and flexibility in bioinformatics workflows, especially when working with complex pipelines like RNA-seq.

```
export FASTQ=/home/tundup/myAnalysis/BulkTranscriptomics/FASTQ
export
FASTQC_Results=/home/tundup/myAnalysis/BulkTranscriptomics/FASTQC_Results
export TRIMMED=/home/tundup/myAnalysis/BulkTranscriptomics/TRIMMED
export ALIGN=/home/tundup/myAnalysis/BulkTranscriptomics/ALIGN
export BAM=/home/tundup/myAnalysis/BulkTranscriptomics/BAM export
REFERENCE=/home/tundup/myAnalysis/BulkTranscriptomics/REFERENCE
export COUNTS=/home/tundup/myAnalysis/BulkTranscriptomics/COUNTS
```

PART C: Download Datasets (from GEO):

To begin RNA-seq analysis using GEO datasets, raw sequencing data must first be downloaded from the NCBI Sequence Read Archive (SRA). This step uses the SRA Toolkit, specifically fasterq-dump, which is optimized for high-speed FASTQ file retrieval.

Install sra-tools

Installs the SRA Toolkit, a command-line suite for downloading and processing data from NCBI's SRA.

```
conda install -c bioconda sra-tools=3.0.7 (provided version number)
```

Downloading SRA Datasets with fasterq-dump

fasterq-dump is a faster alternative to fastq-dump, designed to efficiently retrieve FASTQ files. To retrieve a list of accession numbers (of raw data) from NCBI's SRA Run Selector, go to <https://www.ncbi.nlm.nih.gov/Traces/study/> and enter your SRA Study Accession Number (e.g., PRJNAxxxxxx or SRPxxxxxx) into the search bar. Once the associated sequencing runs are displayed, select the specific samples you are interested in by checking the corresponding boxes. After making your selection, click on the "Accession List" button located at the top-right side of the table. This will download a plain-text file containing the selected SRA Run Accessions (e.g., SRR numbers)

If You Have a Few SRR IDs:

```
fasterq-dump SRR11262284 SRR11262285 SRR11262286 SRR11262292 SRR11262293  
SRR11262294 --threads 10 --progress --split-files
```

- **--threads:** Number of threads for faster processing.
- **--progress:** Shows live progress.
- **--split-files:** Splits paired-end reads into _1.fastq and _2.fastq.

Using GNU parallel (faster downloads):

```
parallel -j 2 'fasterq-dump {} --threads 4 --split-files  
--progress -O $FASTQ' :: SRR11262284 SRR11262285 SRR11262286  
SRR11262292 SRR11262293 SRR11262294
```

- GNU parallel executes multiple downloads concurrently for improved speed.
- **-j 2:** Runs 2 downloads in parallel.
- Ensures efficient use of CPU and bandwidth.

Other Options:

Option A: If you have many SRR IDs (All in one line):

```
for srr in $(cat srr_list.txt); do fasterq-dump "$srr" --threads 6 --progress  
--split-files -O raw_fastq/; done
```

Option B: If there are many files – one SRR ID in one line:

```
while read srr; do fasterq-dump "$srr" --threads 6 --progress --split-files  
-O raw_fastq/; done < srr_list.txt
```

- These loops automate downloading when SRR IDs are saved in a file (srr_list.txt).
Output directory is specified using -O.

PART D: Downloading Reference Genome and Gene Annotation Files:

To perform read alignment and gene-level quantification, you'll need a reference genome file and its corresponding gene annotation file (GTF). These files should match the version used in downstream tools to ensure accurate mapping and counting.

Reference Genome (FASTA)

The grch38_genome.tar.gz file is a pre-built HISAT2 reference genome index for the human GRCh38 assembly, used for aligning RNA-Seq reads in transcriptomics analysis. It contains .ht2 index files required by HISAT2 for fast and accurate alignment. It can be downloaded from: <https://daehwankimlab.github.io/hisat2/download/#h-sapiens>

```
wget -P $REFERENCE https://genome  
idx.s3.amazonaws.com/hisat/grch38_genome.tar.gz
```

The grch38_genome.tar.gz file contains eight HISAT2 index files. These .ht2 files are binary index files generated from the GRCh38 reference genome and are required by HISAT2 to perform sequence alignment. They represent different parts of the indexed genome, split for efficient loading and access during alignment.

Unzip the downloaded gzipped tar file:

```
tar -xzf grch38_genome.tar.gz
```

Gene Annotation File (GTF)

Download the gene annotation file:

```
wget -P $REFERENCE https://ftp.ensembl.org/pub/release  
109/gtf/homo_sapiens/Homo_sapiens.GRCh38.109.gtf.gz
```

Unzip the gz file:

```
gunzip Homo_sapiens.GRCh38.109.gtf.gz
```

This GTF file includes detailed gene structure annotations (e.g., exons, transcripts, gene IDs, gene symbols). It's required by tools like featureCounts for accurate gene-level read summarization.

STEP 01: QUALITY CONTROL (FastQC)

Quality control (QC) is the first and critical step in RNA-seq data analysis. It checks for base quality scores, GC content, adapter contamination, and other issues in raw sequencing reads. This step uses **FastQC** for individual report generation and **MultiQC** to combine them into one summary.

Run FastQC on All FASTQ Files

This command runs FastQC on all paired-end FASTQ files for both experimental groups (LUNG_CA and LUNG_NOR) and stores the results in a designated output folder.

```
fastqc $FASTQ/LUNG_CA/*_1.fastq $FASTQ/LUNG_CA/*_2.fastq  
$FASTQ/LUNG_NOR/*_1.fastq $FASTQ/LUNG_NOR/*_2.fastq -o $FASTQC_Results -t  
10
```

- **fastqc**: The tool used for quality control checks.
- **\$FASTQ/LUNG_CA/*_1.fastq and \$FASTQ/LUNG_CA/*_2.fastq**: Read 1 and Read 2 FASTQ files from lung cancer samples.
- **\$FASTQ/LUNG_NOR/*_1.fastq and \$FASTQ/LUNG_NOR/*_2.fastq**: Read 1

and Read 2 FASTQ files from normal lung tissue.

- `-o $FASTQC_Results`: Specifies the output directory to store FastQC reports.
- `-t 10`: Utilizes 10 threads to speed up processing.

This will generate zip and html file for each SRR files.

Generate a Summary Report with MultiQC

Scans the FastQC output folder and generates a unified report summarizing quality across all samples.

```
multiqc $FASTQC_Results -o $FASTQC_Results
```

- `multiqc`: Runs the tool.
- `$FASTQC_Results`: The folder containing FastQC output files.
- `-o $FASTQC_Results`: Tells MultiQC to save the summary report in the same location.

OUTPUT: A folder called `multiqc_data` and a `multiqc_report.html` file, a comprehensive quality control report.

View the Report on Your Local Machine

Copies the `multiqc_report.html` file from the remote HPC server to your local system, allowing you to open it in a browser.

```
scp username@your_hpc_address:/path/to/multiqc_report.html  
/path/to/destination_folder
```

- `scp`: Secure copy command used to transfer files over SSH.
- `username@your_hpc_address`: Your username and IP address of the remote HPC machine (if you are processing on HPC).
- `/path/to/multiqc_report.html`: Full path to the HTML report on the HPC.
- `/path/to/destination_folder`: Destination path on your **local Windows** machine.

After downloading, simply double-click the `.html` file to open it in your browser and inspect sample quality.

STEP 02: TRIMMING (Trimmomatic)

Trimmomatic is a widely used software tool for trimming and filtering sequencing reads in RNA-seq analysis. Raw reads often contain low-quality bases, adapter sequences and excessively short reads from raw sequencing data that can interfere with accurate alignment and downstream analysis. By trimming these problematic regions, Trimmomatic helps ensure that only high-quality, informative reads are used, thereby improving the reliability and

accuracy of RNA-seq results.

```
trimmomatic PE input_1.fastq input_2.fastq out_1_paired.fq  
out_1_unpaired.fq out_2_paired.fq out_2_unpaired.fq SLIDINGWINDOW:4:20  
MINLEN:36
```

- **trimmomatic**: Calls the Trimmomatic program.
- **PE**: Specifies that the data is paired-end (two input files, one for each read direction).
- **input_1.fastq**: The first input FASTQ file (forward reads).
- **input_2.fastq**: The second input FASTQ file (reverse reads).
- **out_1_paired.fq**: Output file for surviving paired reads from the first input (both mates passed filters).
- **out_1_unpaired.fq**: Output file for reads from the first input whose mate was discarded (unpaired).
- **out_2_paired.fq**: Output file for surviving paired reads from the second input.
- **out_2_unpaired.fq**: Output file for reads from the second input whose mate was discarded.
- **SLIDINGWINDOW:4:20**: Applies a sliding window of 4 bases; if the average quality within the window drops below 20, the read is trimmed at that point.
- **MINLEN:36**: Discards any reads that are shorter than 36 bases after trimming.

STEP 03: ALIGNMENT TO REFERENCE GENOME (HISAT2)

HISAT2 (Hierarchical Indexing for Spliced Alignment of Transcripts) is a fast, memory-efficient aligner designed for RNA-Seq data. It excels at handling spliced alignments and identifying splice junctions, making it ideal for transcriptome analysis. HISAT2 uses a hierarchical indexing strategy combining a global FM index with thousands of local indexes, allowing rapid and accurate mapping of reads across large genomes like human, even those spanning exons or containing variants.

```
hisat2 -p 20 -x $REFERENCE/grch38/genome -1  
$FASTQ/LUNG_CA/SRR11262292_1.fastq -2 $FASTQ/LUNG_CA/SRR11262292_2.fastq -S  
$ALIGN/SRR11262292_aligned.sam
```

Similarly for each SRR pair files

- **hisat2**: Calls the HISAT2 aligner program.
- **-p 20**: Uses 20 CPU threads in parallel to speed up the alignment process.
- **-x \$REFERENCE/grch38/genome**: Specifies the basename of the indexed reference genome to align against. Here, \$REFERENCE/grch38/genome points to the reference genome index files for GRCh38.
- **-1 \$FASTQ/LUNG_CA/SRR11262292_1.fastq**: Input file for the first (forward) reads of paired-end sequencing. The \$FASTQ/LUNG_CA/SRR11262292_1.fastq is a shell variable pointing to the actual file path.
- **-2 \$FASTQ/LUNG_CA/SRR11262292_2.fastq**: Input file for the second (reverse) reads of paired-end sequencing.
- **-S \$ALIGN/SRR11262292_aligned.sam**: Output SAM file where the aligned reads will be written. \$ALIGN/SRR11262292_aligned.sam is a variable for the output directory and file

name.

With loop (one-liner for all files)

```
for fq1 in $FASTQ/LUNG_CA/*_1.fastq; do fq2=${fq1/_1.fastq/_2.fastq};  
base=$(basename $fq1 _1.fastq); hisat2 -p 20 -x $REFERENCE/grch38/genome -1  
$fq1 -2 $fq2 -S $ALIGN/${base}_aligned.sam; done  
  
for fq1 in $FASTQ/LUNG_NOR/*_1.fastq; do fq2=${fq1/_1.fastq/_2.fastq};  
base=$(basename $fq1 _1.fastq); hisat2 -p 20 -x $REFERENCE/grch38/genome -1  
$fq1 -2 $fq2 -S $ALIGN/${base}_aligned.sam; done
```

- `for fq1 in $FASTQ/LUNG_NOR/*_1.fastq; do ... done`: This is a Bash loop. It iterates over all files in the directory `$FASTQ/LUNG_NOR/` that end with `_1.fastq` (i.e., all forward read files for paired-end samples).
- `fq2=${fq1/_1.fastq/_2.fastq}`: For each `fq1` (forward read file), this line creates the corresponding reverse read file name (`fq2`) by replacing `_1.fastq` with `_2.fastq` in the filename.
- `base=$(basename $fq1 _1.fastq)`: Extracts the base name of the sample (without the `_1.fastq` suffix) from the forward read file. This base name is used to name the output SAM file.
 - `hisat2 -p 20 -x $REFERENCE/grch38/genome -1 $fq1 -2 $fq2 -S $ALIGN/${base}_aligned.sam`: Runs HISAT2 for each sample pair:
 - `-p 20`: Uses 20 CPU threads.
 - `-x $REFERENCE/grch38/genome`: Reference genome index.
 - `-1 $fq1`: Forward reads file.
 - `-2 $fq2`: Reverse reads file.
 - `-S $ALIGN/${base}_aligned.sam`: Output SAM file, named for the sample.

The primary output of HISAT2 is an alignment file in SAM (Sequence Alignment/Map) format, which contains detailed information about where each sequencing read maps to the reference genome. This SAM file includes alignment coordinates, mapping quality, CIGAR strings, and various optional tags for each read.

STEP 04: SAM TO BAM CONVERSION (Samtools)

SAMtools is a widely used toolkit for processing sequence alignment data. Converting a SAM (Sequence Alignment/Map) file to BAM (Binary Alignment/Map) format is a common step to reduce file size and improve processing speed. First human-readable SAM file is converted into a compressed, binary BAM file. The BAM file is then sorted by genomic coordinates, which is required for many downstream tools. Finally, an index is generated, creating a .bai file that allows rapid access to specific regions within the BAM file.

Convert, sort, and generate index file (for a single sample)

```
samtools view -@ 20 -bS $ALIGN/SRR11262284_aligned.sam >  
$BAM/SRR11262284.bam
```

```
samtools sort -@ 20 $BAM/SRR11262284.bam -o $BAM/SRR11262284_sorted.bam
```

```
 samtools index $BAM/SRR11262284_sorted.bam
```

Do same for the remaining files as well

- **samtools view:** Runs the samtools utility to view or convert alignment files.
- **-@ 20:** Uses 20 CPU threads for faster processing.
- **-bS:** -b: Output in BAM (binary) format; -S: Input is in SAM (text) format. ▪
\$ALIGN/SRR11262284_aligned.sam: Input SAM file with alignments. □
\$BAM/SRR11262284.bam: Redirects the output to a new BAM file in the \$BAM directory.
- **samtools sort:** Sorts the BAM file by genomic coordinates.
- **-@ 20:** Uses 20 CPU threads.
- **\$BAM/SRR11262284.bam:** Input BAM file to be sorted.
- **-o \$BAM/SRR11262284_sorted.bam:** Output file for the sorted BAM.
- **samtools index:** Creates an index file (.bai) for the sorted BAM. ▪
\$BAM/SRR11262284_sorted.bam: Input sorted BAM file to be indexed.

One-liner command – convert, sort, index file generation (for a single sample)

```
 samtools view -@ 20 -bS $ALIGN/SRR11262284_aligned.sam | samtools sort -@ 20 -o $BAM/SRR11262284_sorted.bam && samtools index $BAM/SRR11262284_sorted.bam
```

Do same for the remaining files as well

Loop for all .sam files present in \$ALIGN (One-liner command)

```
 for sam in $ALIGN/*_aligned.sam; do base=$(basename $sam _aligned.sam);  
 samtools view -@ 20 -bS $sam | samtools sort -@ 20 -o  
 $BAM/${base}_sorted.bam && samtools index $BAM/${base}_sorted.bam; done
```

To view bam file and sorted bam file:

```
 samtools view SRR11262284.bam | head  
  
 samtools view SRR11262284_sorted.bam | head
```

STEP 05: READS COUNTING (FeatureCounts)

featureCounts is a fast and accurate tool used to assign aligned RNA-Seq reads to genomic features such as genes or exons. It takes sorted BAM files and a gene annotation file (e.g., GTF or GFF) as input, and outputs a count matrix representing the number of reads mapped to each gene. This count data is essential for downstream differential gene expression analysis.

```
 featureCounts -T 20 -p -a $REFERENCE/Homo_sapiens.GRCh38.109.gtf -o  
 $COUNTS/SRR11262284_counts.txt $BAM/SRR11262284_sorted.bam
```

Do same for the remaining files as well

- **featureCounts:** The command-line tool from the Subread package used for assigning aligned reads to genomic features (e.g., genes, exons).
- **-T 20:** Use 20 CPU threads to speed up processing.
- **-p:** Enable paired-end mode, meaning reads from the same fragment (R1 and R2) are counted as a single unit.
- **-a \$REFERENCE/Homo_sapiens.GRCh38.109.gtf:** The GTF annotation file specifying gene structures (e.g., exons, gene_id). This file tells featureCounts where to count reads.
- **-o \$COUNTS/SRR11262284_counts.txt:** The output file where the count matrix for this sample will be saved. Each row = gene; last column = number of reads assigned to that gene.
- **\$BAM/SRR11262284_sorted.bam:** The input BAM file containing aligned RNA Seq reads (sorted by coordinate), to be counted.

Loop for all .bam files present in \$BAM (One-liner command)

```
for f in $BAM/*_sorted.bam; do featureCounts -T 20 -p -a  
$REFERENCE/Homo_sapiens.GRCh38.109.gtf -o $COUNTS/${(basename  
"${f%_sorted.bam}")}_counts.txt "$f"; done
```

NOTE: featureCounts outputs raw integer counts i.e. the number of sequencing reads assigned to each gene or feature in each sample. These raw counts reflect the direct output of the quantification step and are not adjusted for differences in sequencing depth, gene length, or other technical factors.

FINAL: Merging Count Files into a Single File

(This step can be also done with R or Python)

After running featureCounts on multiple samples, it is necessary to combine the resulting individual count files into a single count matrix before performing DESeq2 analysis. This is because DESeq2 requires a matrix where each row represents a gene and each column represents a sample, with the entries being the raw read counts. Combining the files ensures that all gene counts are aligned across samples, facilitating accurate normalization, statistical testing, and downstream interpretation. Without this unified matrix, it would be impossible to consistently compare gene expression across samples or conditions in DESeq2

```
mkdir MERGED_COUNTS  
  
export  
MERGED_COUNTS=/home/tundup/myAnalysis/BulkTranscriptomics/COUNTS/MERGED_CO  
NTS
```

1. Extract Gene IDs (from any one file)

```
awk 'NR>1 {print $1}' "$COUNTS/SRR11262284_counts.txt" >  
"$MERGED_COUNTS/geneids.txt"
```

2. Extract count columns from each file

```
for f in "$COUNTS"/*_counts.txt; do  
    sample=$(basename "$f" _counts.txt)  
    awk -v sample="$sample" 'NR==1 {next} {print $7}' "$f" >  
"$MERGED_COUNTS/${sample}_col.txt"  
done
```

3. Merge gene IDs and all count columns

```
paste "$MERGED_COUNTS/geneids.txt" "$MERGED_COUNTS"/*_col.txt >  
"$MERGED_COUNTS/merged_counts_body.txt"
```

4. Renaming Column Names (Full Path as Column name to Filename only)

View the file and column names:

```
head $MERGED_COUNTS/merged_counts_body.txt
```

Rename Column Names to last SRR IDs only:

```
sed '1s/[^[:space:]]*\///g; 1s/_sorted\.bam//g' merged_counts_body.txt >  
merged_counts_clean.txt
```

Confirm the renamed column names:

```
head $MERGED_COUNTS/merged_counts_clean.txt
```