Chapter 5 - Bulk RNA-seq Analysis

Transcript Quantification Using Salmon

Bulk RNA-seq data were processed using Salmon (v1.9.0) to quantify transcript-level read counts. The workflow comprised two main stages: (1) construction of a transcriptome index and (2) sample-specific quantification.

Reference Transcriptome and Index Construction

The GENCODE v46 human transcriptome (GRCh38/hg38 assembly) was used as the reference, providing comprehensive annotation of protein-coding genes, non-coding RNAs, and splice variants. To mitigate spurious alignments to non-transcriptomic regions (e.g., pseudogenes or genomic repeats), decoy sequences from the human genome were incorporated into the index. The transcriptome and decoy files were concatenated to generate a gentrome reference (`gencode.v46.gentrome.fa.gz`), and a Salmon index was built using the following parameters:

```
salmon index -t gencode.v46.gentrome.fa.gz -d gencodev46.decoys.txt -p 12 -i salmon_index --gencode
```

Salmon Quantification Parameters

For each sample, paired-end reads were quantified using Salmon in alignment-based validation mode to enhance mapping accuracy. The workflow included:

1. Library type: Automatically inferred (--libType A).
2. Mapping validation: Enabled (--validateMappings) to filter low-confidence alignments.
3. Abundance estimation: Expectation-Maximization (EM) algorithm (--useEM) for refined transcript abundance estimates.
4. Uncertainty estimation: 10 bootstrap iterations (--numBootstraps 10) to quantify technical variability.
5. Parallelization: 12 CPU threads to accelerate processing.

Example command:

```
salmon quant -i salmon_index -1 $R1 -2 $R2 --validateMappings --threads 12 -o $output_dir --useEM --numBootstraps 10
```

Batch Processing in HPC

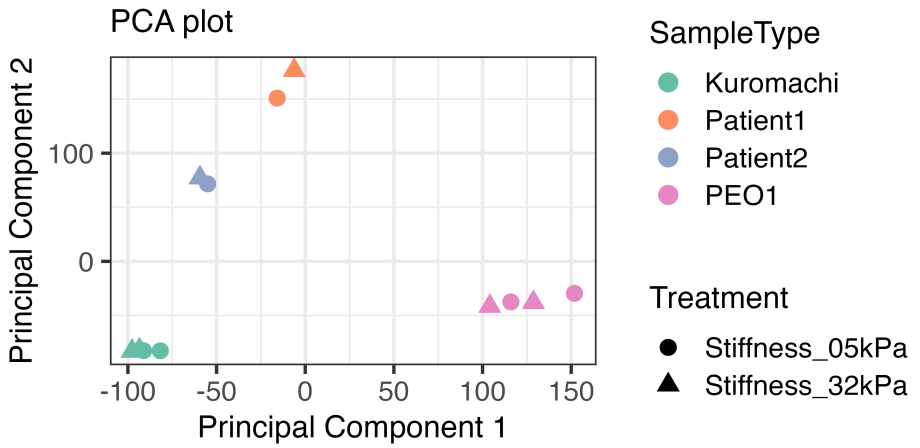
Samples were processed iteratively using a batch script on a high-performance computing cluster. Each sample's paired-end FASTQ files were stored in sample-specific directories, and Salmon outputs (including quantitated counts in `quant.sf` and bootstrapped estimates) were written to dedicated result folders. Computational resources were allocated as follows: 12 CPU cores, 50 GB memory, and 60-hour walltime per sample.

Differential Analysis by DESeq2

Differential expression analysis were conducted in R (v4.4.2) using the following packages: `tximport` and `DESeq2`.

Part 1: Stiffness-Dependent Gene Expression

Two cell lines (Kuromachi, PEO1) and two patient derived plates (Patient 1 and Patient 2) were cultured under two stiffness conditions, mimicked by pressure (0.5 kPa and 32 kPa), with two biological replicates per condition. Initial exploratory PCA based on variance stabilising transformed (`vst`) expression revealed strong batch effects between cell lines, necessitating cell line-specific analyses.



Sample metadata and salmon quantification files (`quant.sf`) were imported, with salmon results (raw counts) mapped to transcript-to-gene annotations curated from GENCODE v46 gtf file using `tximport` to generate gene-level counts. Since initial exploratory analysis demonstrated strong inter-cell-line batch effects, we conducted the differential expression analysis separately within each cell line. DESeq2 object was created with the design formula `~Treatment`, low expression genes were filtered (regained if ≥10 counts in ≥3 samples). Differential expression were tested using Wald tests with the contrast `Treatment_Stiffness_32kPa_vs_Stiffness_05kPa`. Log2 fold change were shrunk using the `apeglm` method to increase visibility and reduce noise. The two patient plates were treated as one sample type during the analysis, such that the final analysis output is denoted as DGE for PEO1, Kuromachi, and Patient. Multiple testing correction was conducted by false discovery rate (default by the name `BH` in the documentation, from the Benjamini & Hochberg 1995 paper). Pre-ranked gene set enrichment analysis was conducted based on the differential gene expression analysis ranked by their log2FC. Gene ontology biological process, hallmarks, and KEGG database were queried.

Part 2: YAP1 Knockout and Drug Treatment Interaction

A 2×2 factorial design evaluated YAP1 knockout (wild-type [wt] vs. knockout [ko]) and drug treatment (control vs. treatment), yielding four experimental groups. The analysis modelled additive effects, hypothesising that gene expression would follow a gradient: *YAP1wt + drug > YAP1wt + control > YAP1ko + drug > YAP1ko + control*. Accordingly, a

pseudo-linear variable (`Treatment.1`) encoded the ordinal combination of knockout and drug effects were created.
Similar preprocessing was applied to this dataset, including importation using `tximport` and pre-filtering before differential gene expression analysis. The differential expression analysis was then performed based on the pseudo-linear variable. In addition, sub-group analysis was conducted to compare drug effect (`Treatment_YAP1ko_treatment_vs_YAP1ko_wt`) and knockout effect (`Treatment_YAP1ko_wt_vs_YAP1wt_wt`). The output differential expression table was then imported into GSEA to perform pre-ranked gene set enrichment analysis.