Bulk RNA-Seq Analysis: Raw Reads to Differential Expression

Dr Suriya

2025-11-27

Contents

Complete RNA-Seq Analysis **3**

Project Overview . . . 3

Experimental Design . . . 3

Computational Environment . . . 3

Data Analysis Workflow . . . 3

1. Software Installation and Setup . . . 4

1.1 Conda Environment Setup (Terminal) . . . 4

Optional . . . 4

1.2 FastQC Installation and Setup . . . 4

1.3 MultiQC Setup . . . 4

1.4 Trim Galore Installation . . . 4

1.5 R Package Installation (Rstudio) . . . 5

1.6 Software Versions . . . 5

2. Data Processing (Terminal) . . . 6

2.1 Check Raw Files . . . 6

2.2 Download Reference Genome Preparation Files . . . 6

2.3 Decompress Reference Files . . . 6

3. Quality Control with FastQC and MultiQC . . . 7

3.1 Quality Control . . . 7

3.2 Trimming Adapters using Trim Glore . . . 7

4. Reference Genome Indexing (RStudio) . . . 9

4.1 Build Rsubread Index . . . 9

5. Sequence Alignment with Rsubread . . . 10

5.1 Prepare Read Files . . . 10

5.2 Perform Alignment . . . 10

6. Read Quantification with featureCounts . . . 11

6.1 Prepare BAM Files . . . 11
6.2 Run featureCounts . . . 11
6.3 Export Count Results . . . 11
7. Differential Expression Analysis with DESeq2 . . . 13
7.1 Data Preparation . . . 13
7.2 Create DESeq2 Object . . . 14
7.3 Factor Level Setting . . . 14
7.4 Differential Expression Analysis . . . 14
7.5 Export Differential Expression Results . . . 14
7.6 Add Gene Symbols . . . 15
8. Data Normalization and Visualization . . . 16
8.1 Variance Stabilizing Transformation . . . 16
8.2 Prepare Data for Visualization . . . 16
8.3 Volcano Plot . . . 16
8.4 Heatmap Visualization . . . 17
9 Session Information . . . 19
Output Files Generated . . . 20

Complete RNA-Seq Analysis

Project Overview

Experimental Design

- **Disease Model**: Breast cancer clinical biopsy samples
- **Sequencing Platform**: Illumina HiSeq 2500 platform
- **Sample Groups**:
  - Tumor samples (n=3)
  - Control samples (n=3)
- **Sequencing Type**: Bulk RNA-Seq, paired-end reads

Computational Environment

- **Quality Control & Preprocessing**: Conda environment on Linux system
- **Alignment & Differential Expression**: RStudio Server environment
- **Analysis Type**: End-to-end transcriptomic profiling

Data Analysis Workflow

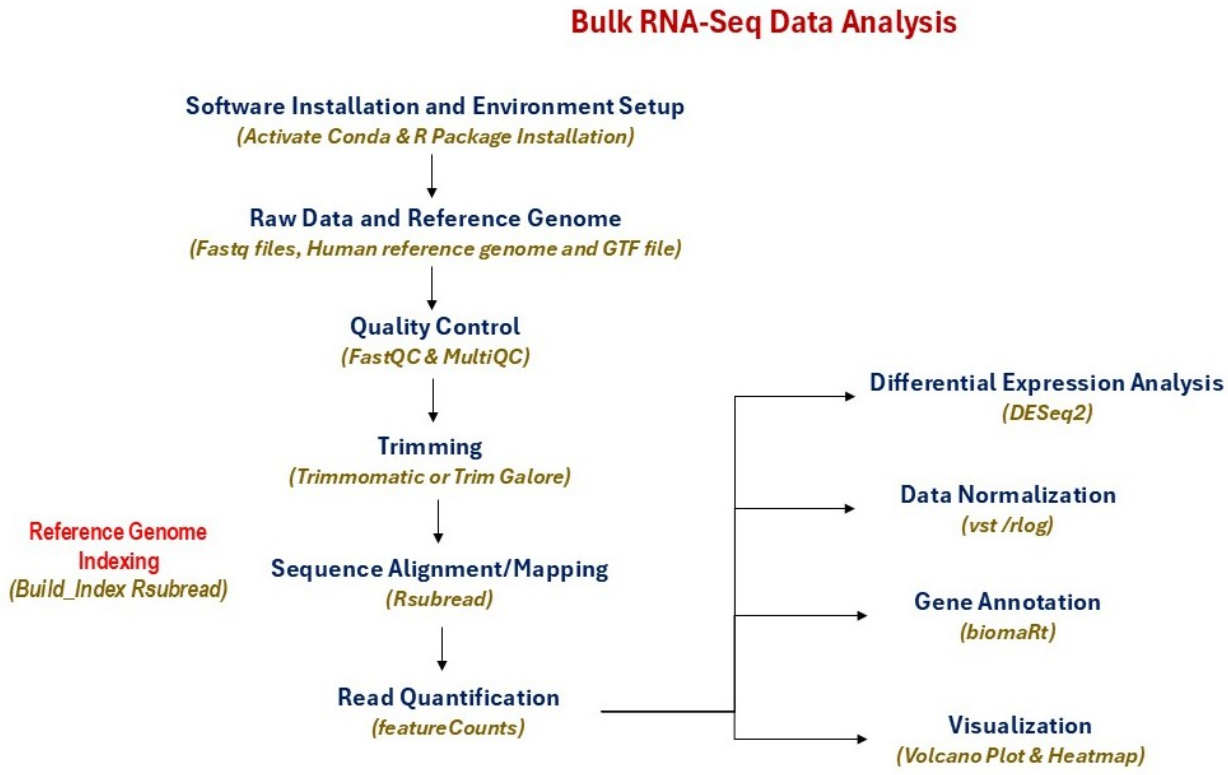

Figure 1: Data Analysis Workflow

1. Software Installation and Setup

1.1 Conda Environment Setup (Terminal)

```
# Create and activate conda environment
conda create -n ngs_analysis python=3.10 r-base=4.2.0
conda activate ngs_analysis

# Install core bioinformatics tools
conda install -c bioconda fastqc multiqc trim-galore cutadapt samtools subread
```

```
# Check command-line tool versions
fastqc --version
multiqc --version
trim_galore --version
cutadapt --version
```

Optional

1.2 FastQC Installation and Setup

```
# Download and setup FastQC
wget https://www.bioinformatics.babraham.ac.uk/projects/fastqc/fastqc_v0.12.1.zip
unzip fastqc_v0.12.1.zip
cd FastQC/
chmod 755 fastqc
```

1.3 MultiQC Setup

```
# Install MultiQC via conda
conda install -c bioconda multiqc
multiqc --version

# Alternative: Create dedicated environment
conda create -n multiqc_env python=3.10
conda activate multiqc_env
conda install -c bioconda multiqc
```

1.4 Trim Galore Installation

```
# Method 1: Direct download
wget https://github.com/FelixKrueger/TrimGalore/archive/refs/tags/0.6.10.zip
unzip 0.6.10.zip
cd TrimGalore-0.6.10
chmod +x trim_galore
sudo ln -s $(pwd)/trim_galore /usr/local/bin/
```

```
# Install cutadapt dependency
pip install cutadapt
cutadapt --version

# Method 2: Conda installation
conda install -c bioconda trim-galore
```

1.5 R Package Installation (Rstudio)

```
# Install Bioconductor packages
if (!require("BiocManager", quietly = TRUE))
    install.packages("BiocManager")

BiocManager::install(c(
    "Rsubread", "DESeq2", "biomaRt",  "pheatmap", "enrichplot", "dplyr",
    "ggplot2", "ggrepel", "forcats" ))
```

1.6 Software Versions

```
# Check installed versions
library(Rsubread)
packageVersion("Rsubread")

library(DESeq2)
packageVersion("DESeq2")
```

2. Data Processing (Terminal)

2.1 Check Raw Files

```
# Check the data
pwd
echo $'\n'; echo "======== listing files ========";  ls; echo $'\n'
```

```
(base) [vamouda@pelican3 Raw_Data]$ echo $'\n'; echo "======== listing files ========";  ls; echo $'\n'


======== listing files ========
JIPAI5_Tumour_R1.fastq.gz  JIPAI6_Tumour_R2.fastq.gz  JIPAI7_Tumour_R1.fastq.gz  JIPB53_Normal_R2.fastq.gz
JIPAI5_Tumour_R2.fastq.gz  JIPAI7_Normal_R1.fastq.gz  JIPAI7_Tumour_R2.fastq.gz  JIPB64_Normal_R1.fastq.gz
JIPAI6_Tumour_R1.fastq.gz  JIPAI7_Normal_R2.fastq.gz  JIPB53_Normal_R1.fastq.gz  JIPB64_Normal_R2.fastq.gz
```

Figure 2: Files Description

2.2 Download Reference Genome Preparation Files

```
# Create directory for reference files
mkdir -p ~/Suriya/Align/Ref_Genome
cd ~/Suriya/Align/Ref_Genome

# Download GTF annotation file
wget https://ftp.ensembl.org/pub/release-115/gtf/homo_sapiens/Homo_sapiens.GRCh38.115.gtf.gz

# Download reference genome (soft-masked primary assembly)
wget https://ftp.ensembl.org/pub/release-115/fasta/homo_sapiens/dna
                                        /Homo_sapiens.GRCh38.dna_sm.primary_assembly.fa.gz

# Verify downloads
ls -lh *.gz
```

2.3 Decompress Reference Files

```
# Decompress reference genome
gunzip -c Homo_sapiens.GRCh38.dna_sm.primary_assembly.fa.gz >
Homo_sapiens.GRCh38.dna_sm.primary_assembly.fa

# Decompress annotation file
gunzip -c Homo_sapiens.GRCh38.115.gtf.gz > Homo_sapiens.GRCh38.115.gtf

# Verify decompression
ls -lh Homo_sapiens.GRCh38*
```

Explanation: - **Soft-masked genome**: Lowercase letters indicate repetitive regions - **Primary assembly**: Contains main chromosomes without alternate haplotypes - **Release 115**: Specific Ensembl version for reproducibility

3. Quality Control with FastQC and MultiQC

3.1 Quality Control

```
# Navigate to data directory
cd /home/vamouda/Suriya/Data/

# Run FastQC on all FASTQ files
fastqc *.fastq.gz -o ../FastQC_Results/

# Run MultiQC to aggregate reports
multiqc ../FastQC_Results/ -o ../MultiQC_Report/
```

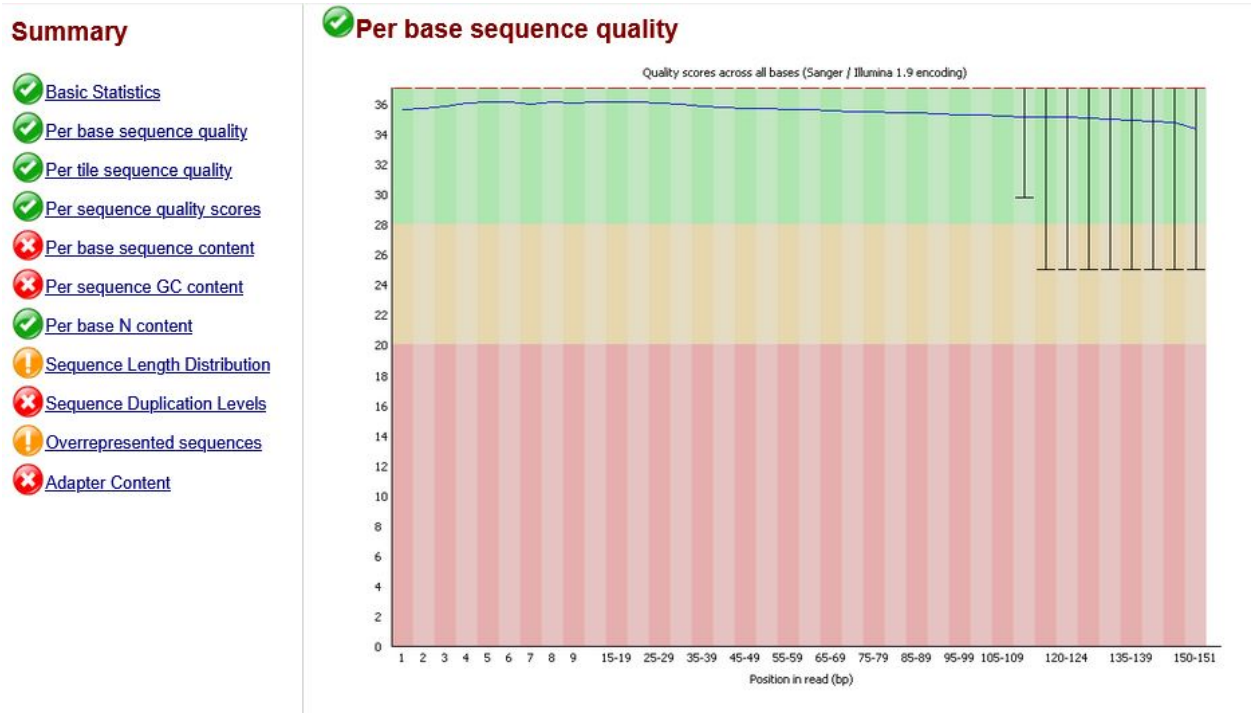

Figure 3: Quality Checking FastQC for Single Sequence

3.2 Trimming Adapters using Trim Glore

```
# Navigate to data directory
cd /home/vamouda/Suriya/Data/

# Batch trimming for all paired-end samples
find /home/vamouda/Suriya/Data/ -name "*_R1.fastq.gz" | while read r1; do
    r2="${r1/_R1.fastq.gz/_R2.fastq.gz}"
    if [[ -f "$r2" ]]; then
        echo "Processing: $(basename $r1) and $(basename $r2)"
        trim_galore --paired --quality 30 --fastqc "$r1" "$r2"
    else
```

```
        echo "Warning: Missing paired file for $r1"
    fi
done
```

Trim Galore Parameters: - `--paired`: Treat files as paired-end reads - `--quality 30`: Phred quality score cutoff (Q30) - `--fastqc`: Run FastQC on trimmed files

4. Reference Genome Indexing (RStudio)

4.1 Build Rsubread Index

```
# Set working directory
setwd("~/Suriya/Align")

# Load Rsubread library
library(Rsubread)

# Build genome index
buildindex(
  basename = "Ref_Genome_GRCh38",
  reference = "Homo_sapiens.GRCh38.dna_sm.primary_assembly.fa",
  memory = 16000,                    # 16GB RAM allocation
  indexSplit = TRUE                  # Split index for large genomes
)
```

Indexing Parameters: - `memory = 16000`: Allocate 16GB RAM for indexing - `indexSplit = TRUE`: Essential for large genomes (>4GB)

```
========listingfiles========
Homo_sapiens.GRCh38.dna_sm.primary_assembly.fa     Ref_Genome_GRCh38.01.b.tab
Homo_sapiens.GRCh38.dna_sm.primary_assembly.fa.gz  Ref_Genome_GRCh38.files
Ref_Genome_GRCh38.00.b.array                       Ref_Genome_GRCh38.log
Ref_Genome_GRCh38.00.b.tab                         Ref_Genome_GRCh38.reads
Ref_Genome_GRCh38.01.b.array
```

Figure 4: Indexed Ref Genome

5. Sequence Alignment with Rsubread

5.1 Prepare Read Files

```
# Define path containing trimmed reads
path <- "/home/vamouda/Suriya/Align/Seq"

# Identify paired-end read files
reads1 <- list.files(path = path, pattern = "*_R1_val_1.fq.gz$", full.names = TRUE)
reads2 <- list.files(path = path, pattern = "*_R2_val_2.fq.gz$", full.names = TRUE)

# Validate file pairs
if(length(reads1) != length(reads2)) {
  stop("Mismatch between forward (R1) and reverse (R2) read files!")
}
# Extract sample names
sample_names <- gsub("_R1_val_1.fq.gz$", "", basename(reads1))

# Display sample information
cat("## Sample Processing Information ##\n")
cat("Total samples:", length(sample_names), "\n\n")
for(i in 1:length(sample_names)) {
  cat("Sample", i, ":", sample_names[i], "\n")
}
```

5.2 Perform Alignment

```
# Execute alignment
align(
  index = "Ref_Genome_GRCh38",
  readfile1 = reads1,           # Forward reads
  readfile2 = reads2,           # Reverse reads
  input_format = "gzFASTQ",     # Compressed FASTQ input
  output_file = paste0(sample_names, ".bam"),  # Output BAM files
  output_format = "BAM",        # Binary Alignment Map format
  nthreads = 32,                # Multi-threading for speed
  unique = TRUE                 # Report only uniquely mapped reads
)
```

Alignment Parameters: - `nthreads = 32`: Utilize 32 CPU cores for parallel processing - `unique = TRUE`: Filter for uniquely mapped reads - `input_format = "gzFASTQ"`: Direct processing of compressed files

6. Read Quantification with featureCounts

6.1 Prepare BAM Files

```
# Define BAM file directory
bam_path <- "/home/vamouda/Suriya/Align/Bam"
bam_files <- list.files(path = bam_path, pattern = "\\.bam$", full.names = TRUE)
sample_names <- gsub("\\.bam$", "", basename(bam_files))

cat("BAM files found:", length(bam_files), "\n")
print(basename(bam_files))
```

```
========listingfiles========
JIPAI5_Tumour.bam  JIPAI7_Normal.bam  JIPB53_Normal.bam
JIPAI6_Tumour.bam  JIPAI7_Tumour.bam  JIPB64_Normal.bam
```

Figure 5: BAM Files

6.2 Run featureCounts

```
# Perform read counting
feature_Count <- featureCounts(
  files = bam_files,
  annot.ext = "/home/vamouda/Suriya/Align/GTF/Homo_sapiens.GRCh38.115.gtf.gz",
  isGTFAnnotationFile = TRUE,
  GTF.featureType = "exon",
  GTF.attrType = "gene_id",
  isPairedEnd = TRUE,
  requireBothEndsMapped = TRUE,
  minMQS = 10,
  strandSpecific = 0,
  countMultiMappingReads = FALSE,
  allowMultiOverlap = TRUE,
  largestOverlap = TRUE,
  nthreads = 32
)
```

6.3 Export Count Results

```
# Display summary statistics
cat("Count matrix dimensions:", dim(feature_Count$counts), "\n")
cat("Total counts per sample:\n")
print(colSums(feature_Count$counts))

# Export results
write.csv(feature_Count$counts, "gene_expression_counts.csv")
```

```r
# Count matrix with gene IDs
count_matrix <- data.frame(
  GeneID = rownames(feature_Count$counts),
  feature_Count$counts
)
write.csv(count_matrix, "gene_expression_counts_with_ids.csv", row.names = FALSE)

# Export alignment statistics
write.csv(feature_Count$stat, "alignment_assignment_statistics.csv")
```

7. Differential Expression Analysis with DESeq2

7.1 Data Preparation

```
# Load required libraries
library(DESeq2)
library(dplyr)

# Read count data and metadata
data <- read.csv("Count_data.csv", header = TRUE, row.names = 1)
meta <- read.csv("Meta_data.csv", row.names = 1)

# Display data
head(data, 6)
```

```
##                 JIPAI5_Tumour JIPAI6_Tumour JIPAI7_Normal JIPAI7_Tumour
## ENSG00000000003          1308           908           722           985
## ENSG00000000005             8             2             0             4
## ENSG00000000419          1771           976           843          1537
## ENSG00000000457          1239           581           649           538
## ENSG00000000460           194           373           363           134
## ENSG00000000938          2093          1312            36           264
##                 JIPB53_Normal JIPB64_Normal
## ENSG00000000003          1037          1228
## ENSG00000000005             2             4
## ENSG00000000419          1113          1350
## ENSG00000000457           365           766
## ENSG00000000460            74           339
## ENSG00000000938           156           337
```

```
head(meta, 6)
```

```
##                Tissue
## JIPAI5_Tumour  Tumour
## JIPAI6_Tumour  Tumour
## JIPAI7_Normal  Normal
## JIPAI7_Tumour  Tumour
## JIPB53_Normal  Normal
## JIPB64_Normal  Normal
```

```
# Validate sample matching
all(colnames(data) %in% rownames(meta))
```

```
## [1] TRUE
```

```
all(colnames(data) == rownames(meta))
```

```
## [1] TRUE
```

7.2 Create DESeq2 Object

```
# Create DESeqDataSet
dds <- DESeqDataSetFromMatrix(
  countData = data,
  colData = meta,
  design = ~ Tissue
)

# Pre-filter low count genes
keep <- rowSums(counts(dds)) >= 10
dds <- dds[keep,]
cat("Genes after filtering:", nrow(dds), "\n")
```

7.3 Factor Level Setting

```
# Set factor levels (important for interpretation)
dds$Tissue <- factor(dds$Tissue, levels = c("Tumor", "Normal"))
dds$Tissue <- relevel(dds$Tissue, ref = "Normal")
dds$Tissue <- droplevels(dds$Tissue)
```

7.4 Differential Expression Analysis

```
# Run DESeq2 analysis
dds <- DESeq(dds)

# Extract results
res <- results(dds)
cat("Summary of results:\n")
summary(res)

# Results with different significance thresholds
res05 <- results(dds, alpha = 0.05)
cat("Genes with padj < 0.05:", sum(res05$padj < 0.05, na.rm = TRUE), "\n")
```

7.5 Export Differential Expression Results

```
# Extract significantly differentially expressed genes
de_genes <- subset(res, res$pvalue <= 0.05 &
                   (res$log2FoldChange > 1 | res$log2FoldChange < -1))
write.csv(de_genes, "Data_UpandDown.csv")

# Upregulated genes
up_genes <- subset(res, res$pvalue <= 0.05 & res$log2FoldChange > 1)
write.csv(up_genes, "Data_Upregulated.csv")

# Downregulated genes
```

```r
down_genes <- subset(res, res$pvalue <= 0.05 & res$log2FoldChange < -1)
write.csv(down_genes, "Data_Downregulated.csv")
```

7.6 Add Gene Symbols

```r
library(biomaRt)

# Connect to Ensembl database
mart <- useDataset("hsapiens_gene_ensembl", useMart("ensembl"))

# Read DEG results
df <- read.csv("Data_UpandDown.csv")
genes <- df$ensembl_gene_id

# Get gene symbols
G_list <- getBM(
  filters = "ensembl_gene_id",
  attributes = c("ensembl_gene_id", "hgnc_symbol"),
  values = genes,
  mart = mart
)

# Merge with results
annotated_results <- merge(df, G_list, by = "ensembl_gene_id")
write.csv(annotated_results, "Data_UpandDown_withsymbol.csv")
```

8. Data Normalization and Visualization

8.1 Variance Stabilizing Transformation

```
# VST normalization for downstream analysis
vsd <- vst(dds, blind = FALSE)
rld <- rlog(dds, blind = FALSE)

# Export normalized data
write.csv(assay(vsd), "Data_vst_norm.csv")
write.csv(assay(rld), "Data_rlog_norm.csv")
```

8.2 Prepare Data for Visualization

```
# Extract normalized data for DEGs
DGE.results.sorted <- read.csv("Data_UpandDown.csv", row.names = 1)
DGEgenes <- rownames(subset(DGE.results.sorted, padj < 0.05))

normalizeddata <- read.csv("Data_vst_norm.csv", row.names = 1)
Visualization <- normalizeddata[DGEgenes,] %>% data.frame()
write.csv(Visualization, "Data_UpandDown_Visualization.csv")
```

8.3 Volcano Plot

```
#Volcano_plot
library(forcats)
library("ggrepel")
library(knitr)
data <- read.csv("Data.csv", header = TRUE, row.names = 1)
data <- data %>%
  dplyr::mutate(Expression = case_when(log2FoldChange >= 1 & pvalue <= 0.05 ~ "up",
                                       log2FoldChange <= -1 & pvalue <= 0.05 ~ "down",
                                       TRUE ~ "ns"))
#as.character(data$Expression)
#data %>%
#  count(Expression) %>%
#  knitr::kable()

data <- data %>%
  mutate(Expression = fct_relevel(Expression, "up", "down"))

data %>%
  distinct(Expression) %>%
  pull()

cols <- c("up" = "#bb0c00", "down" = "#00AFBB", "ns" = "grey")
sizes <- c("up" = 1.5, "down" = 1.5, "ns" = 0.5)
alphas <- c("up" = 1, "down" = 1, "ns" = 0.5)
```

```r
data %>%
  ggplot(aes(x = log2FoldChange,
             y = -log10(pvalue),
             fill = Expression,
             size = Expression,
             alpha = Expression)) +
  geom_point(shape = 21, colour = "grey90") +
  #geom_text_repel(data = sig_il_genes,  aes(label = Gene),
                  #size=3, nudge_x = 0.01, fontface='bold') +
  scale_fill_manual(values = cols) +
  scale_size_manual(values = sizes) +
  scale_alpha_manual(values = alphas) +
  scale_x_continuous(breaks = c(seq(-7, 7, 2)),
                     limits = c(-5, 7)) + scale_y_continuous(limits = c(1, 20)) +
  labs(title = "Differentially Expressed Genes") + theme_void() +
  theme(axis.line.x.bottom = element_line(color = 'black'),
        axis.line.y.left   = element_line(color = 'black'),
        panel.border = element_blank(),
        panel.grid.minor = element_blank(),
        panel.grid.major = element_blank(),
        panel.background = element_blank()) + theme_classic()
```

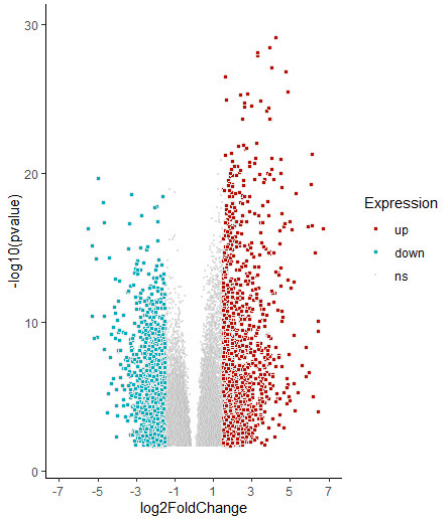

Figure 6: Volcano Plot

8.4 Heatmap Visualization

```r
library(pheatmap)

# Read data
Counts <- read.csv("Data_updown_Visualization.csv", header = TRUE, row.names = 1)
factors <- read.csv("Data.csv", header = TRUE, row.names = 1)

# Prepare annotation data
```

```r
factorsDS <- dplyr::select(factors, sex, Tissue)
factorsDS$sex <- factor(factorsDS$sex, levels = c("male", "female"))
factorsDS$Tissue <- factor(factorsDS$Tissue, levels = c("Normal", "Tumour"))

# Define colors
SexCol <- c("male" = "darkorchid", "female" = "darkorange")
TissueCol <- c("Normal" = "forestgreen", "Tumour" = "blue1")
AnnColour <- list(sex = SexCol, Tissue = TissueCol)

# Create heatmap
pheatmap(Counts,
         scale = "row",
         clustering_distance_rows = "manhattan",
         cluster_rows = TRUE,
         cluster_cols = TRUE,
         clustering_method = 'average',
         annotation_col = factorsDS,
         annotation_colors = AnnColour,
         show_rownames = FALSE,
         show_colnames = TRUE,
         fontsize = 6)
```

9 Session Information

```
sessionInfo()
```

```
## R version 4.3.3 (2024-02-29)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Oracle Linux Server 7.9
##
## Matrix products: default
## BLAS:   /usr/local/lib64/R/lib/libRblas.so
## LAPACK: /usr/local/lib64/R/lib/libRlapack.so;  LAPACK version 3.11.0
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## time zone: Asia/Kolkata
## tzcode source: system (glibc)
##
## attached base packages:
## [1] stats4    stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] dplyr_1.1.4                 DESeq2_1.42.1
##  [3] SummarizedExperiment_1.32.0 Biobase_2.62.0
##  [5] MatrixGenerics_1.14.0       matrixStats_1.5.0
##  [7] GenomicRanges_1.54.1        GenomeInfoDb_1.38.8
##  [9] IRanges_2.36.0              S4Vectors_0.40.2
## [11] BiocGenerics_0.48.1
##
## loaded via a namespace (and not attached):
##  [1] generics_0.1.4          SparseArray_1.2.4       bitops_1.0-9
##  [4] lattice_0.22-5          digest_0.6.37           magrittr_2.0.4
##  [7] evaluate_1.0.5          grid_4.3.3              RColorBrewer_1.1-3
## [10] fastmap_1.2.0           Matrix_1.6-5            scales_1.4.0
## [13] codetools_0.2-19        abind_1.4-8             cli_3.6.5
## [16] rlang_1.1.6             crayon_1.5.3            XVector_0.42.0
## [19] DelayedArray_0.28.0     yaml_2.3.10             S4Arrays_1.2.1
## [22] tools_4.3.3             parallel_4.3.3          BiocParallel_1.36.0
## [25] ggplot2_4.0.0           locfit_1.5-9.12         GenomeInfoDbData_1.2.11
## [28] vctrs_0.6.5             R6_2.6.1                lifecycle_1.0.4
## [31] zlibbioc_1.48.2         pkgconfig_2.0.3         pillar_1.11.1
## [34] gtable_0.3.6            glue_1.8.0              Rcpp_1.1.0
## [37] tidyselect_1.2.1        tibble_3.3.0            xfun_0.53
## [40] rstudioapi_0.17.1       knitr_1.50              dichromat_2.0-0.1
## [43] farver_2.1.2            htmltools_0.5.8.1       rmarkdown_2.30
## [46] compiler_4.3.3          S7_0.2.0                RCurl_1.98-1.17
```

Output Files Generated

- **Quality Control**: MultiQC reports, FastQC results
- **Alignment**: BAM files, alignment statistics
- **Quantification**: Gene count matrices
- **Differential Expression**: DEG lists, normalized counts
- **Visualization**: Volcano plots, heatmaps
- **Annotation**: Gene symbol-annotated results

This pipeline provides a complete, reproducible RNA-Seq analysis workflow from raw sequencing data to biological insights.