Project 2: RNASeq Analysis

Name: Iris Lee
ID: U05303694

Introduction

Type 1 diabetes (T1D) is an autoimmune disease in which the body's immune system mistakenly destroys insulin-producing beta-cells in the pancreas. Tyrosine kinase 2 (TYK2), a gene in the Janus kinase (JAK) family, plays a key role in type-I interferon signaling pathways and influencing beta-cell function and immune responses, making it a candidate gene associated with T1D. In this project, we aimed to investigate the data surrounding TYK2 knockout (KO) during pancreatic lineage differentiation, specifically focusing on the S5 (Endocrine Precursors) stage comparing WT and TYK2 KO [DOI]. The authors of the reference publication used bioinformatic techniques such as RNA-Sequencing to capture genome-wide expression changes, STAR for read alignment, FASTQC/MultiQC for quality assessment, and GSEA for gene ranking via fold change. This project reproduces the RNA-Sequencing and Differential Expression Sequencing analysis techniques in S5 of the reference publication, examines significant genes and pathways, and compares results to the original publication.

Methods

Source of Data

In our workflow, we analyzed sample data from the S5 (Endocrine Precursors) stage of the original publication [ref]. Paired-end reads in raw FASTQ format, the human reference genome (GRCh38), and annotation GTF file were all obtained from the original publication. The paired-end reads were formatted and imported into the workflow via Nextflow channels.

Quality Control

Initial quality control of paired-end read sequences was performed via FASTQC v0.12.1 [ref] with default parameters. FASTQC outputted HTML reports (`*.html`) which included statistics on per-base sequence quality, GC content, and adapter contamination. FASTQC outputs were compiled via MultiQC v1.25 [ref] with the parameter `-f .` into a single HTML report of sequence quality and summary statistics.

Genome Indexing and Alignment

The initial annotation GTF file was preprocessed via a Python v3.13.7 [ref] GTF extraction module (EXTRACT_GTF) to extract gene names and Ensembl ID's. The reference genome was indexed via STAR v2.7.11b [ref] with the parameters `--runThreadN $task.cpus --runMode genomeGenerate --genomeDir STAR_index --genomeFastaFiles $genome --sjdbGTFfile $gtf`. The paired-end reads were then aligned to the indexed reference genome via STAR with the parameters `--runThreadN $task.cpus --genomeDir $index --readFilesIn ${reads.join(" ")} --readFilesCommand zcat --outFileNamePrefix ${name}. --outSAMtype BAM Unsorted 2> ${name}.Log.final.out`, which generated sorted BAM files (`*.bam`) and log files (`*.Log.final.out`) documenting alignment metrics for each sample.

Read Quantification

Aligned reads were quantified via VERSE v0.1.5 [ref] with the parameter `-S`, and annotation GTF file as input. VERSE produced raw read counts in TXT format (`*.txt`) for each sample. Afterwards, all individual TXT files were compiled into a single counts matrix in TXT format (`*.txt`) via a Python v3.13.7 concatenate module (VERSE_CONCAT).

Results and Analyses

Quality Control Evaluation

Based on our MultiQC report, the number of reads across all samples varied within a consistent range, implying that sequencing depth was adequate and evenly distributed. There were no abnormally low read counts. FASTQC also showed no major issues across per-base sequence quality, GC content, and sequence duplication. Most samples passed, or only had minor warnings. MultiQC reported that only 12 samples had an extremely low percentage (<1%) of reads made up of overrepresented sequences. In addition, adapter content plots showed minimal to no adapter contamination (<0.1%). Finally, the STAR alignment summary showed high percentages of mapped reads to the reference genome. Based on this evaluation, we believe that the experiment was of high quality and the data was suitable for downstream analyses.

Filtering the Counts Matrix

We will first load in the required packages.

```r
library(DESeq2)
library(tidyverse)
library(dplyr)
library(ggplot2)
library(ggrepel)
library(scales)
library(reshape2)
```

Next, we will load in our data.

```r
counts <- read.table("results/verse_counts_matrix.txt", header = TRUE, sep = "\t", row.names = 1)
```

The filtering strategy we will use is a common DESeq2 filtering approach. We will keep genes with counts greater than or equal to 10 in at least three samples, since three is our smallest group size. This ensures that genes detected in only one or two replicates are excluded.

```r
keep <- rowSums(counts >= 10) >= 3
filtered_counts <- counts[keep, ]
```

To demonstrate the effects of our filtering strategy, we will record the number of counts before and after filtering in the following table:

```r
summary_df <- data.frame(
  Stage = c("Before Filtering", "After Filtering"),
  Number_of_genes = c(nrow(counts), nrow(filtered_counts))
)
print(summary_df)
```

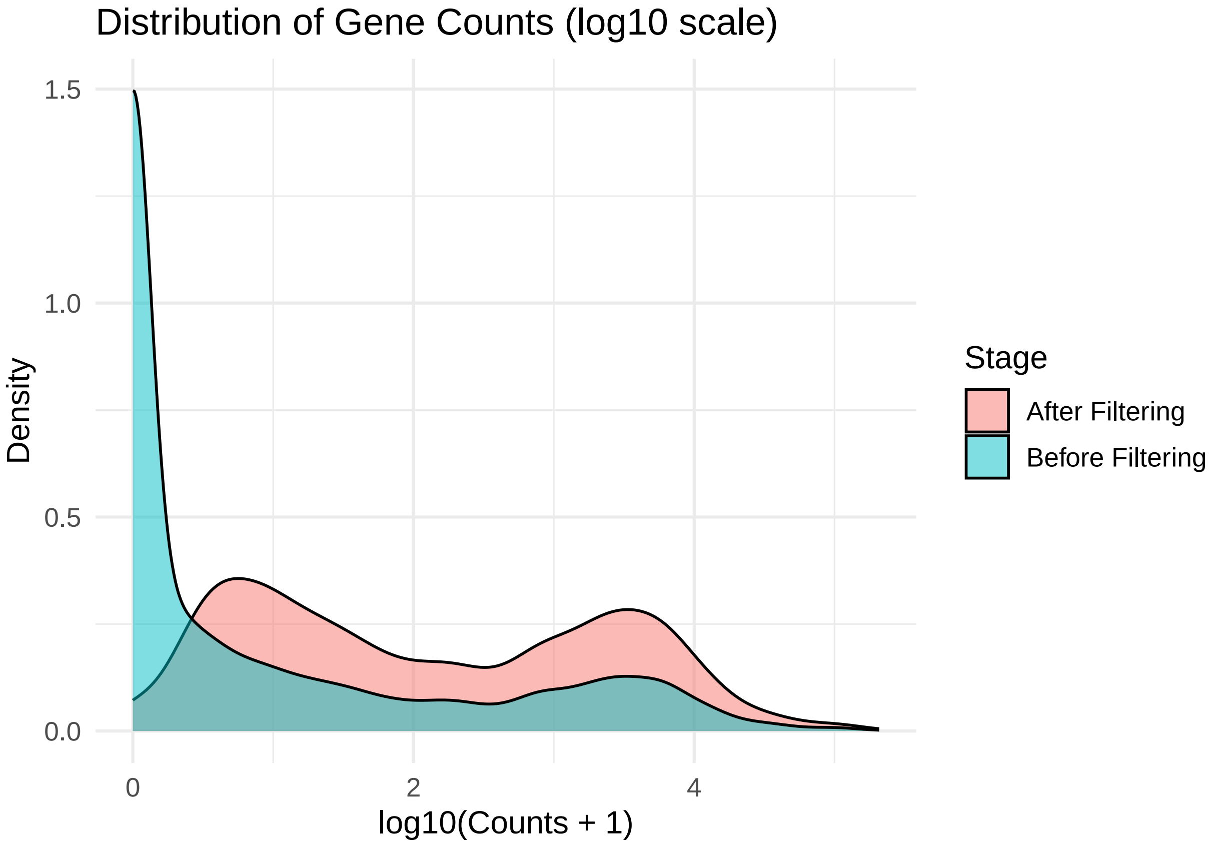
```
##              Stage Number_of_genes
## 1 Before Filtering           63242
## 2  After Filtering           28112
```

As shown in the table above, there were **63,242** genes before filtering and **28,112** genes after performing our filtering strategy.

To visualize the distribution of counts before and after filtering, we will plot a density plot of log-transformed counts before vs. after filtering. The reason we are using log-transformed counts is because raw counts are highly skewed.

```r
# ensure that all values in counts and filtered_counts are numeric
counts_numeric <- as.data.frame(sapply(counts, as.numeric))
rownames(counts_numeric) <- rownames(counts)
filtered_counts_numeric <- as.data.frame(sapply(filtered_counts, as.numeric))
```

```r
rownames(filtered_counts_numeric) <- rownames(filtered_counts)

# log-transform counts
log_counts_before <- log10(counts_numeric + 1)
log_counts_after <- log10(filtered_counts_numeric + 1)

# subsample genes (for faster plotting)
set.seed(123)
sample_genes <- sample(rownames(log_counts_before), min(5000, nrow(log_counts_before)))

# plot with reshape2 and ggplot2
plot_df <- data.frame(
value = c(as.numeric(as.matrix(log_counts_before[sample_genes, ])),
as.numeric(as.matrix(log_counts_after[sample_genes, ]))),
Stage = rep(c("Before Filtering", "After Filtering"),
each = length(sample_genes) * ncol(counts))
)
ggplot(plot_df, aes(x = value, fill = Stage)) +
geom_density(alpha = 0.5) +
labs(
title = "Distribution of Gene Counts (log10 scale)",
x = "log10(Counts + 1)",
y = "Density"
) +
theme_minimal(base_size = 12)
```



The density plot outlines that counts before filtering are right-skewed, and have a sharp peak near zero. This shows that there is a large number of genes that had very low or near-zero counts across samples, which is typical for raw RNAseq data prior to filtering. On the other hand, the counts after filtering have significantly less lowly expressed genes, as shown by the lack of low-count peak near zero. The remaining genes have much higher log-transformed counts, and form a smoother distribution of moderate to high expression levels (log10 counts between 1 and 4). Overall, this filtering strategy greatly improved our reads by eliminating reads with low expression in most samples.

Differential Expression Analysis

We will first load in our data for DESeq2. We will need our count data and our column data.

```r
# load in count data
filtered_counts <- data.frame(lapply(filtered_counts, as.numeric),
                              row.names = rownames(filtered_counts))
filtered_counts[is.na(filtered_counts)] <- 0

# load in column data
coldata <- data.frame(
  row.names = colnames(filtered_counts),
  condition = c("control", "control", "exp", "exp", "exp", "control")
)
```

Then, we will create our DESeq2 object.

```r
dds <- DESeqDataSetFromMatrix(countData = filtered_counts,
                              colData = coldata,
                              design = ~ condition)
```

Finally, we will run DESeq2 and extract its results.

```r
dds <- DESeq(dds)
res <- results(dds)
```

Next, we will add gene names and sort the genes by padj. This is done by loading in our output from EXTRACT_GTF to acquire all the Ensembl IDs and merging them with our DESeq2 results.

```r
gene_names <- read.table("results/gene_map.txt", header = TRUE, sep = "\t")
```

Before we proceed, we will first convert the DESeq2 results to a dataframe and merge it with the gene_names dataframe.

```r
# convert res to dataframe
res_df <- as.data.frame(res) %>%
  rownames_to_column(var = "ensembl_id")

# merge res and gene_names
merged_res <- res_df %>% left_join(gene_names) %>% relocate(gene_name)
```

We will then display the top 10 significant genes ranked by padj.

```r
# sort results by padj
sorted_padj_res <- merged_res %>%
  arrange(padj)

# display top 10 significant genes ranked by padj
top10_padj <- sorted_padj_res %>%
  slice(1:10)
top10_padj
```

```
##          gene_name         ensembl_id   baseMean log2FoldChange     lfcSE
## 1           RPS4Y1 ENSG00000129824.16 10666.7604      -8.774849 0.1461857
## 2  ENSG00000289575  ENSG00000289575.1   729.9084       3.696959 0.1278625
## 3             PNPO ENSG00000108439.11   603.9408      -4.157933 0.1442796
## 4         PCDHGA10  ENSG00000253846.3   495.6468       4.479698 0.1964898
## 5         YPEL3-DT  ENSG00000250616.4   524.3294      -2.421262 0.1196547
## 6        LINC02506  ENSG00000251129.3  1299.6450      -1.715761 0.1021718
## 7  ENSG00000289456  ENSG00000289456.1   383.0534      -2.134925 0.1447789
## 8  ENSG00000286339  ENSG00000286339.1   905.8164      -1.717322 0.1171004
## 9  ENSG00000282914  ENSG00000282914.1   120.8529       3.757790 0.2644911
## 10        SVIL-AS1  ENSG00000291093.1   152.5893       2.881192 0.2057644
##         stat        pvalue          padj
## 1  -60.02534  0.000000e+00  0.000000e+00
## 2   28.91356 8.064825e-184 8.258784e-180
## 3  -28.81857 1.255501e-182 8.571304e-179
## 4   22.79864 4.730326e-115 2.422045e-111
## 5  -20.23542  4.775258e-91  1.956041e-87
## 6  -16.79290  2.750651e-63  9.389346e-60
## 7  -14.74611  3.259383e-49  9.536490e-46
## 8  -14.66538  1.074169e-48  2.750007e-45
## 9   14.20762  8.216674e-46  1.869841e-42
## 10  14.00238  1.507315e-44  3.087131e-41
```

We will now choose a p-value threshold (padj < 0.05) and report the number of significant genes at this threshold.

```r
# Filter upregulated genes with padj < 0.05
upreg_sig_genes <- sorted_padj_res %>% filter(padj < 0.05 & log2FoldChange > 1) %>% select(gene_name)
write.table(upreg_sig_genes, file = "upreg_sig_genes.txt", col.names = F, row.names = F, quote = F)
upreg_sig_genes_count <- nrow(upreg_sig_genes)

# Filter downregulated genes with padj < 0.05
downreg_sig_genes <- sorted_padj_res %>% filter(padj < 0.05 & log2FoldChange <= -1) %>% select(gene_nam
write.table(downreg_sig_genes, file = "downreg_sig_genes.txt", col.names = F, row.names = F, quote = F)
downreg_sig_genes_count <- nrow(downreg_sig_genes)

# Display counts
print(paste("Significant Upregulated Genes:", upreg_sig_genes_count))
```

```
## [1] "Significant Upregulated Genes: 117"
```

```r
print(paste("Significant Downregulated Genes:", downreg_sig_genes_count))
```

```
## [1] "Significant Downregulated Genes: 181"
```

We can note that there are **117** significant upregulated genes and **181** significant downregulated genes after our filtering methods.

Afterwards, we will run an ENRICHR analysis on these significant genes.

After inputting the list of upregulated genes into ENRICHR, top pathway results (ranked by p-value) showed that these genes were most related to pathways for extracellular matrix organization, cardiac development and diseases, as well as cell growth signaling. Downregulated genes correlated most with tumor suppressors and cell signaling pathways, especially inflammatory and other immune responses, as well as cell cycle and apoptosis mechanisms.

Using our RNAseq results, we will perform a GSEA analysis via FGSEA. To perform FGSEA analysis, we will first generate a ranked list of our genes by log2 fold change (descending order).

```r
library(fgsea)

deseq2_res_ordered <- merged_res %>%
  drop_na() %>%
  distinct(gene_name, .keep_all = TRUE) %>%
  arrange(desc(log2FoldChange))

rnk_list <- setNames(deseq2_res_ordered$log2FoldChange, deseq2_res_ordered$gene_name)
```

Then, we will read in our GMT file, which is the C2 canonical pathways dataset from the Human MSigDB Collections.

```r
pathways <- gmtPathways("c2.cp.v2025.1.Hs.symbols.gmt")
```

We will now run FGSEA with default parameters.

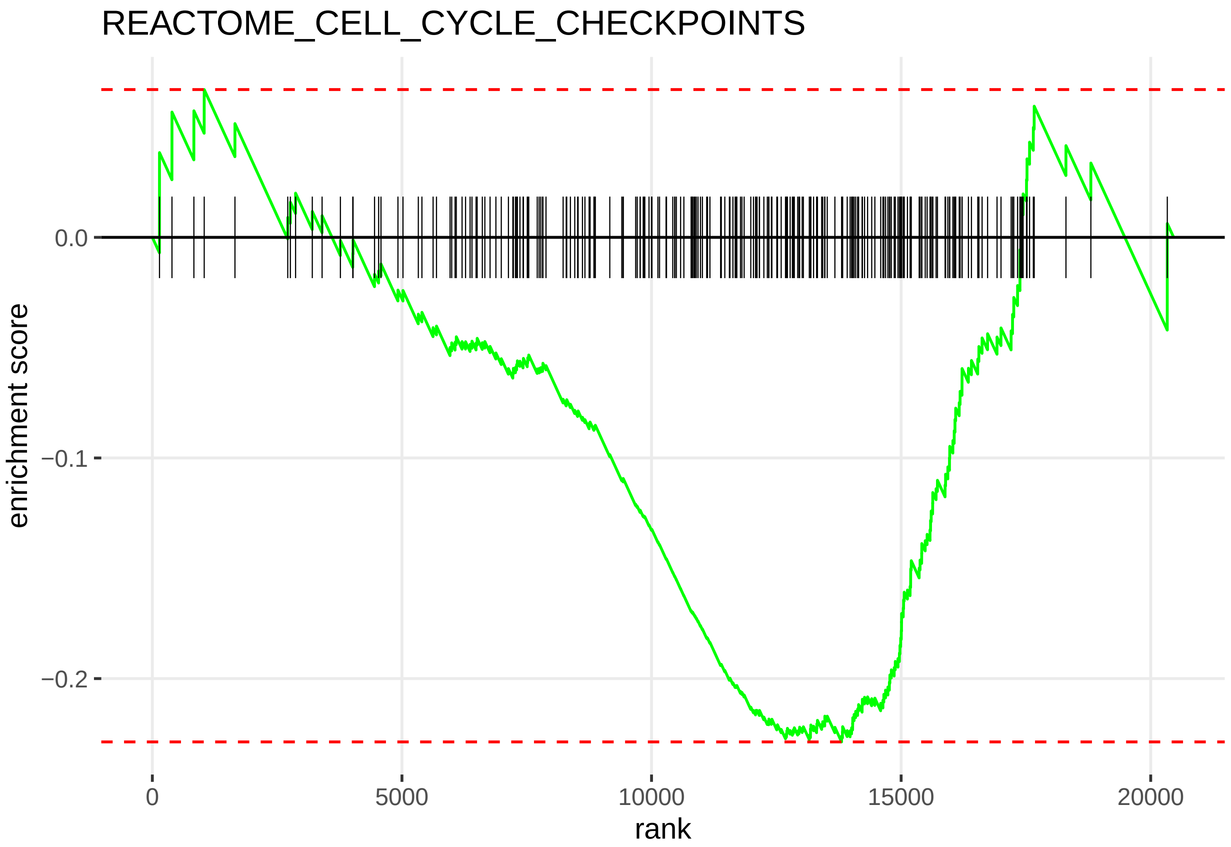
```r
fgseaRes <- fgsea(pathways, rnk_list)
```

```r
head(fgseaRes[order(padj), ])
```

```
##                                                                   pathway
##                                                                    <char>
## 1:                                              REACTOME_NEURONAL_SYSTEM
## 2:                                            PID_P53_DOWNSTREAM_PATHWAY
## 3:                                       WP_ADHD_AND_AUTISM_ASD_PATHWAYS
## 4:                     WP_CELL_LINEAGE_MAP_FOR_NEURONAL_DIFFERENTIATION
## 5:                       REACTOME_TRANSMISSION_ACROSS_CHEMICAL_SYNAPSES
## 6: REACTOME_ASSEMBLY_OF_COLLAGEN_FIBRILS_AND_OTHER_MULTIMERIC_STRUCTURES
##            pval         padj   log2err         ES       NES  size  leadingEdge
##           <num>        <num>     <num>      <num>     <num> <int>       <list>
## 1: 5.679315e-11 2.283653e-07 0.8513391 -0.5256818 -1.965363   350 SLITRK2,....
## 2: 1.203611e-06 1.613240e-03 0.6435518  0.5947273  2.052059   135 TP63, SP....
## 3: 9.360701e-07 1.613240e-03 0.6594444 -0.4773099 -1.774012   318 PNPO, CN....
## 4: 8.315347e-06 8.359002e-03 0.5933255 -0.6006011 -1.965929   105 SLC6A2, ....
## 5: 1.332562e-05 1.071647e-02 0.5933255 -0.4893647 -1.760171   228 DLG2, GN....
## 6: 3.240814e-05 2.067144e-02 0.5573322  0.6867219  2.098873    52 COL6A6, ....
```

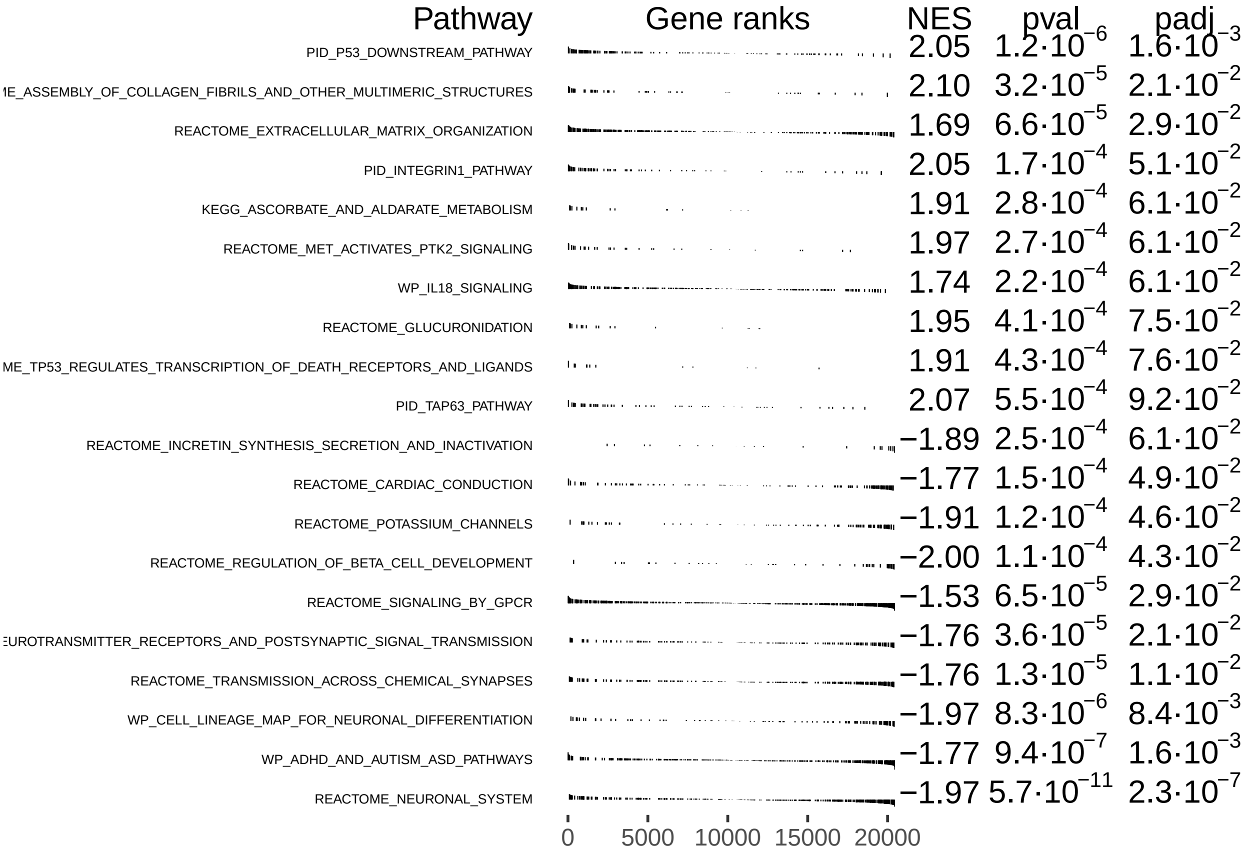
We can also plot these pathways and their enrichment scores:

```r
plotEnrichment(pathways[["REACTOME_CELL_CYCLE_CHECKPOINTS"]],
               rnk_list) + labs(title="REACTOME_CELL_CYCLE_CHECKPOINTS")
```



Using padj as our statistical threshold, we will generate a figure that displays the top most significant pathways from the FGSEA results.

```
topPathwaysUp <- fgseaRes[ES > 0][head(order(padj), n=10), pathway]
topPathwaysDown <- fgseaRes[ES < 0][head(order(padj), n=10), pathway]
topPathways <- c(topPathwaysUp, rev(topPathwaysDown))
plotGseaTable(pathways[topPathways], rnk_list, fgseaRes,
              gseaParam=0.5, pathwayLabelStyle=list(size=5))
```



The results of the FGSEA analysis show that the top most significant upregulated pathways were related to cell signaling, cell proliferation, extracellular matrix organization, and inflammatory responses, which is similar to our ENRICHR results.

The overlapping significant pathways of cell proliferation, extracellular matrix organization, and inflammatory/immune responses suggest that the treatment affects biological processes such as cell-to-cell interaction, cell apoptosis, and the immune system.

RNAseq Quality Control

We will now choose VST for our normalization strategy and generate a normalized counts matrix for this experiment.

```
vsd <- vst(dds, blind=FALSE)
norm_counts <- assay(vsd)

head(norm_counts)
```

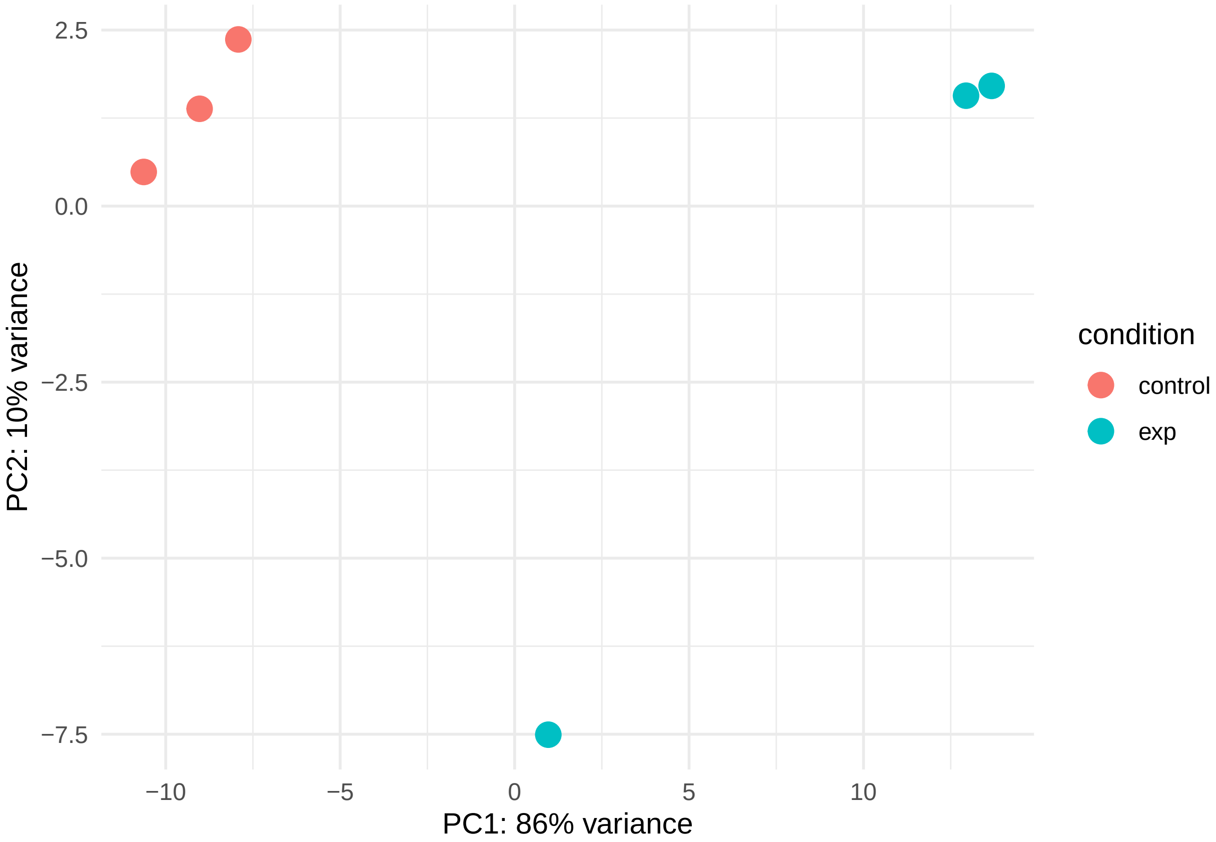
```
##                      control_rep2 control_rep1  exp_rep2  exp_rep1  exp_rep3
## ENSG00000000003.16      14.542093    14.410108 14.375483 14.554814 14.522734
## ENSG00000000005.6        7.385798     7.525201  7.630932  7.636458  7.433531
## ENSG00000000419.14      11.398441    11.321276 11.337540 11.400810 11.331836
## ENSG00000000457.14       9.926623     9.955094 10.108323 10.133692  9.845390
## ENSG00000000460.17      10.891424    10.826097 10.885406 10.912159 10.800797
## ENSG00000000971.17       7.093056     7.023229  7.339781  7.282827  7.185858
##                      control_rep3
## ENSG00000000003.16      14.461824
## ENSG00000000005.6        7.419032
## ENSG00000000419.14    11.310684
## ENSG00000000457.14     9.813055
## ENSG00000000460.17    10.866432
## ENSG00000000971.17     7.101022
```

Then, we will perform PCA on the normalized counts matrix and overlay the sample information in a biplot of PC1 vs. PC2.

```
pcaData <- plotPCA(vsd, intgroup = c("condition"), returnData = TRUE)
```

```
## using ntop=500 top features by variance
```

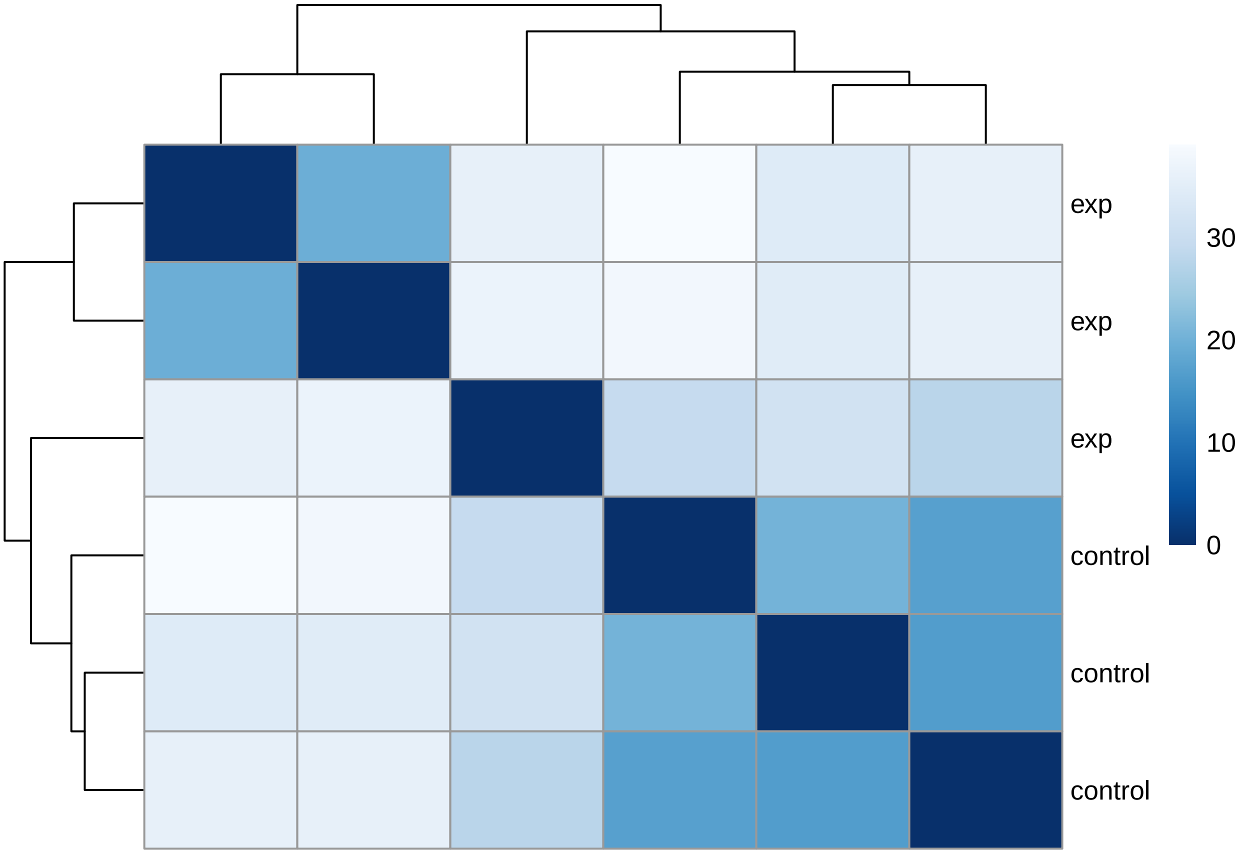
```
percentVar <- round(100 * attr(pcaData, "percentVar"))

ggplot(pcaData, aes(PC1, PC2, color = condition)) +
  geom_point(size = 4) +
  xlab(paste0("PC1: ", percentVar[1], "% variance")) +
  ylab(paste0("PC2: ", percentVar[2], "% variance")) +
  theme_minimal()
```



Afterwards, we will create a heatmap of the sample-to-sample distances for the experiment.

```
library("RColorBrewer")
library("pheatmap")

sampleDists <- dist(t(assay(vsd)))
sampleDistMatrix <- as.matrix(sampleDists)
rownames(sampleDistMatrix) <- paste(vsd$condition)
colnames(sampleDistMatrix) <- NULL
```

```r
colors <- colorRampPalette( rev(brewer.pal(9, "Blues")) )(255)
pheatmap(sampleDistMatrix,
         clustering_distance_rows=sampleDists,
         clustering_distance_cols=sampleDists,
         col=colors)
```



The PCA plot shows distinct separation between the control and experiment groups along the PC1 axis, thus explaining the 86% variance. In other words, technical noise is likely not responsible for a majority of the variation in the dataset. It can be hypothesized that the variation corresponds to the actual experimental condition. The close clustering of replicates within each group indicates consistent sample processing and high reproducibility.

The sample-to-sample distance matrix supports the PCA results in that samples closely cluster within their experiment groups and have great distance between groups. This pattern confirms that biological differences between conditions dominate over technical variation.

In conclusion, the distinct clustering of smaples within groups and great distance between groups in these plots depict high quality RNA-Seq data. Hence, the experiment was successful and the data is suitable for downstream differential expression analysis.

Replicating Figures 3C and 3F from Original Publication

The original publication generates figures 3C and 3F, which are a volcano plot and enrichment bar graph respectively.

We will first replicate Figure 3C with our data.

```r
volcano_data <- merged_res %>%
  mutate(
    significance = case_when(
      padj < 0.05 & log2FoldChange > 1 ~ "Upregulated",
      padj < 0.05 & log2FoldChange < -1 ~ "Downregulated",
      TRUE ~ "Not Significant"
    ),
    neglog10p = -log10(pvalue)
  )
# create labels that are spread out vertically to match Figure 3C
top20_labels <- bind_rows(
  volcano_data %>%
    filter(significance == "Upregulated", between(neglog10p, 0, 15)) %>%
    arrange(padj) %>%
    slice_head(n = 12) %>%
    mutate(label_side = "right"),
  volcano_data %>%
    filter(significance == "Downregulated", between(neglog10p, 0, 15)) %>%
    arrange(padj) %>%
    slice_head(n = 12) %>%
    mutate(label_side = "left")
)
# plot and format as close as possible
ggplot(volcano_data, aes(x = log2FoldChange, y = neglog10p)) +
  geom_point(aes(color = significance, shape = significance), alpha = 0.8, size = 1.8) +
  geom_text_repel(
    data = filter(top20_labels, label_side == "left"), aes(label = gene_name),
    color = "black", size = 3.4, box.padding = 0.9, point.padding = 0.8,
    segment.color = "grey60", direction = "y", nudge_x = -5, max.overlaps = Inf,
    force = 3, min.segment.length = 0, segment.curvature = -0.1, segment.angle = 20,
    seed = 123) +
  geom_text_repel(
    data = filter(top20_labels, label_side == "right"), aes(label = gene_name),
    color = "black", size = 3.4, box.padding = 0.9, point.padding = 0.8,
    segment.color = "grey60", direction = "y", nudge_x = 5, max.overlaps = Inf,
    force = 3, min.segment.length = 0, segment.curvature = 0.1, segment.angle = 20,
    seed = 123) +
  geom_vline(xintercept = c(-1, 1), linetype = "dashed", color = "black") +
  geom_hline(yintercept = -log10(0.05), linetype = "dashed", color = "black") +
  scale_color_manual(values = c(
    "Upregulated" = "salmon", "Downregulated" = "turquoise3", "Not Significant" = "gray70")) +
  scale_shape_manual(values = c(
    "Upregulated" = 16, "Downregulated" = 16, "Not Significant" = 16)) +
  coord_cartesian(ylim = c(0, 20), xlim = c(-10, 10)) +
  scale_y_continuous(breaks = seq(0, 20, 5)) +
  theme_minimal(base_size = 13) +
  theme(
    legend.position = "top",
    legend.title = element_blank(),
    plot.title = element_text(face = "bold", size = 15),
    plot.subtitle = element_text(size = 12, hjust = 0.5)
  ) +
  labs(
    title = "Figure 3C Volcano Plot of Differentially Expressed Genes",
    x = expression(Expression~Difference~(Log[2]~Fold~Change)),
    y = expression(Significance~(-Log[10]~pvalue))
  )
```

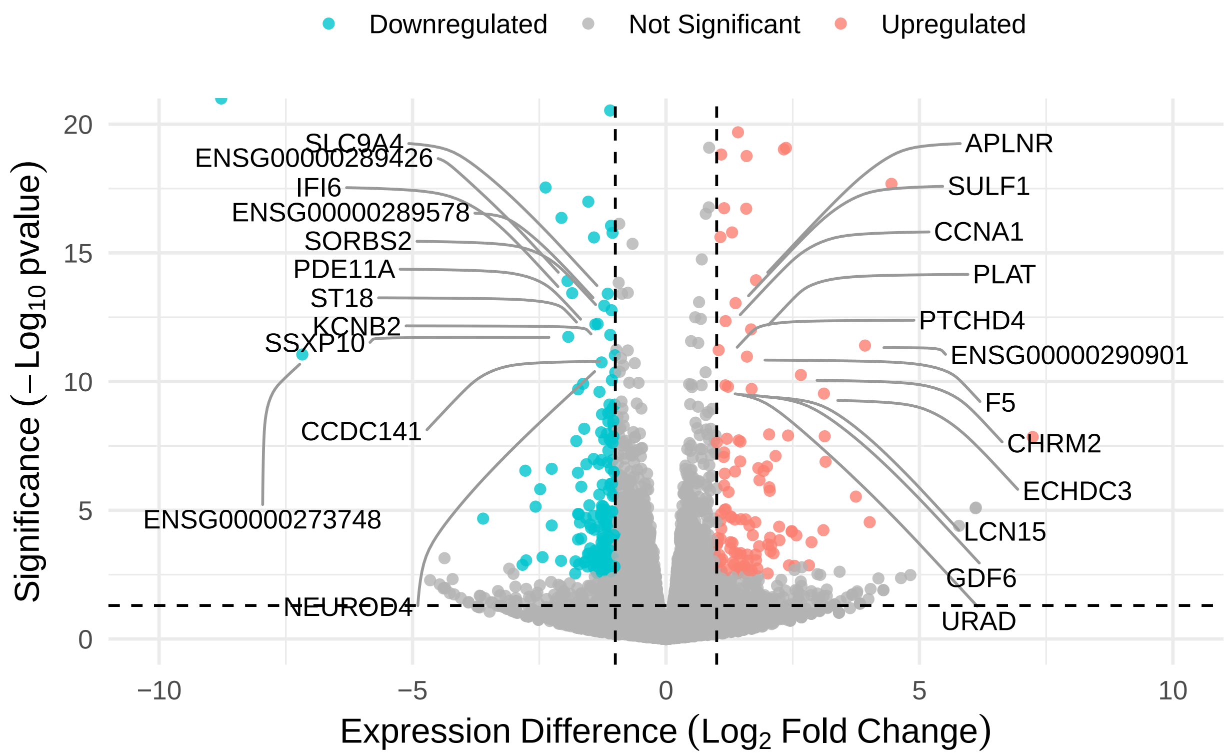
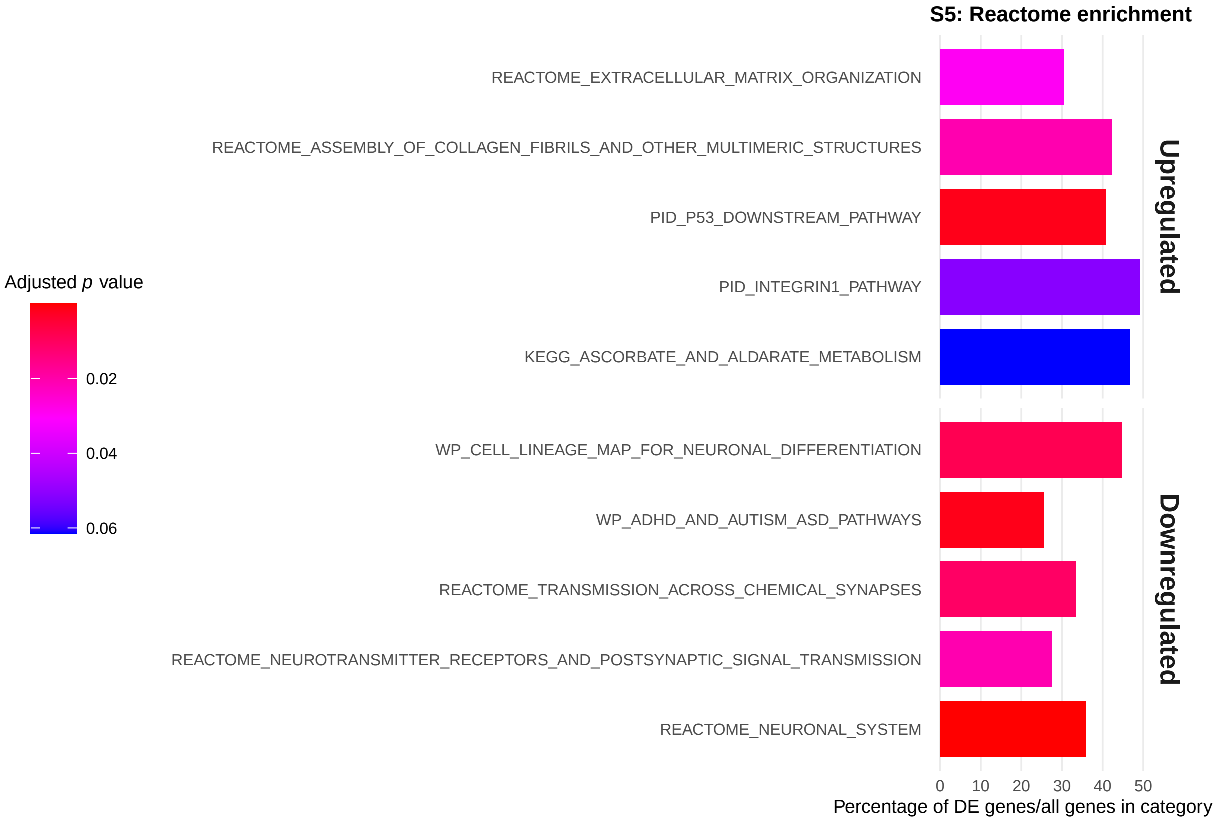


Figure 3C Volcano Plot of Differentially Expressed Genes

Next, we will replicate Figure 3F with our FGSEA data.

```
# reformat upregulated and downregulated pathway results into dataframes
fgseaRes_df <- fgseaRes %>%
  mutate(perc_DE = (lengths(leadingEdge) / size) * 100)

topPathwaysDown_df <- fgseaRes_df %>%
  filter(ES < 0) %>%
  arrange(padj) %>%
  slice_head(n = 5) %>%
  mutate(Direction = "Downregulated")

topPathwaysUp_df <- fgseaRes_df %>%
  filter(ES > 0) %>%
  arrange(padj) %>%
  slice_head(n = 5) %>%
  mutate(Direction = "Upregulated")

topPathways_df <- bind_rows(topPathwaysUp_df, topPathwaysDown_df)

topPathways_df <- topPathways_df %>%
  group_by(Direction) %>%
  arrange(padj, .by_group = TRUE) %>%
  mutate(pathway = factor(pathway, levels = unique(pathway)), pathway = str_wrap(pathway, width=20)) %>%
  ungroup()
```

```
topPathways_df$Direction <- factor(topPathways_df$Direction,
                                levels = c("Upregulated", "Downregulated"))

# make the barplot
ggplot(topPathways_df, aes(
  x = perc_DE,
  y = pathway,
  fill = padj
)) +
  geom_col(width = 0.8) +
  scale_fill_gradientn(
    colors = c("blue", "magenta", "red"),
    name = expression("Adjusted " * italic(p) * " value"),
    trans = "reverse",
    guide = guide_colorbar(reverse = FALSE)
  ) +
  facet_wrap(~Direction,
             scales = "free_y",
             ncol = 1,
             strip.position = "right") +
  labs(
    x = "Percentage of DE genes/all genes in category (%)",
    y = NULL,
    title = "S5: Reactome enrichment"
  ) +
  theme_minimal(base_size = 7) +
  theme(
    axis.text.y = element_text(size = 6, hjust = 1),
    axis.text.x = element_text(size = 6),
    strip.text = element_text(size = 10, face = "bold"),
    legend.position = "left",
    legend.title.align = 0.5,
    panel.grid.major.y = element_blank(),
    panel.grid.minor = element_blank(),
    plot.title = element_text(face = "bold", hjust = 0),
    plot.margin = margin(10, 10, 10, 10)
  ) +
  coord_cartesian(clip = "off")
```



In the original publication, their PCA results generated 319 upregulated and 412 downregulated genes. On the other hand, our PCA results generated 117 significant upregulated genes and 181 significant downregulated genes. The original publication specifically reports upregulation of genes related to signalling by receptor tyrosine kinases such as DUSP6, KRAS, SPP1, LAMB2, and COL2A1. Out of these upregulated genes, our list of upregulated genes does include KRAS and SPP1. Meanwhile, the original publication also reports downregulation of gene-sets associated with beta-cell development such as INSM1, NEUROD1, ONECUT1, PAX4, and PAX6. Out of these downregulated genes, our list of downregulated genes includes NEUROD4 and PAX6. It is possible that these differences are due to our procedures having stricter thresholds, and thus reporting a smaller number of significant genes.

By comparing our enrichment bar graph and that of the original publication, we can observe that there are striking similarities in upregulated and downregulated pathways. For instance, both graphs include upregulated extracellular matrix organization and integrin interaction pathways. For downregulated pathways, both exhibit neuronal system pathways. The original publication centers around TYK2 signaling and beta-cell production, and hence includes results that support their focus (i.e. upregulated signaling by receptor tyrosine kinases, downregulated beta-cell development and regulation of gene expression in beta-cells). Since our analyses were performed based on a more holistic perspective, our results may not be as specific as those of the original publication and are thus less focused on TYK2 and beta-cells. Overall, our project only includes data from the S5 stage of the original publication, which can explain some differences in our results. These differences may also be due to using different GMT pathway files.

Session Info

Below is a record of our session information.

```
sessioninfo::session_info()
```

```
## - Session info ---------------------------------------------------------------
##  setting  value
##  version  R version 4.4.3 (2025-02-28)
##  os       AlmaLinux 8.10 (Cerulean Leopard)
##  system   x86_64, linux-gnu
##  ui       X11
##  language (EN)
##  collate  en_US.UTF-8
##  ctype    en_US.UTF-8
##  tz       America/New_York
##  date     2025-10-24
##  pandoc   3.2 @ /usr/local/ood/rstudio-server-2024.12.0+467/bin/quarto/bin/tools/x86_64/ (via rmarkdo
##  quarto   1.5.57 @ /usr/local/ood/rstudio-server-2024.12.0+467/bin/quarto/bin/quarto
##
## - Packages -------------------------------------------------------------------
##  package              * version  date (UTC) lib source
##  abind                  1.4-8    2024-09-12 [2] CRAN (R 4.4.3)
##  Biobase              * 2.66.0   2024-10-29 [2] Bioconductor 3.20 (R 4.4.3)
##  BiocGenerics         * 0.52.0   2024-10-29 [2] Bioconductor 3.20 (R 4.4.3)
##  BiocParallel           1.40.2   2025-10-06 [2] Bioconductor
##  cli                    3.6.4    2025-02-13 [2] CRAN (R 4.4.3)
##  codetools              0.2-20   2024-03-31 [2] CRAN (R 4.4.3)
##  colorspace             2.1-2    2024-11-29 [2] R-Forge (R 4.4.3)
##  cowplot                1.1.3    2024-01-22 [2] CRAN (R 4.4.3)
##  crayon                 1.5.3    2024-06-20 [2] CRAN (R 4.4.3)
##  data.table             1.17.0   2025-02-22 [2] CRAN (R 4.4.3)
##  DelayedArray           0.32.0   2024-10-29 [2] Bioconductor 3.20 (R 4.4.3)
##  DESeq2               * 1.46.0   2024-10-29 [2] Bioconductor 3.20 (R 4.4.3)
##  digest                 0.6.37   2024-08-19 [2] CRAN (R 4.4.3)
##  dplyr                * 1.1.4    2023-11-17 [2] CRAN (R 4.4.3)
##  evaluate               1.0.3    2025-01-10 [2] CRAN (R 4.4.3)
##  farver                 2.1.2    2024-05-13 [2] CRAN (R 4.4.3)
##  fastmap                1.2.0    2024-05-15 [2] CRAN (R 4.4.3)
##  fastmatch              1.1-6    2024-12-23 [2] CRAN (R 4.4.3)
##  fgsea                * 1.32.4   2025-03-20 [1] Bioconductor 3.20 (R 4.4.3)
##  forcats              * 1.0.0    2023-01-29 [2] CRAN (R 4.4.3)
##  generics               0.1.3    2022-07-05 [2] CRAN (R 4.4.3)
##  GenomeInfoDb         * 1.42.3   2025-01-27 [2] Bioconductor 3.20 (R 4.4.3)
##  GenomeInfoDbData       1.2.13   2025-10-06 [2] Bioconductor
##  GenomicRanges        * 1.58.0   2024-10-29 [2] Bioconductor 3.20 (R 4.4.3)
##  ggplot2              * 3.5.1    2024-04-23 [2] CRAN (R 4.4.3)
##  ggrepel              * 0.9.6    2024-09-07 [2] CRAN (R 4.4.3)
##  glue                   1.8.0    2024-09-30 [2] CRAN (R 4.4.3)
##  gtable                 0.3.6    2024-10-25 [2] CRAN (R 4.4.3)
##  hms                    1.1.3    2023-03-21 [2] CRAN (R 4.4.3)
##  htmltools              0.5.8.1  2024-04-04 [2] CRAN (R 4.4.3)
##  httr                   1.4.7    2023-08-15 [2] CRAN (R 4.4.3)
##  IRanges              * 2.40.1   2024-12-05 [2] Bioconductor 3.20 (R 4.4.3)
##  jsonlite               1.9.1    2025-03-03 [2] CRAN (R 4.4.3)
##  knitr                  1.49     2024-11-08 [2] CRAN (R 4.4.3)
##  labeling               0.4.3    2023-08-29 [2] CRAN (R 4.4.3)
##  lattice                0.22-6   2024-03-20 [2] CRAN (R 4.4.3)
##  lifecycle              1.0.4    2023-11-07 [2] CRAN (R 4.4.3)
##  locfit                 1.5-9.11 2025-02-03 [2] CRAN (R 4.4.3)
##  lubridate            * 1.9.4    2024-12-08 [2] CRAN (R 4.4.3)
##  magrittr               2.0.3    2022-03-30 [2] CRAN (R 4.4.3)
##  Matrix                 1.7-4    2025-08-28 [2] CRAN (R 4.4.3)
##  MatrixGenerics       * 1.18.1   2025-01-09 [2] Bioconductor 3.20 (R 4.4.3)
##  matrixStats          * 1.5.0    2025-01-07 [2] CRAN (R 4.4.3)
##  munsell                0.5.1    2024-04-01 [2] CRAN (R 4.4.3)
##  pheatmap             * 1.0.12   2019-01-04 [2] CRAN (R 4.4.3)
##  pillar                 1.10.1   2025-01-07 [2] CRAN (R 4.4.3)
##  pkgconfig              2.0.3    2019-09-22 [2] CRAN (R 4.4.3)
##  plyr                   1.8.9    2023-10-02 [2] CRAN (R 4.4.3)
##  purrr                * 1.0.4    2025-02-05 [2] CRAN (R 4.4.3)
##  R6                     2.6.1    2025-02-15 [2] CRAN (R 4.4.3)
##  RColorBrewer         * 1.1-3    2022-04-03 [2] CRAN (R 4.4.3)
##  Rcpp                   1.0.14   2025-01-12 [2] CRAN (R 4.4.3)
##  readr                * 2.1.5    2024-01-10 [2] CRAN (R 4.4.3)
##  reshape2             * 1.4.4    2020-04-09 [2] CRAN (R 4.4.3)
##  rlang                  1.1.5    2025-01-17 [2] CRAN (R 4.4.3)
##  rmarkdown              2.29     2024-11-04 [2] CRAN (R 4.4.3)
##  rstudioapi             0.17.1   2024-10-22 [2] CRAN (R 4.4.3)
##  S4Arrays               1.6.0    2024-10-29 [2] Bioconductor 3.20 (R 4.4.3)
##  S4Vectors            * 0.44.0   2024-10-29 [2] Bioconductor 3.20 (R 4.4.3)
##  scales               * 1.3.0    2023-11-28 [2] CRAN (R 4.4.3)
##  sessioninfo            1.2.3    2025-02-05 [2] CRAN (R 4.4.3)
##  SparseArray            1.6.2    2025-02-20 [2] Bioconductor 3.20 (R 4.4.3)
##  stringi                1.8.4    2024-05-06 [2] CRAN (R 4.4.3)
##  stringr              * 1.5.1    2023-11-14 [2] CRAN (R 4.4.3)
##  SummarizedExperiment * 1.36.0   2024-10-29 [2] Bioconductor 3.20 (R 4.4.3)
##  tibble               * 3.2.1    2023-03-20 [2] CRAN (R 4.4.3)
##  tidyr                * 1.3.1    2024-01-24 [2] CRAN (R 4.4.3)
##  tidyselect             1.2.1    2024-03-11 [2] CRAN (R 4.4.3)
##  tidyverse            * 2.0.0    2023-02-22 [2] CRAN (R 4.4.3)
##  timechange             0.3.0    2024-01-18 [2] CRAN (R 4.4.3)
##  tinytex                0.56     2025-02-26 [2] CRAN (R 4.4.3)
##  tzdb                   0.4.0    2023-05-12 [2] CRAN (R 4.4.3)
##  UCSC.utils             1.2.0    2024-10-29 [2] Bioconductor 3.20 (R 4.4.3)
##  vctrs                  0.6.5    2023-12-01 [2] CRAN (R 4.4.3)
##  withr                  3.0.2    2024-10-28 [2] CRAN (R 4.4.3)
##  xfun                   0.51     2025-02-19 [2] CRAN (R 4.4.3)
##  XVector                0.46.0   2024-10-29 [2] Bioconductor 3.20 (R 4.4.3)
##  yaml                   2.3.10   2024-07-26 [2] CRAN (R 4.4.3)
##  zlibbioc               1.52.0   2024-10-29 [2] Bioconductor 3.20 (R 4.4.3)
##
##  [1] /usr4/bf527/ihlee14/R/x86_64-pc-linux-gnu-library/4.4
##  [2] /share/pkg.8/r/4.4.3/install/lib64/R/library
##  * -- Packages attached to the search path.
##
## ------------------------------------------------------------------------------
```