Shuwenyu BF528 Project2 Report

Shuwenyu

2025-10-24

Shuwenyu BF528 Project2 Report

Introduction

- What is the biological background of the study?

Type 1 diabetes(T1D) is a long lasting autoimmune condition where the immune system mistakenly attacks and destroys the insulin-producing beta cells in the pancreas. A key gene linked to T1D risk is tyrosine kinase2(TYK) a part of the JAK family, strongly liked to T1D risk because it's vital for type-l interferon (IFN-I) signaling, which promotes the immune attack on ß-cells. Understanding TYK2's role in pancreatic development, function, and its response to immune challenges is crucial for developing new treatments for T1D.

- Why was the study performed?

The study was performed to understand the role of the T1D candidate gene, TYK2, in pancreatic -cell development, function, and its responses to immune challenges like type-I interferons. The goal was to uncover mechanisms through which TYK2 contributes to T1D and identify potential therapeutic targets to halt its progression.

- Why did the authors use the bioinformatic techniques they did?

• Deep RNA sequencing (RNA-seq) and Principal Component Analysis (PCA): These were used to globally map gene expression patterns throughout pancreatic development in both wild-type and TYK2 knockout cells. This helped identify T1D candidate genes, confirm TYK2's expression profile, and uncover broad transcriptional changes, such as the unexpected upregulation of KRAS, distinguishing the TYK2 KO phenotype. PCA specifically visualized these differences across developmental stages.

• Reactome and KEGG enrichment analyses: Applied to both bulk and single-cell RNA-seq data, these analyses helped pinpoint specific biological pathways and processes significantly affected by TYK2 loss. This was crucial for understanding how TYK2 impacts -cell development, cell cycle regulation, and immune responses like antigen processing and presentation.

• Single-cell RNA sequencing (scRNA-seq): This advanced technique allowed for a detailed, specific understanding of how TYK2 knockout impacted different pancreatic cell populations and their responses to interferon- . It provided granular insights into which cell types were most affected and the specific genes involved. The Seurat pipeline was essential for processing and interpreting this complex single-cell data, including estimating cell cycle phases.

• Correlation studies using existing human RNA-seq datasets: By analyzing data from human fetal pancreatic and adult islets, the authors could corroborate their findings from stem cell models with real human tissue, validating the physiological relevance of TYK2's correlation with genes like NEUROG3, INS, and KRAS.

• Phenome-wide association analysis (FinnGen cohort): This large-scale genetic analysis was used to confirm in a real-world human population that specific loss of function TYK2 gene variants offer protection against

T1D and other autoimmune diseases, providing strong clinical validation for the therapeutic potential of TYK2 inhibition.

Methods

All analyses were performed using a Nextflow-based RNA-seq pipeline executed with the command nextflow run main.nf -profile singularity,local. Each process was containerized using pre-built Singularity images hosted in the BF528 GitHub Container Registry (e.g., ghcr.io/bf528/star:latest). These containers ensured version consistency and reproducibility across all tools, including FastQC, MultiQC, STAR, VERSE, and pandas/Python utilities.

- Gene Annotation Parsing (PARSE_GTF)

To extract gene-level information from the reference annotation, a custom Python script (parse_gtf.py) was used to parse the GTF file and generate a tab-delimited mapping between Ensembl gene IDs and gene symbols. This step was run within the ghcr.io/bf528/pandas:latest container using the following parameters:

```
python parse_gtf.py -i <annotation.gtf> -o gene_ids_to_names.txt
```

The -i flag specified the input GTF file, and -o defined the output file path. All other settings followed the script's default configuration. The resulting file (gene_ids_to_names.txt) was later used for gene name annotation in downstream DESeq2 analysis.

- Quality Control of Raw Reads (FASTQC)

Sequencing read quality was assessed using FastQC (v0.12.1) within the ghcr.io/bf528/fastqc:latest container. Each paired-end FASTQ file was analyzed independently to evaluate sequence quality, GC content, and adapter contamination. FastQC was executed with the following parameters:

```
fastqc <input.fastq.gz>
```

The program generated both .zip and .html outputs for each sample, which were used for visual inspection and MultiQC aggregation.

- Genome Index Generation (STAR INDEX)

To prepare the reference for alignment, STAR (v2.7.11b) was used to build a genome index from the provided FASTA and GTF files within the container ghcr.io/bf528/star:latest. The process ran under the process_high label (16 CPUs) using the command:

```
STAR –runMode genomeGenerate
–runThreadN $task.cpus
–genomeDir star_index
–genomeFastaFiles <reference_genome.fa>
–sjdbGTFfile <annotation.gtf>
```

The parameters –runMode genomeGenerate, –runThreadN, –genomeDir, –genomeFastaFiles, and –sjdbGTFfile explicitly defined the mode, thread number, output directory, genome FASTA input, and annotation file respectively. All other STAR options were left at default values.

- Read Alignment to the Reference Genome (STAR ALIGN)

Paired-end reads were aligned to the indexed human genome using STAR (v2.7.11b) within the same container. Each sample was processed with the following parameters:

```
STAR –runThreadN $task.cpus
–genomeDir star_index
–readFilesIn <read1.fastq.gz> <read2.fastq.gz>
–readFilesCommand zcat
–outFileNamePrefix .
–outSAMtype BAM SortedByCoordinate
2> .Log.final.out
```

The –runThreadN option defined available CPUs, –readFilesCommand zcat enabled reading compressed files, and –outSAMtype BAM SortedByCoordinate generated coordinate-sorted BAM files. Standard error (2>) was redirected to a .Log.final.out file, which summarized mapping statistics. All unlisted parameters used STAR's defaults.

- Quantifying Gene Expression (VERSE)

Aligned reads were quantified at the exon level using VERSE (v0.1.5) within the ghcr.io/bf528/verse:latest container. This process was using the command:

verse -a <annotation.gtf> -s -o <input.bam>

-a specified the GTF annotation file, -s indicated stranded counting mode, and -o defined the output prefix for each sample.

Each run produced an output file named .exon.txt, containing two columns: gene identifier and raw read count. All other VERSE parameters were kept at default values.

- Post-Alignment Quality Aggregation (MULTIQC)

Quality-control metrics from both FastQC and STAR were aggregated using MultiQC (v1.25) within the ghcr.io/bf528/multiqc:latest container. All relevant .zip, .html, and .Log.final.out files were collected into a single directory and analyzed with:

multiqc -f .

The -f flag allowed overwriting existing reports. The output file multiqc_report.html summarized overall sequencing quality, adapter content, and alignment performance for all samples.

- Combining Gene Counts (CONCAT_COUNTS)

Individual count tables from VERSE were merged into a single matrix using a custom Python script (concat_counts.py) executed in the ghcr.io/bf528/pandas:latest container. This script concatenated all exon-level count files column-wise, replacing missing values with zeros. The command was:

python concat_counts.py -i -o counts_matrix.tsv

The -i argument accepted multiple VERSE output files, and -o specified the name of the merged matrix file. The resulting matrix contained genes as rows and samples as columns.

- Workflow

In the nextflow pipeline main.nf, raw paired-end FASTQ files were paired using Channel.fromFilePairs and processed by FASTQC to assess sequencing quality. In parallel, PARSE_GTF extracted gene identifiers from the reference GTF, and STAR generated the genome index used by STAR_ALIGN for read alignment to the reference genome. Alignment outputs, including sorted BAM files and log summaries, were aggregated by MULTIQC to produce an overall quality report. Aligned reads were then quantified at the exon level using VERSE, and the resulting count tables were merged by CONCAT_COUNTS into a unified counts_matrix.tsv for downstream DESeq2 analysis.

- Differential Expression Analysis

Downstream statistical analysis was performed in R (version 4.4.3) using the DESeq2 package. The merged count matrix and sample metadata were imported, and a DESeqDataSet object was constructed using the design formula ~ condition (experimental vs. control). DESeq2 normalized counts using the median-of-ratios method and estimated gene-wise dispersions and fold changes using the Wald test. Genes with adjusted p-values (padj) < 0.05 and $|\log \text{FoldChange}| > 1$ were considered significantly differentially expressed. All DESeq2 functions were executed with default parameters.

- Normalization and Quality Control

To assess sample variability and normalization performance, counts were transformed using the regularized log transformation (rlog(dds, blind = TRUE)). Principal component analysis (PCA) was performed using

DESeq2's plotPCA() function, and the first two principal components (PC1 and PC2) were plotted to visualize clustering by condition. A sample-to-sample distance matrix was computed using dist(t(assay(rld))) and visualized as a heatmap using pheatmap. All steps used default parameters unless otherwise specified.

- Functional Enrichment and Gene Set Analysis

Functional enrichment analyses were performed using both over-representation and gene set enrichment approaches.

Over-Representation Analysis (ORA): Significantly up and down regulated gene lists were exported from DESeq2 and analyzed separately using the Enrichr web tools. Each gene list was tested for enrichment in the WikiPathways 2024 Human pathway databases. Default settings were used for both platforms.

Gene Set Enrichment Analysis (GSEA) The complete ranked gene list was analyzed using the fgsea package in R. Canonical pathway sets were imported from the MSigDB C2 collection in Gene Symbol format. The fgsea() function was executed as follows:

fgsea(pathways, rnk_list, minSize = 15, maxSize = 500)

Pathways were considered significant if $padj < 0.05$. Enrichment plots were generated using plotEnrichment(), and the top 10 up and down regulated pathways were visualized using plotGseaTable().

- Visualization and Reporting

A volcano plot was generated using ggplot2, displaying log fold changes versus –log (padj) to highlight significant genes. PCA and heatmap plots summarized global transcriptomic variation and sample similarity.

Quality Control Evaluation

The total number of reads across all samples ranged from approximately 80 to 110 million paired-end reads per sample. FastQC analysis did not flag any major quality issues across samples, and per-base sequence quality scores remained high throughout the read length. Less than 1% of reads in all 12 samples were identified as overrepresented sequences, and no adapter contamination was detected.

STAR alignment showed consistently high mapping rates across all six samples, with 95.8–97.8% of reads successfully aligned to the reference genome. Among these, 91.9–94.0% were uniquely mapped, and the mismatch rate remained low at approximately 0.2%. The average mapped read length was around 199 bp. Based on the overall quality metrics from both FastQC and STAR, the sequencing data were of high quality and suitable for downstream differential expression analysis.

Load the DESeq2 Library and tidyverse

colData

We also build what DESeq2 refers to as the colData, which is a dataframe that describes the sample ids and whether they are control group or experimental group samples.

```r
coldata <- read.csv('results/coldata.csv')
coldata
```

```
##      sample_id condition
## 1 control_rep1   control
## 2 control_rep2   control
## 3 control_rep3   control
## 4     exp_rep1       exp
## 5     exp_rep2       exp
## 6     exp_rep3       exp
```

Read in the counts matrix

R has a number of built-in functions to read from a dataframe / csv. We use `read.csv` to read in the contents of the counts_matrix.tsv file.

```
cts <- as.matrix(read.delim("results/counts_matrix.tsv", row.names='gene'))
cts <- cts[, coldata$sample_id]
head(cts,10)
```

```
##                    control_rep1 control_rep2 control_rep3 exp_rep1 exp_rep2
## ENSG00000290825.1             0            0            0        2        3
## ENSG00000223972.6             0            0            0        0        0
## ENSG00000227232.6           150          146          220      164      200
## ENSG00000278267.1             0            0            2        0        0
## ENSG00000243485.5             2            1            2        4        4
## ENSG00000284332.1             0            0            0        0        0
## ENSG00000237613.2             0            0            0        0        0
## ENSG00000268020.3             0            0            0        0        0
## ENSG00000290826.1             0            0            0        0        0
## ENSG00000240361.3             0            0            0        0        0
##                    exp_rep3
## ENSG00000290825.1         1
## ENSG00000223972.6         0
## ENSG00000227232.6       212
## ENSG00000278267.1         0
## ENSG00000243485.5         1
## ENSG00000284332.1         0
## ENSG00000237613.2         0
## ENSG00000268020.3         0
## ENSG00000290826.1         0
## ENSG00000240361.3         0
```

```
identical(colnames(cts), coldata$sample_id)
```

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

Constructing the DESeq object

Once we have all of the above inputs, we need to load them into a DESeq2 object that DESeq2 will then use to perform differential expression testing.

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

```
## Warning in DESeqDataSet(se, design = design, ignoreRank): some variables in
## design formula are characters, converting to factors
```

You can see that we have provided the DESeqDataSetFromMatrix function the necessary inputs and this pre-built function will generate a DESeq2 object, which is often named dds.

Filtering the counts matrix

To improve the reliability of the differential expression analysis, genes with low read counts across samples were filtered out prior to running DESeq2. Specifically, genes were required to have at least 10 read counts in a minimum of three samples, corresponding to the smallest experimental group size. This criterion ensures that only genes with sufficient expression evidence across biological replicates are retained for downstream analysis. Before filtering, a total of 63,241 genes were present in the dataset, which was reduced to 20,579 genes after applying the threshold.

The distribution plots below illustrate this change: before filtering, the majority of genes exhibited low total counts (log counts < 2), whereas after filtering, the distribution shifted toward higher counts, indicating the effective removal of lowly expressed and uninformative genes. This filtering step enhances statistical power and reduces noise in subsequent differential expression analyses.

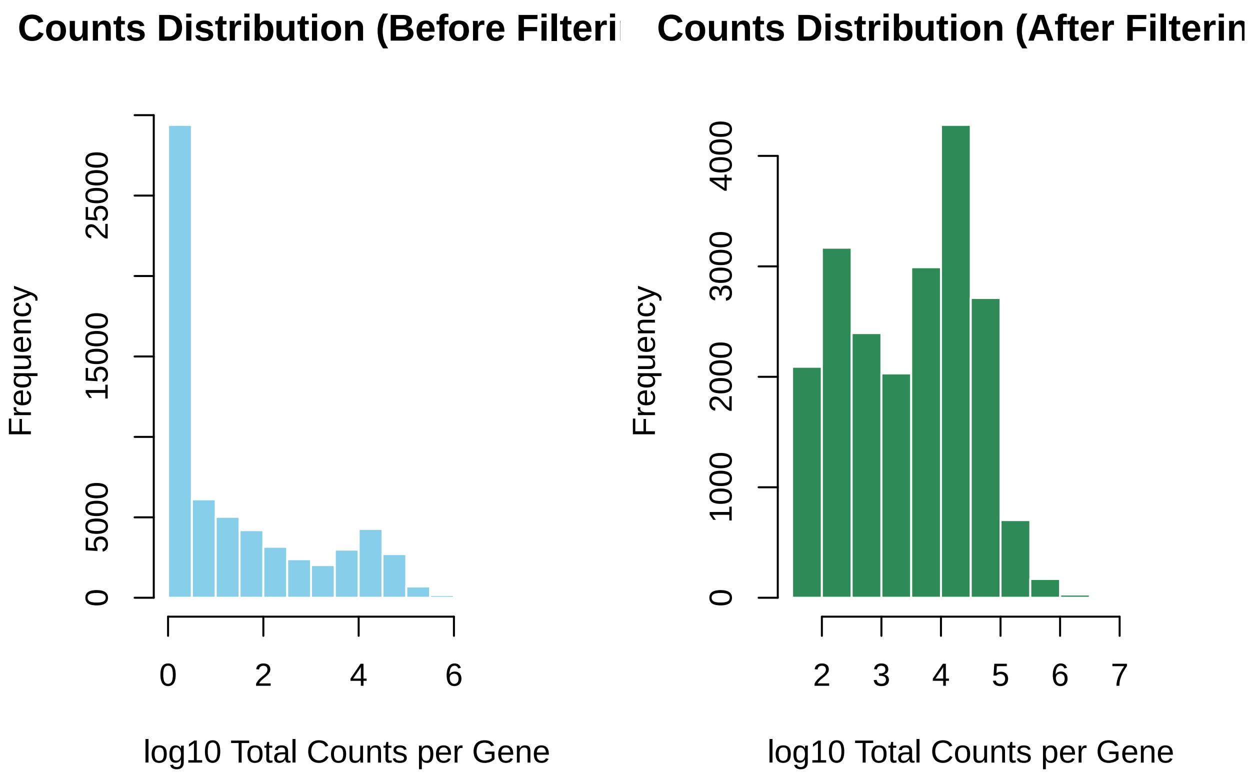
```r
smallestGroupSize <- 3

counts_before <- counts(dds)
tot_before <- rowSums(counts_before)


keep <- rowSums(counts_before >= 10) >= smallestGroupSize
n_before <- nrow(dds)
dds <- dds[keep, ]
n_after <- nrow(dds)

counts_after <- counts(dds)
tot_after <- rowSums(counts_after)

cat("Genes before filtering:", n_before, "\n")
```

```
## Genes before filtering: 63241
```

```r
cat("Genes after filtering:", n_after, "\n")
```

```
## Genes after filtering: 20579
```

```r
par(mfrow=c(1,2))
hist(log10(tot_before + 1),
     main="Counts Distribution (Before Filtering)",
     xlab="log10 Total Counts per Gene",
     col="skyblue", border="white")
hist(log10(tot_after + 1),
     main="Counts Distribution (After Filtering)",
     xlab="log10 Total Counts per Gene",
     col="seagreen", border="white")
```



```
par(mfrow=c(1,1))
```

```
cat("Genes before filtering:", n_before, "\n")
```

```
## Genes before filtering: 63241
```

```
cat("Genes after filtering:", n_after, "\n")
```

```
## Genes after filtering: 20579
```

Running DESeq

Once you have your object, you will run DESeq2 as such:

```
dds <- DESeq(dds)
```

```
## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing
```

```
res <- results(dds, contrast=c("condition","exp","control"))
```

The results function can extract the results of interest from the dds object where the DESeq2 software stored both the input and the outputs it generated.

The following code simply orders the results by those with the lowest padj.

```
resOrdered <- res[order(res$padj),]
sorted_padj_res <- as_tibble(resOrdered, rownames='gene')
head(sorted_padj_res,10)
```

```
## # A tibble: 10 x 7
##    gene               baseMean log2FoldChange  lfcSE  stat    pvalue      padj
##    <chr>                 <dbl>          <dbl>  <dbl> <dbl>     <dbl>     <dbl>
##  1 ENSG00000129824.16   10677.          -8.77 0.142  -61.7 0         0
##  2 ENSG00000289575.1      732.           3.70 0.125   29.7 1.66e-193 1.58e-189
##  3 ENSG00000108439.11     605.          -4.16 0.141  -29.5 2.13e-191 1.35e-187
##  4 ENSG00000253846.3      497.           4.48 0.200   22.4 6.36e-111 3.02e-107
##  5 ENSG00000250616.4      525.          -2.42 0.115  -21.0 1.03e- 97 3.92e- 94
##  6 ENSG00000251129.3     1301.          -1.71 0.0986 -17.4 1.17e- 67 3.70e- 64
##  7 ENSG00000173262.12     634.           4.69 0.279   16.8 2.51e- 63 6.81e- 60
##  8 ENSG00000286339.1      907.          -1.71 0.115  -14.9 2.41e- 50 5.71e- 47
##  9 ENSG00000289456.1      384.          -2.13 0.145  -14.7 3.28e- 49 6.92e- 46
## 10 ENSG00000282914.1      121.           3.76 0.264   14.2 6.86e- 46 1.30e- 42
```

Differential Expression Analysis

1. Table of the top 10 differentially expressed genes and the statistics provided by DESeq2 as ranked by adjusted p-value

The DESeq2 results were ranked by adjusted p-value (padj) to identify the top 10 most significantly differentially expressed genes between the experimental and control groups. Each gene entry includes the baseMean, log fold change, standard error, Wald statistic, and p-values, summarizing both the magnitude and reliability of expression changes. These top genes represent those with the strongest statistical evidence for differential expression and will guide downstream biological interpretation.

```
summary(res)
```

```
##
## out of 20579 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)       : 750, 3.6%
## LFC < 0 (down)     : 860, 4.2%
## outliers [1]       : 0, 0%
## low counts [2]     : 1596, 7.8%
## (mean count < 14)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

Joining our gene names

Most languages that allow you to manipulate dataframes (tabular, 2-dimensional data) enable you to perform `join` operations. These are ways of joining two different dataframes or tables together on some common column.

Read in our output from `PARSE_GTF`. Use a tidyverse function to read in the gene_ids_to_names.txt file as a tibble. You may view the file if that helps you figure out which function to use. When you read in the file, ensure that you name the columns `gene` and `symbol`, respectively.

Print out the file you read in as a tibble.

```
ids<- read_tsv("results/gene_ids_to_names.txt", skip = 1, col_names = c("gene", "symbol"))
```

```
## Rows: 63241 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (2): gene, symbol
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(ids,10)
```

```
## # A tibble: 10 x 2
##    gene               symbol
##    <chr>              <chr>
##  1 ENSG00000290825.1 DDX11L2
##  2 ENSG00000223972.6 DDX11L1
##  3 ENSG00000227232.6 WASH7P
##  4 ENSG00000278267.1 MIR6859-1
##  5 ENSG00000243485.5 MIR1302-2HG
##  6 ENSG00000284332.1 MIR1302-2
##  7 ENSG00000237613.2 FAM138A
##  8 ENSG00000268020.3 OR4G4P
##  9 ENSG00000290826.1 ENSG00000290826
## 10 ENSG00000240361.3 OR4G11P
```

Notice how the gene column matches up with the gene column in our DESeq2 results. We can use this column to join the information from the dataframe (tibble) we just created with our DESeq2 results. The `%>%` is a tidyverse function that operates similarly to the bash pipe `|`.

Use left_join function and the `%>%` to join the deseq2_res dataframe with the dataframe you just created above. When you manage to do this, your new dataframe should have all of the columns from the DESeq2 output as well as a new column for the gene symbols. Save this to the same dataframe and overwrite it (sorted_padj_res)

```
sorted_padj_res <- sorted_padj_res %>%
  left_join(ids) %>%
  relocate(symbol)
```

```
## Joining with `by = join_by(gene)`
```

```
head(sorted_padj_res, 10)
```

```
## # A tibble: 10 x 8
##    symbol          gene  baseMean log2FoldChange  lfcSE  stat    pvalue      padj
##    <chr>           <chr>    <dbl>          <dbl>  <dbl> <dbl>     <dbl>     <dbl>
##  1 RPS4Y1          ENSG~   10677.          -8.77 0.142  -61.7 0         0
##  2 ENSG000002895~ ENSG~     732.           3.70 0.125   29.7 1.66e-193 1.58e-189
##  3 PNPO            ENSG~     605.          -4.16 0.141  -29.5 2.13e-191 1.35e-187
##  4 PCDHGA10        ENSG~     497.           4.48 0.200   22.4 6.36e-111 3.02e-107
##  5 YPEL3-DT        ENSG~     525.          -2.42 0.115  -21.0 1.03e- 97 3.92e- 94
##  6 LINC02506       ENSG~    1301.          -1.71 0.0986 -17.4 1.17e- 67 3.70e- 64
##  7 SLC2A14         ENSG~     634.           4.69 0.279   16.8 2.51e- 63 6.81e- 60
##  8 ENSG000002863~ ENSG~     907.          -1.71 0.115  -14.9 2.41e- 50 5.71e- 47
##  9 ENSG000002894~ ENSG~     384.          -2.13 0.145  -14.7 3.28e- 49 6.92e- 46
## 10 ENSG000002829~ ENSG~     121.           3.76 0.264   14.2 6.86e- 46 1.30e- 42
```

The most basic of gene set enrichment analyis is to use a webtool like DAVID or EnrichR on a list of significantly differentially expressed genes.

To do this, we need to threshold our list of genes by some metric and we will often choose a padj threshold.

2. A padj threshold and report the number of significant genes at this threshold

A significance threshold of adjusted p-value (padj) < 0.05 was applied to identify differentially expressed genes. Genes with log fold change > 1 were classified as upregulated, while those with log fold change < -1 were considered downregulated. Based on these criteria, 106 genes were upregulated and 177 genes were downregulated. The resulting gene lists were exported as text files (upreg_sig_genes.txt and downreg_sig_genes.txt) for downstream functional enrichment analysis using DAVID or Enrichr, enabling the identification of biological pathways and processes associated with these significant genes.

3. DAVID or ENRICHR analysis and the biological processes

*The Enrichr WikiPathways 2024 Human analysis revealed distinct biological themes among the upregulated and downregulated gene sets. Upregulated genes were significantly enriched in pathways related to cardiac and metabolic functions, including Arrhythmogenic Right Ventricular Cardiomyopathy, Gastric Acid Production, and Enterohepatic Circulation of Bile Acids. These results indicate increased activity in structural remodeling, metabolism, and stress-related signaling upon TYK2 knockout. In contrast, downregulated genes were enriched in pathways related to neural and developmental processes, such as Chronic Hyperglycemia Impairment of Neuron Function, Heart Development, and PI3K-Akt Signaling. This suggests that TYK2 loss suppresses differentiation and neuronal lineage programs, aligning with the observed decrease in endocrine or -cell developmental markers.

4. Fgsea results

The fgsea analysis using the MSigDB C2 canonical pathways identified both up- and downregulated pathways consistent with the Enrichr results, but with additional resolution due to the ranked-based approach. Pathways with positive normalized enrichment scores (NES) included P53 Downstream Signaling, IL-18 Signaling, and Extracellular Matrix Organization, reflecting enhanced cellular stress response, immune signaling, and extracellular remodeling. Pathways with negative NES values, such as Neuronal System, Regulation of -cell Development, and Synaptic Vesicle Pathway, indicated downregulation of neuronal and secretory activity. These results highlight a transcriptional shift from neuroendocrine differentiation toward structural and immune-associated programs.

5.Compare the results from DAVID/ENRICHR analysis and fgsea analysis

Both Enrichr and fgsea analyses converged on a similar biological interpretation—TYK2 knockout enhances stress and extracellular matrix pathways while suppressing neuronal and developmental processes. However, the Enrichr (ORA) method relies on a discrete gene list filtered by significance thresholds, capturing the most strongly affected pathways, whereas fgsea considers all genes ranked by expression change, revealing subtler coordinated effects. As a result, fgsea identified additional immune and stress-related pathways that were not detected by Enrichr, while Enrichr emphasized bile acid and cardiomyopathy pathways specific to highly differentially expressed genes. Together, these complementary approaches provide a comprehensive view of the molecular impact of TYK2 loss.

```r
upreg_sig_genes <- sorted_padj_res %>%
  filter(padj < 0.05 & log2FoldChange > 1) %>%
  select(symbol)
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(symbol)
write.table(downreg_sig_genes, file = "downreg_sig_genes.txt",
            col.names = F, row.names = F, quote = F)
```

```r
downreg_sig_genes_count <- nrow(downreg_sig_genes)

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

```
## [1] "Upregulated Genes: 106"
```

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

```
## [1] "Downregulated Genes: 177"
```

"The blind argument should be set to TRUE for exploratory analysis (e.g., PCA, clustering), and to FALSE for visualization of results where the design is known and desired to be preserved."

```r
rld <- rlog(dds, blind = TRUE)
```

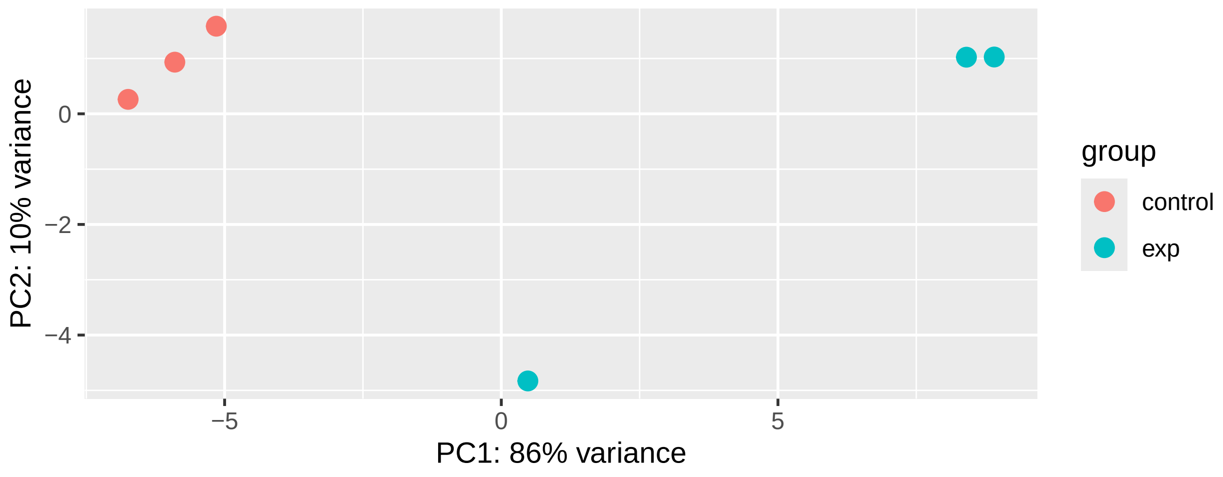
RNAseq Quality Control

PCA plot

- PCA analysis Principal Component Analysis (PCA) was performed on the rlog-transformed count data to assess overall sample variation. PC1 accounted for 86% of the total variance and successfully separated the control and experimental groups, indicating that the main source of variation arises from biological treatment effects rather than technical noise. PC2 explained 10% of the variance and revealed that the three control replicates clustered tightly together, demonstrating excellent reproducibility. The experimental samples also grouped together; however, exp_rep1 showed a slight deviation along PC2, suggesting minor within-group heterogeneity that could reflect biological variability or technical factors such as differences in library concentration or sequencing depth. In summary, the PCA demonstrates strong treatment-specific expression differences and consistent data quality across replicates.

```r
plotPCA(rld, intgroup = "condition")
```

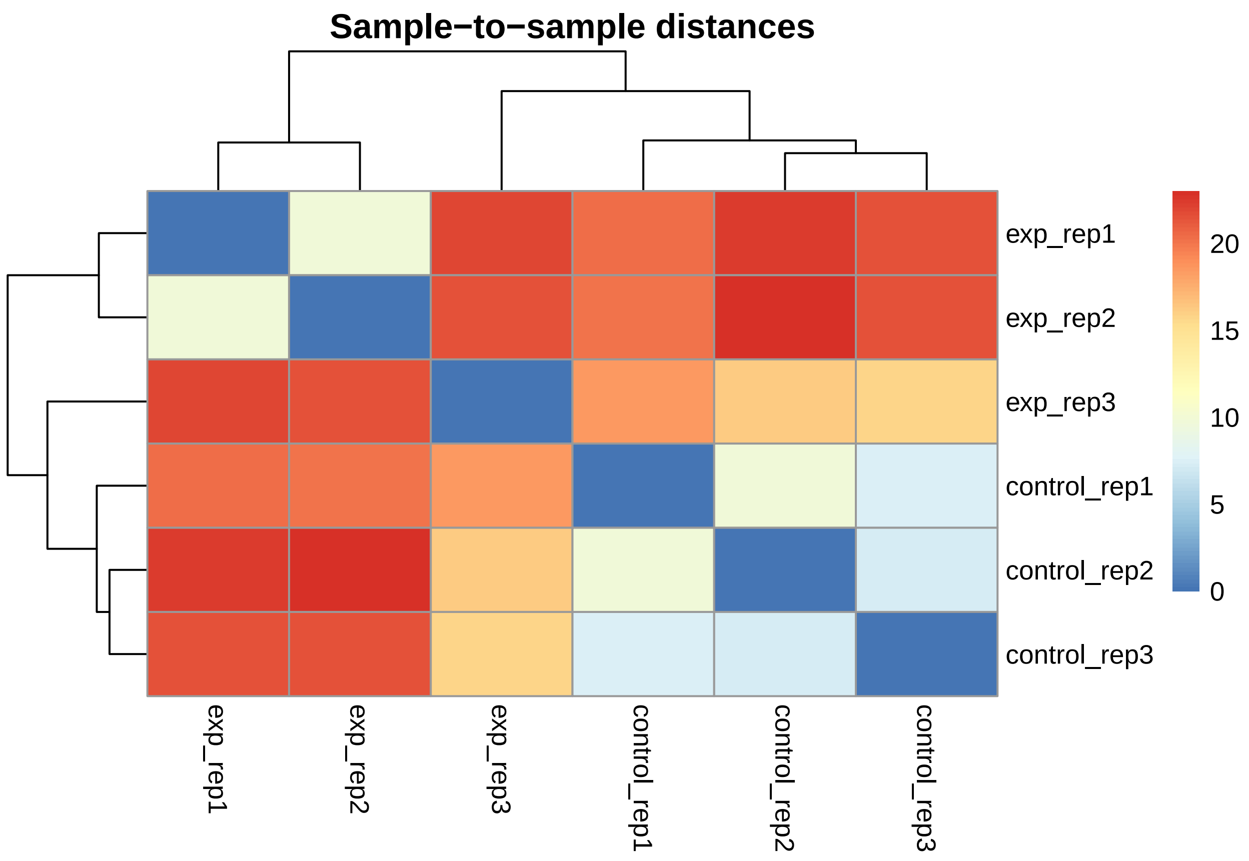
```
## using ntop=500 top features by variance

## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with `aes()`.
## i See also `vignette("ggplot2-in-packages")` for more information.
## i The deprecated feature was likely used in the DESeq2 package.
##   Please report the issue to the authors.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



Sample-to-sample distance matrix

- Sample-to-sample distance heatmap A heatmap of sample-to-sample distances confirmed the PCA results. Biological replicates from the same condition showed small pairwise distances (blue regions), while control and experimental samples displayed larger cross-group distances (orange-red regions), indicating strong condition-driven expression differences. The tight clustering among control samples and the mostly consistent clustering among experimental samples reflect high reproducibility and minimal batch effects. The slightly higher distance for exp_rep1 mirrors its position in the PCA plot, suggesting mild variation but not enough to compromise downstream analyses. These QC results indicate that the RNA-seq data are reliable and suitable for differential expression and pathway analysis

```r
library("pheatmap")
sampleDists <- dist(t(assay(rld)))
pheatmap(as.matrix(sampleDists), main = "Sample-to-sample distances")
```



Performing GSEA (Broad Methodology)

To perform GSEA (broad methodology), we need two main pieces of data: 1. A ranked list of all of the genes discovered in our experiment, 2. A collection of gene sets.

We already have the first element (though we will need to manipulate our data into the right format) and we can obtain the second, a collection of gene sets, from the MSigDB hosted by the Broad.These gene sets are stored in a GMT format, which you can view as a text file. I included one in the repo but you could just as easily download it from the Broad yourself.

Installing fgsea

The fgsea package, which is a re-implementation of the original GSEA algorithm (Broad) with a number of improvements.

```r
library(fgsea)
```

Generated a ranked list of our genes by Log2FoldChange

In order to run GSEA, we need aranked list of genes that have no duplicate values, no NA values, and is ranked by descending log2FoldChange.

Use sorted_padj_res tibble and a combination of the following functions to generate a dataframe (tibble) with the right format: drop_na(), distinct(), arrange() and desc(). When you think you have managed this, save the tibble to a new variable called `deseq2_res_ordered`

Once you have the tibble, use the `setNames` function and the notation, `deseq2_res_ordered$log2FoldChange` and `deseq2_res_ordered$symbol` to create a ranked named list of your genes by log2FoldChange.

```r
deseq2_res_ordered <- sorted_padj_res %>%
  drop_na() %>%
  distinct(symbol, .keep_all = TRUE) %>%
  arrange(desc(log2FoldChange))

rnk_list <- setNames(deseq2_res_ordered$log2FoldChange, deseq2_res_ordered$symbol)
head(rnk_list, 10)
```

```
##          ZNF558          SLC2A14        PCDHGA10 ENSG00000290588 ENSG00000290901
##       12.271037         4.687023        4.482379        4.450196        3.928760
## ENSG00000282914 ENSG00000289575             REN           RCSD1          ECHDC3
##        3.760454         3.699572        3.150114        3.133332        3.116413
```

Read in the GMT file

fgsea provides a convenience function that will read in our gene set collection in the format that it requires to run.

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

```
## $SA_B_CELL_RECEPTOR_COMPLEXES
##  [1] "ATF2"     "BCR"      "BLNK"     "ELK1"     "FOS"      "GRB2"    
##  [7] "HRAS"     "JUN"      "LYN"      "MAP2K1"   "MAP3K1"   "MAPK1"   
## [13] "MAPK3"    "MAPK8IP3" "PAPPA"    "RAC1"     "RPS6KA1"  "RPS6KA3" 
## [19] "SHC1"     "SOS1"     "SYK"      "VAV1"     "VAV2"     "VAV3"    
## 
## $SA_CASPASE_CASCADE
##  [1] "APAF1"  "BIRC2"  "BIRC3"  "CASP10" "CASP3"  "CASP7"  "CASP8"  "CASP9" 
##  [9] "DFFA"   "DFFB"   "FAS"    "FASLG"  "GZMB"   "PARP1"  "PRF1"   "SCAP"  
## [17] "SREBF1" "SREBF2" "XIAP"  
## 
## $SA_FAS_SIGNALING
## [1] "BCL2"    "CASP3"   "CASP8"   "CFL1"    "CFLAR"   "FAS"     "FASLG"  
## [8] "PDE6D"   "S100A10"
## 
## $SA_G1_AND_S_PHASES
##  [1] "ARF1"   "ARF3"   "CCND1"  "CDK2"   "CDK4"   "CDKN1A" "CDKN1B" "CDKN2A"
##  [9] "CFL1"   "E2F1"   "E2F2"   "MDM2"   "NXT1"   "PRB1"   "TP53"  
## 
## $SA_G2_AND_M_PHASES
## [1] "CDC25A" "CDC25B" "CDK1"   "CDK7"   "CDKN1A" "CHEK1"  "NEK1"   "WEE1"
```

```
##
## $SA_MMP_CYTOKINE_CONNECTION
##  [1] "ACE"      "CD44"     "CSF1"     "FCGR3A"   "IL1B"     "IL6R"
##  [7] "SELL"     "SPN"      "TGFB1"    "TGFB2"    "TNF"      "TNFRSF1A"
## [13] "TNFRSF1B" "TNFRSF8"  "TNFSF8"
```

Running FGSEA

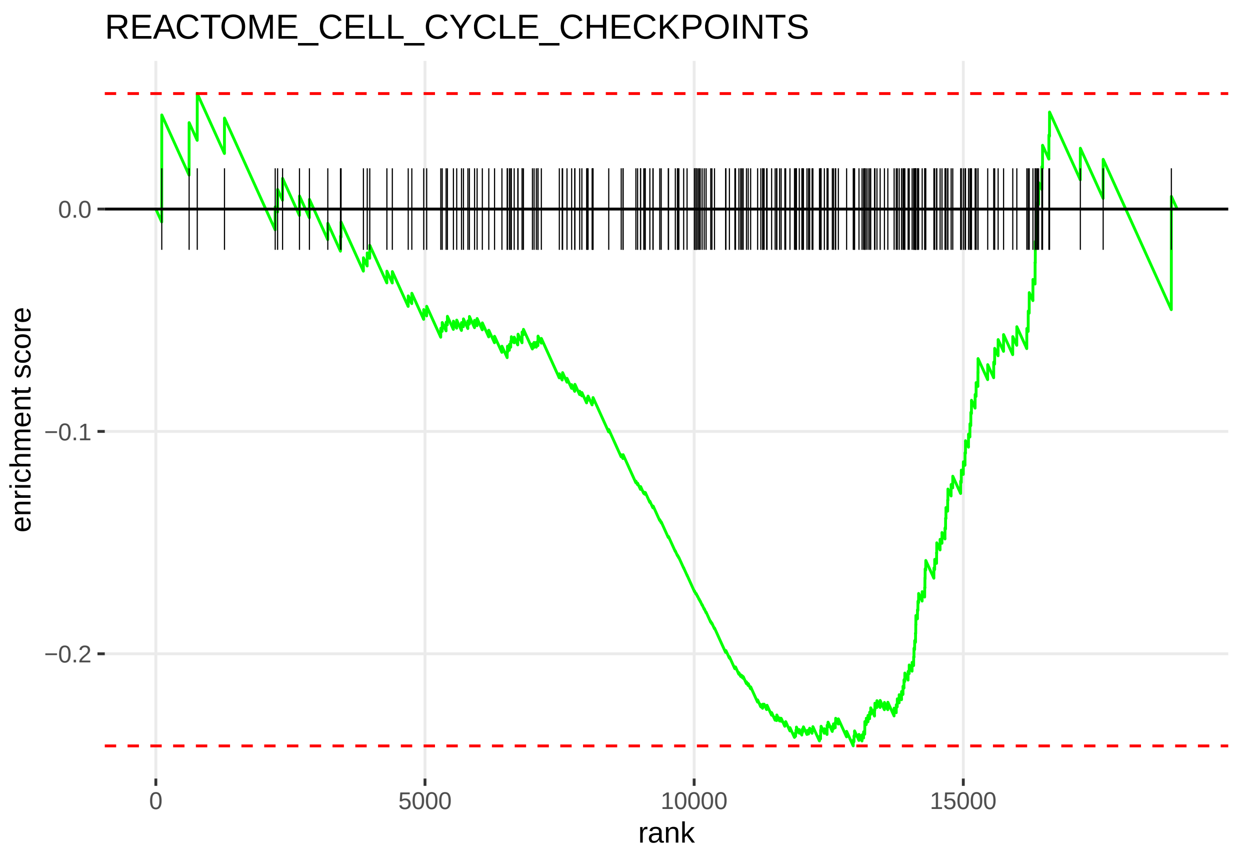
The minSize and maxSize control which gene sets are used based on how many genes are contained within each gene set. It has been empirically found that the GSEA methodology does not perform well on gene sets that are either too small or too large.

```
fgseaRes <- fgsea(pathways, rnk_list, minSize=15, maxSize=500)
```

```
head(fgseaRes[order(pval), ])
```

```
##                                             pathway         pval         padj
##                                              <char>        <num>        <num>
## 1:                         REACTOME_NEURONAL_SYSTEM 8.618829e-11 2.025425e-07
## 2:                   WP_ADHD_AND_AUTISM_ASD_PATHWAYS 8.480353e-08 9.964414e-05
## 3:                       PID_P53_DOWNSTREAM_PATHWAY 4.138482e-07 3.241811e-04
## 4:        REACTOME_EXTRACELLULAR_MATRIX_ORGANIZATION 2.433281e-06 1.429553e-03
## 5: WP_CELL_LINEAGE_MAP_FOR_NEURONAL_DIFFERENTIATION 3.069922e-06 1.442863e-03
## 6:                               WP_IL18_SIGNALING 3.831021e-06 1.500483e-03
##      log2err         ES       NES  size  leadingEdge
##        <num>      <num>     <num> <int>       <list>
## 1: 0.8390889 -0.5355615 -1.968262   342 SLITRK2,....
## 2: 0.7049757 -0.5039704 -1.847048   309 PNPO, CN....
## 3: 0.6749629  0.6188742  2.178011   134 TP63, SP....
## 4: 0.6272567  0.4707280  1.800111   273 LAMA4, M....
## 5: 0.6272567 -0.6131147 -1.999679   101 SLC6A2, ....
## 6: 0.6105269  0.4942781  1.848469   229 MYH6, CO....
```

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

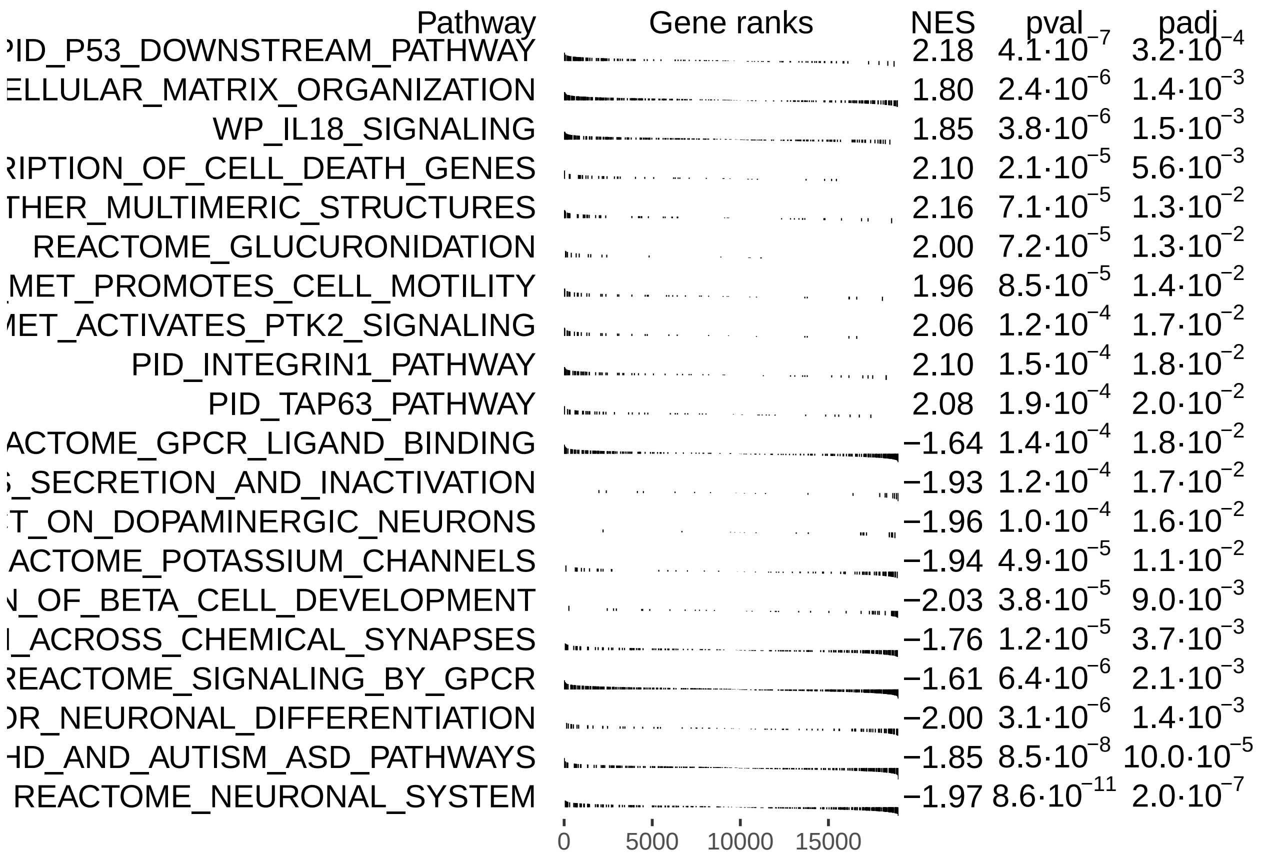


Plotting multiple results

I simply copied this code from their vignette. All it does is grab pathways with a positive and negative enrichment score and order them by p-value.

- fgsea result

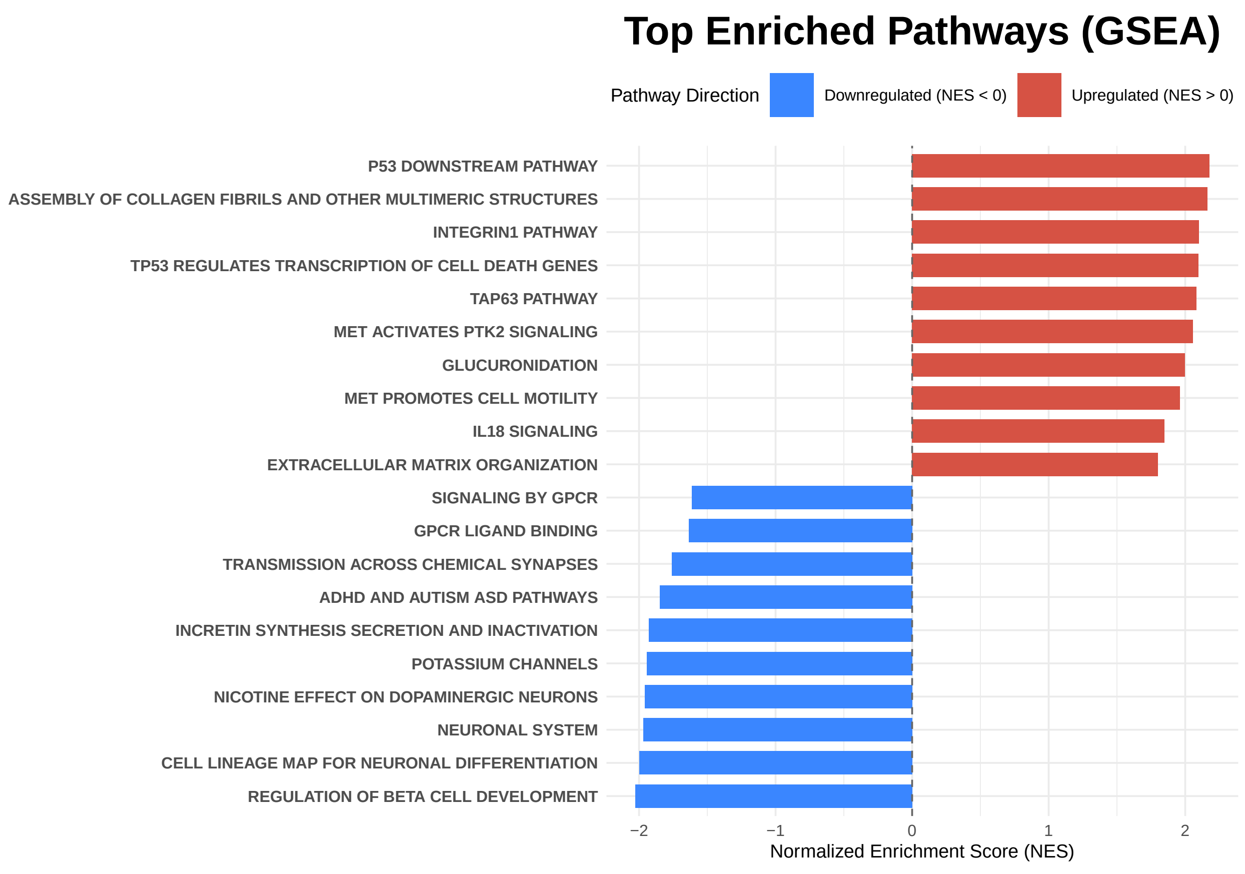
```
topPathwaysUp <- fgseaRes[ES > 0][head(order(pval), n=10), pathway]
topPathwaysDown <- fgseaRes[ES < 0][head(order(pval), n=10), pathway]
topPathways <- c(topPathwaysUp, rev(topPathwaysDown))
plotGseaTable(pathways[topPathways], rnk_list, fgseaRes,
              gseaParam=0.5)
```



Pathway	NES	pval	padj
?ID_P53_DOWNSTREAM_PATHWAY	2.18	$4.1\cdot10^{-7}$	$3.2\cdot10^{-4}$
ELLULAR_MATRIX_ORGANIZATION	1.80	$2.4\cdot10^{-6}$	$1.4\cdot10^{-3}$
WP_IL18_SIGNALING	1.85	$3.8\cdot10^{-6}$	$1.5\cdot10^{-3}$
RIPTION_OF_CELL_DEATH_GENES	2.10	$2.1\cdot10^{-5}$	$5.6\cdot10^{-3}$
THER_MULTIMERIC_STRUCTURES	2.16	$7.1\cdot10^{-5}$	$1.3\cdot10^{-2}$
REACTOME_GLUCURONIDATION	2.00	$7.2\cdot10^{-5}$	$1.3\cdot10^{-2}$
MET_PROMOTES_CELL_MOTILITY	1.96	$8.5\cdot10^{-5}$	$1.4\cdot10^{-2}$
IET_ACTIVATES_PTK2_SIGNALING	2.06	$1.2\cdot10^{-4}$	$1.7\cdot10^{-2}$
PID_INTEGRIN1_PATHWAY	2.10	$1.5\cdot10^{-4}$	$1.8\cdot10^{-2}$
PID_TAP63_PATHWAY	2.08	$1.9\cdot10^{-4}$	$2.0\cdot10^{-2}$
ACTOME_GPCR_LIGAND_BINDING	−1.64	$1.4\cdot10^{-4}$	$1.8\cdot10^{-2}$
;_SECRETION_AND_INACTIVATION	−1.93	$1.2\cdot10^{-4}$	$1.7\cdot10^{-2}$
T_ON_DOPAMINERGIC_NEURONS	−1.96	$1.0\cdot10^{-4}$	$1.6\cdot10^{-2}$
ACTOME_POTASSIUM_CHANNELS	−1.94	$4.9\cdot10^{-5}$	$1.1\cdot10^{-2}$
I_OF_BETA_CELL_DEVELOPMENT	−2.03	$3.8\cdot10^{-5}$	$9.0\cdot10^{-3}$
I_ACROSS_CHEMICAL_SYNAPSES	−1.76	$1.2\cdot10^{-5}$	$3.7\cdot10^{-3}$
REACTOME_SIGNALING_BY_GPCR	−1.61	$6.4\cdot10^{-6}$	$2.1\cdot10^{-3}$
)R_NEURONAL_DIFFERENTIATION	−2.00	$3.1\cdot10^{-6}$	$1.4\cdot10^{-3}$
HD_AND_AUTISM_ASD_PATHWAYS	−1.85	$8.5\cdot10^{-8}$	$10.0\cdot10^{-5}$
REACTOME_NEURONAL_SYSTEM	−1.97	$8.6\cdot10^{-11}$	$2.0\cdot10^{-7}$

```r
library(ggplot2)
library(dplyr)
library(forcats)

# Assume fgseaRes already exists
# Select top 10 upregulated and top 10 downregulated pathways
top_up <- fgseaRes %>%
  filter(NES > 0) %>%
  arrange(padj) %>%
  slice_head(n = 10)

top_down <- fgseaRes %>%
  filter(NES < 0) %>%
  arrange(padj) %>%
  slice_head(n = 10)

# Combine both
top_pathways <- bind_rows(top_up, top_down)

# Clean pathway names (optional)
top_pathways$pathway <- gsub("REACTOME_|WP_|PID_|BIOCARTA_", "", top_pathways$pathway)
top_pathways$pathway <- gsub("_", " ", top_pathways$pathway)

# Order by NES for plotting
top_pathways <- top_pathways %>%
  mutate(pathway = fct_reorder(pathway, NES))
```

```r
# Define color scheme
cols <- c("Upregulated (NES > 0)" = "#D65244",    # red
          "Downregulated (NES < 0)" = "#3A86FF") # blue

# Add direction column
top_pathways <- top_pathways %>%
  mutate(direction = ifelse(NES > 0, "Upregulated (NES > 0)", "Downregulated (NES < 0)"))

# --- Plot ---
ggplot(top_pathways, aes(x = NES, y = pathway, fill = direction)) +
  geom_col(width = 0.7) +
  scale_fill_manual(values = cols) +
  theme_minimal(base_size = 7) +
  labs(
    title = "Top Enriched Pathways (GSEA)",
    x = "Normalized Enrichment Score (NES)",
    y = NULL,
    fill = "Pathway Direction"
  ) +
  geom_vline(xintercept = 0, color = "grey40", linetype = "dashed") +
  theme(
    plot.title = element_text(face = "bold", size = 15, hjust = 0.5),
    axis.text.y = element_text(face = "bold"),
    legend.position = "top"
  )
```



Replicate figure 3C and 3F

- In the original study by Chandra et al. (2022), differential expression analysis at Stage 5 identified 319 upregulated and 412 downregulated genes in TYK2 KO cells compared with wild-type controls. Using the same adjusted p-value threshold (padj < 0.05) but applying an additional fold-change filter (|log FC| 1), our analysis detected 106 upregulated and 177 downregulated genes. The smaller number of DEGs in our dataset likely reflects the more stringent fold-change criterion, differences in normalization methods, and variation between data-processing pipelines. Despite these quantitative differences, both analyses consistently demonstrate the strong transcriptional impact of TYK2 loss at the endocrine progenitor (Stage 5) stage.

- The volcano plot reproduced a similar expression pattern to Figure 3C of the original paper. Genes associated with neuronal differentiation and -cell identity—such as PAX6, NEUROD1, and ONECUT1—were significantly down-regulated, whereas extracellular-matrix-related genes including COL2A1, LAMB2, and SPP1 were up-regulated. These patterns confirm the loss of endocrine lineage commitment and the activation of structural and signaling pathways upon TYK2 knockout. Overall, the distribution and polarity of DEGs between up- and down-regulated clusters align well with the reported transcriptional reprogramming in the original study.

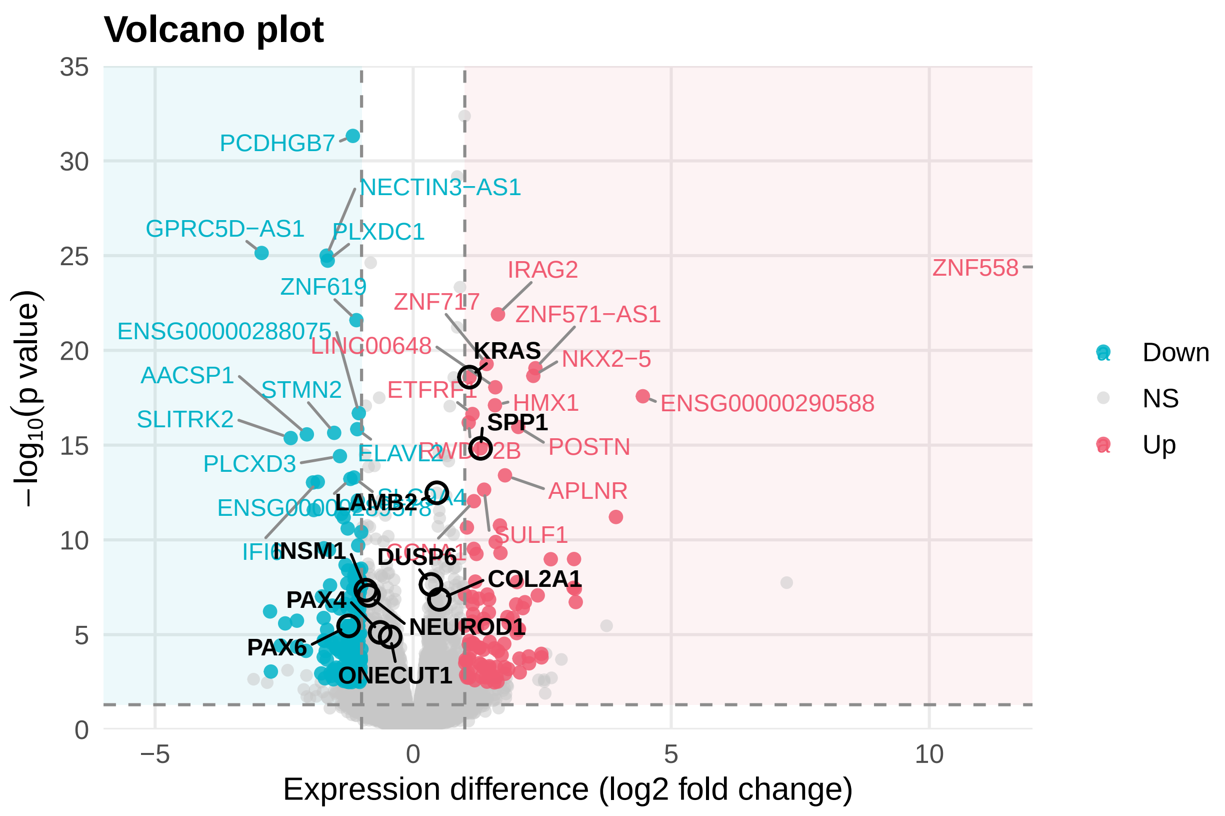
- Compared the enrichment results with those reported by Chandra et al. (2022). Using the MSigDB C2 canonical pathway collection (c2.cp.v2025.1.Hs), the fgsea analysis revealed upregulated pathways related to extracellular matrix organization, integrin signaling, and glucuronidation, while downregulated pathways were enriched for neuronal system and -cell development. These findings closely mirrored the published study, which also identified suppression of neuronal and -cell differentiation programs alongside activation of extracellular matrix and receptor signaling. Minor discrepancies, such as the additional detection of P53 downstream and IL-18 signaling pathways in our results, might arise from methodological differences, including our use of stricter |log FC| 1 thresholds, continuous gene ranking, and integration of fgsea and Enrichr tools, compared with the analysis used in the original paper. Despite these variations, both analyses converge on the conclusion that TYK2 disruption suppresses neuronal differentiation while promoting extracellular matrix remodeling during endocrine progenitor development.

- volcano plot

```r
library(dplyr)
library(ggplot2)
library(ggrepel)
library(scales)
```

```
##
## Attaching package: 'scales'

## The following object is masked from 'package:purrr':
##
##     discard

## The following object is masked from 'package:readr':
##
##     col_factor
```

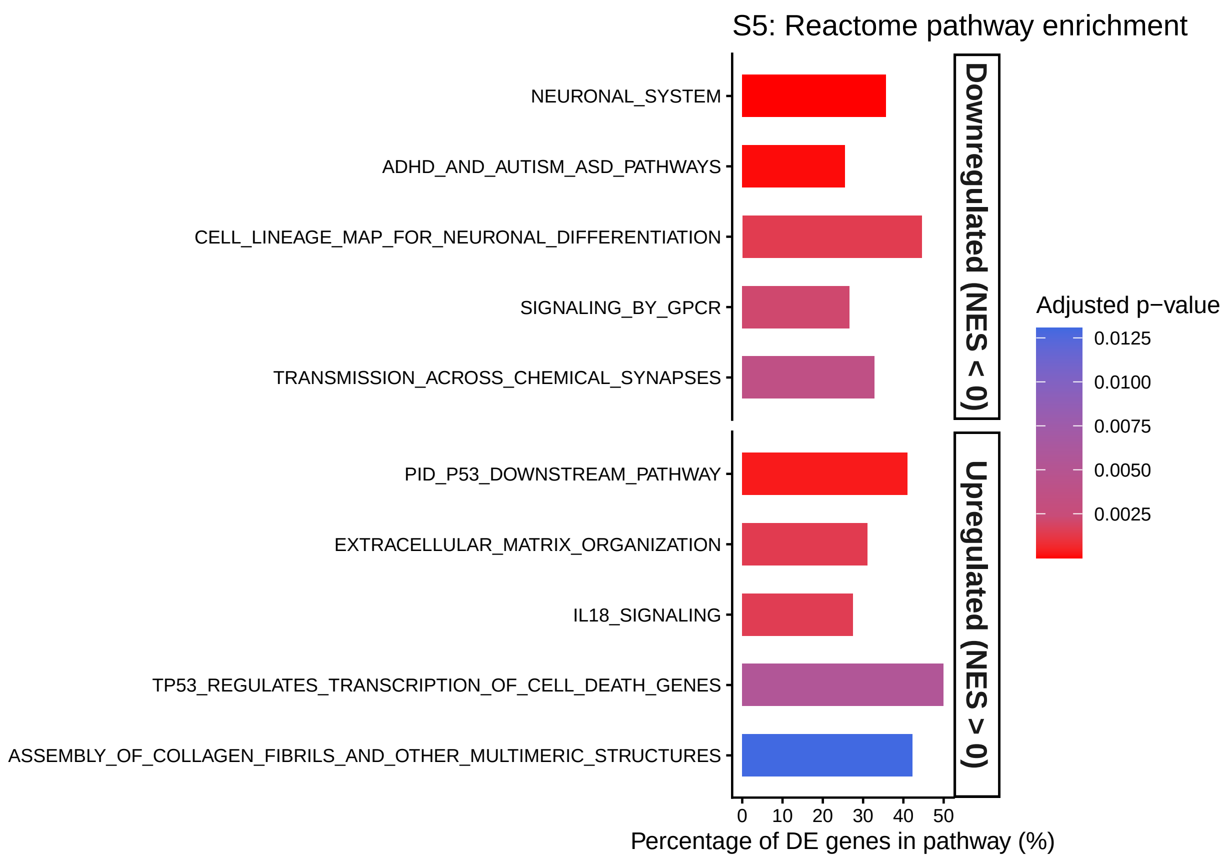
```r
res <- sorted_padj_res %>%
  mutate(
    negLog10P = -log10(pvalue),
    status = case_when(
      padj < 0.05 & log2FoldChange >=  1 ~ "Up",
      padj < 0.05 & log2FoldChange <= -1 ~ "Down",
      TRUE ~ "NS"
    )
  )%>%
  filter(negLog10P <=35)

force_genes <- c("INSM1","NEUROD1","ONECUT1","PAX4","PAX6",
                 "DUSP6","KRAS","SPP1","LAMB2","COL2A1")

force_df <- res %>%
  mutate(symbol_upper = toupper(symbol)) %>%
  filter(symbol_upper %in% toupper(force_genes)) %>%
  distinct(symbol, .keep_all = TRUE)

n_label <- 14
lab_up   <- res %>% filter(status == "Up")   %>% arrange(padj) %>%
  filter(!(symbol %in% force_df$symbol)) %>% slice_head(n = n_label)
lab_down <- res %>% filter(status == "Down") %>% arrange(padj) %>%
  filter(!(symbol %in% force_df$symbol)) %>% slice_head(n = n_label)
labs_df  <- bind_rows(lab_up, lab_down)


x_min <- -6; x_max <- 12
y_max <- 35

cols <- c("Down"="#02B3C8", "Up"="#F05A71", "NS"="grey78")

alpha_zone <- 0.07
thr_x <- 1
thr_y <- -log10(0.05)

p_volcano <- ggplot(res, aes(x = log2FoldChange, y = negLog10P)) +
  annotate("rect",
           xmin = thr_x, xmax = Inf, ymin = thr_y, ymax = Inf,
           fill = "#F05A71", alpha = alpha_zone) +
  annotate("rect",
           xmin = -Inf, xmax = -thr_x, ymin = thr_y, ymax = Inf,
           fill = "#02B3C8", alpha = alpha_zone) +

  geom_point(data = subset(res, status == "NS"),
             aes(color = status), size = 1.5, alpha = 0.5) +
  geom_point(data = subset(res, status != "NS"),
             aes(color = status), size = 1.8, alpha = 0.85) +

  geom_vline(xintercept = c(-thr_x, thr_x), linetype = "dashed", color = "grey55") +
  geom_hline(yintercept = thr_y, linetype   = "dashed", color = "grey55") +

  geom_text_repel(
    data = labs_df,
    aes(label = symbol, color = status),
    size = 3.0, max.overlaps = 100,
    box.padding = 0.35, point.padding = 0.25,
    segment.color = "grey55", min.segment.length = 0
  ) +

  geom_point(data = force_df,
             aes(x = log2FoldChange, y = negLog10P),
             shape = 21, size = 3.0, stroke = 1.0, color = "black", fill = NA) +
  geom_text_repel(
    data = force_df,
    aes(label = symbol),
    size = 3.2, fontface = "bold",
    box.padding = 0.40, point.padding = 0.30,
    segment.color = "black", max.overlaps = Inf, force = 6,
    min.segment.length = 0
  ) +

  scale_color_manual(values = cols) +
  coord_cartesian(xlim = c(x_min, x_max), ylim = c(0, y_max), expand = 0) +
  scale_x_continuous(breaks = pretty(c(x_min, x_max), n = 6)) +
  scale_y_continuous(breaks = pretty(c(0, y_max), n = 6)) +

  labs(
    title = "Volcano plot",
    x = "Expression difference (log2 fold change)",
    y = expression(-log[10](p~value)),
    color = NULL,
    caption = "Thresholds: |log2FC|   1 & padj < 0.05 (dashed lines). Shaded = significant zones."
  ) +
  theme_minimal(base_size = 12) +
  theme(
    legend.position = "right",
    panel.grid.minor = element_blank(),
    plot.title = element_text(face = "bold"),
    plot.caption = element_text(color = "grey40")
  )

p_volcano
```



Thresholds: |log2FC| >= 1 & padj < 0.05 (dashed lines). Shaded = significant zones.

- enrichment results

```r
library(dplyr)
library(ggplot2)
library(forcats)
library(scales)

enr <- fgseaRes %>%
  mutate(
    leading_n = vapply(leadingEdge, length, integer(1)),
    percent   = 100 * leading_n / size,
    direction = if_else(NES >= 0, "Upregulated (NES > 0)", "Downregulated (NES < 0)"),
    pathway = gsub("REACTOME_|WP_", "", pathway)
  )

# top5 significant pathways
top_n <- 5
top_up <- enr %>% filter(NES > 0)  %>% arrange(padj) %>% slice_head(n = top_n)
top_dn <- enr %>% filter(NES < 0)  %>% arrange(padj) %>% slice_head(n = top_n)
enr_top <- bind_rows(top_up, top_dn) %>%
  group_by(direction) %>%
  mutate(pathway = fct_reorder(pathway, -log10(padj))) %>%  #
  ungroup()

# plot
ggplot(enr_top, aes(x = percent, y = pathway, fill = padj)) +
  geom_col(width = 0.6) +
  facet_grid(rows = vars(direction), scales = "free_y", space = "free_y") +
  scale_fill_gradientn(
    colours = c("#FF0000", "#4169E1"),
    values = rescale(c(0.001, 0.01, 0.05)),
    name = "Adjusted p-value"
  ) +
  labs(
    title = "S5: Reactome pathway enrichment",
    x = "Percentage of DE genes in pathway (%)",
    y = NULL
  ) +
  theme_classic(base_size = 9) +
  theme(
    strip.text.y = element_text(face = "bold", size = 11),
    legend.position = "right"
  )
```



SessionInfo

The sessioninfo function is one way we can attempt to be transparent about our environment management in R.

This will print out a summary of the current packages installed in the environment.

```r
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 [1] CRAN (R 4.4.0)
## Biobase              * 2.66.0   2024-10-29 [1] Bioconductor 3.20 (R 4.4.0)
## BiocGenerics         * 0.52.0   2024-10-29 [1] Bioconductor 3.20 (R 4.4.0)
## BiocManager            1.30.26  2025-06-05 [1] CRAN (R 4.4.0)
## BiocParallel           1.40.2   2025-10-06 [1] Bioconductor
## bit                    4.6.0    2025-03-06 [1] CRAN (R 4.4.0)
## bit64                  4.6.0-1  2025-01-16 [1] CRAN (R 4.4.0)
## cli                    3.6.5    2025-04-23 [1] CRAN (R 4.4.0)
## codetools              0.2-20   2024-03-31 [2] CRAN (R 4.4.3)
## cowplot                1.2.0    2025-07-07 [1] CRAN (R 4.4.0)
## crayon                 1.5.3    2024-06-20 [1] CRAN (R 4.4.0)
## data.table             1.17.8   2025-07-10 [1] CRAN (R 4.4.0)
## DelayedArray           0.32.0   2024-10-29 [2] Bioconductor 3.20 (R 4.4.3)
## DESeq2               * 1.46.0   2024-10-29 [1] Bioconductor 3.20 (R 4.4.0)
## dichromat              2.0-0.1  2022-05-02 [2] CRAN (R 4.4.3)
## digest                 0.6.37   2024-08-19 [1] CRAN (R 4.4.0)
## dplyr                * 1.1.4    2023-11-17 [2] CRAN (R 4.4.3)
## evaluate               1.0.5    2025-08-27 [1] CRAN (R 4.4.0)
## farver                 2.1.2    2024-05-13 [1] CRAN (R 4.4.0)
## fastmap                1.2.0    2024-05-15 [1] CRAN (R 4.4.0)
## fastmatch              1.1-6    2024-12-23 [1] CRAN (R 4.4.0)
## fgsea                * 1.32.4   2025-03-20 [1] Bioconductor 3.20 (R 4.4.0)
## forcats              * 1.0.1    2025-09-25 [1] CRAN (R 4.4.0)
## generics               0.1.4    2025-05-09 [1] CRAN (R 4.4.0)
## GenomeInfoDb         * 1.42.3   2025-01-27 [1] Bioconductor 3.20 (R 4.4.0)
## GenomeInfoDbData       1.2.13   2025-10-06 [1] Bioconductor
## GenomicRanges        * 1.58.0   2024-10-29 [1] Bioconductor 3.20 (R 4.4.0)
## ggplot2              * 4.0.0    2025-09-11 [1] CRAN (R 4.4.0)
## ggrepel              * 0.9.6    2024-09-07 [1] CRAN (R 4.4.0)
## glue                   1.8.0    2024-09-30 [1] CRAN (R 4.4.0)
## gtable                 0.3.6    2024-10-25 [1] CRAN (R 4.4.0)
## 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               2.0.0    2025-03-27 [1] CRAN (R 4.4.0)
## knitr                  1.50     2025-03-16 [1] CRAN (R 4.4.0)
## labeling               0.4.3    2023-08-29 [2] CRAN (R 4.4.3)
## lattice                0.22-7   2025-04-02 [1] CRAN (R 4.4.0)
## lifecycle              1.0.4    2023-11-07 [2] CRAN (R 4.4.3)
##  locfit                 1.5-9.12 2025-03-05 [1] CRAN (R 4.4.0)
##  lubridate            * 1.9.4    2024-12-08 [1] CRAN (R 4.4.0)
##  magrittr               2.0.4    2025-09-12 [1] CRAN (R 4.4.0)
##  Matrix                 1.7-4    2025-08-28 [1] CRAN (R 4.4.0)
##  MatrixGenerics       * 1.18.1   2025-01-09 [2] Bioconductor 3.20 (R 4.4.3)
##  matrixStats          * 1.5.0    2025-01-07 [1] CRAN (R 4.4.0)
##  pheatmap             * 1.0.13   2025-06-05 [1] CRAN (R 4.4.0)
##  pillar                 1.11.1   2025-09-17 [1] CRAN (R 4.4.0)
##  pkgconfig              2.0.3    2019-09-22 [2] CRAN (R 4.4.3)
##  purrr                * 1.1.0    2025-07-10 [1] CRAN (R 4.4.0)
##  R6                     2.6.1    2025-02-15 [1] CRAN (R 4.4.0)
##  RColorBrewer           1.1-3    2022-04-03 [2] CRAN (R 4.4.3)
##  Rcpp                   1.1.0    2025-07-02 [1] CRAN (R 4.4.0)
##  readr                * 2.1.5    2024-01-10 [2] CRAN (R 4.4.3)
##  rlang                  1.1.6    2025-04-11 [1] CRAN (R 4.4.0)
##  rmarkdown              2.30     2025-09-28 [1] CRAN (R 4.4.0)
##  rstudioapi             0.17.1   2024-10-22 [1] CRAN (R 4.4.0)
##  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)
##  S7                     0.2.0    2024-11-07 [1] CRAN (R 4.4.0)
##  scales               * 1.4.0    2025-04-24 [1] CRAN (R 4.4.0)
##  sessioninfo            1.2.3    2025-02-05 [1] CRAN (R 4.4.0)
##  SparseArray            1.6.2    2025-02-20 [2] Bioconductor 3.20 (R 4.4.3)
##  stringi                1.8.7    2025-03-27 [1] CRAN (R 4.4.0)
##  stringr              * 1.5.2    2025-09-08 [1] CRAN (R 4.4.0)
##  SummarizedExperiment * 1.36.0   2024-10-29 [2] Bioconductor 3.20 (R 4.4.3)
##  tibble               * 3.3.0    2025-06-08 [1] CRAN (R 4.4.0)
##  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                0.57     2025-04-15 [1] CRAN (R 4.4.0)
##  tzdb                   0.5.0    2025-03-15 [1] CRAN (R 4.4.0)
##  UCSC.utils             1.2.0    2024-10-29 [2] Bioconductor 3.20 (R 4.4.3)
##  utf8                   1.2.6    2025-06-08 [1] CRAN (R 4.4.0)
##  vctrs                  0.6.5    2023-12-01 [2] CRAN (R 4.4.3)
##  vroom                  1.6.6    2025-09-19 [1] CRAN (R 4.4.0)
##  withr                  3.0.2    2024-10-28 [1] CRAN (R 4.4.0)
##  xfun                   0.53     2025-08-19 [1] CRAN (R 4.4.0)
##  XVector                0.46.0   2024-10-29 [2] Bioconductor 3.20 (R 4.4.3)
##  yaml                   2.3.10   2024-07-26 [1] CRAN (R 4.4.0)
##  zlibbioc               1.52.0   2024-10-29 [2] Bioconductor 3.20 (R 4.4.3)
##
##  [1] /usr4/bf527/shuwenyu/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.
##
## ------------------------------------------------------------------------------
```