Pipeline for RNA sequencing data analysis by combination of Nextflow and R

Jia-Hua Qu, MD, PhD *

Department of Pathology, University of California San Francisco, San Francisco, CA 94143, USA

Abstract

With the development of high-throughput technologies, RNA sequencing (RNA-seq) has become a widely used technology in biological studies and thus a large number of RNA-seq data are emerging and remain to be analyzed. Although there are many different options for analysis methods and tools, a unified pipeline for RNA-seq data analysis is always necessary for a laboratory. Given the update of new methods and tools, I summarized my customized analysis codes to generate an updated pipeline for RNA-seq data analysis. During aging, gene mutations are accumulated, and hormone regulation is disrupted, which may exacerbate age-related diseases in return. Therefore, we generated a dataset from mice with a gene mutation or not and under different hormone treatments to study the effects of two factors, i.e., hormone and gene mutation, on the transcriptome. Based on the Nextflow nf-core rnaseq pipeline, I established my pipeline consisting of three stages: (1) upstream analysis containing quality control of fastq files before and after trimming, trimming, alignment, and quantification; (2) midstream analysis containing count normalization, differentially expressed genes analysis, and visualization via boxplot, PCA, t-SNE, sample distance heatmap, MA plot, volcano plot, and gene expression heatmap; and (3) downstream analysis containing functional enrichments of KEGG pathways and GO terms. Results showed distinct effects of the single factor as well as interactive effects of the two factors. Codes are also provided for readers who want to customize their analysis pipeline adapted from this pipeline easily.

Introduction

In the field of modern biological studies, omics are widely used, and thus high-throughput data are generated every day in need of customized analysis. There are many different options for analysis methods and tools. They can be classified into two major types: graphical user interface and command line interface. The former can be divided into website applications and local computer applications; the latter can be divided into programs written in different programming languages, such as Bash, R, Python, Perl, Groovy, etc.

The graphical user interface is convenient for a broad range of scientists who want to make full of omics but don't have programming skills to analyze high-throughput data, whereas the command line interface is more flexible in customizing analysis and helps

save time in repeated procedures. Website applications are usually updated more frequently than local computer applications, so website applications often provide access to the latest version of databases, but they may be unstable. In contrast, local computer applications are more stable and can be used to reproduce results better than website applications. Partek Flow is a representative website application with a graphical user interface, and it provides convenience in omics data analysis, particularly useful in single-cell RNA-seq data analysis[1]. JMP and GraphPad Prism are two well-used local computer applications with graphical user interfaces with statistical functions as well[2].

Despite the advantages and convenience of the graphical user interface, the command line interface is preferred by professional bioinformaticians because of its more powerful functions which can also be extendible and scalable. For example, fastq files of a larger number of samples are usually analyzed in the Linux system, because the Linux system can be used to process high-throughput data in parallel and fast[3]. R and Python are more and more popular because both are open-sourced and free[4]. A lot of R packages and Python libraries are available and provide conveniences for bioinformatics analysis.

In addition, there are some tools invented especially for building pipelines, such as Nextflow[5], Snakemake[6], WDL and Cromwell[7], etc. For example, Nextflow is well-used, because it enables scalable and reproducible scientific workflows and allows the adaptation of pipelines written in the most common scripting languages. In addition, Nextflow has a very active user community which released many curated nf-core pipelines (https://nf-co.re/)[8] that can be modified for customized use.

Given a typical pipeline for RNA sequencing (RNA-seq) data analysis consists of multiple steps, I divided it into three stages: (1) upstream data process; (2) midstream visualization of count matrix; and (3) downstream functional analysis. Different tools have different advantages at different stages and different sets of data require customized analysis, so the combination of different tools is a better choice in a practical project.

Because of the advantages of different tools, the combination of various tools allows us to customize our bioinformatics analysis flexibly[9]. My pipeline was built by the combination of Nextflow nf-core rnaseq pipeline and R with many R packages. During aging, gene mutations are accumulated, and hormone regulation is disrupted, which may exacerbate age-related diseases in return. I applied this pipeline to our in-house dataset that was generated from mice with a gene mutation or not and under different hormone treatments. My results showed distinct effects of a single factor as well as interactive effects of the two factors, i.e., hormone and gene mutation, on mouse transcriptome. This pipeline can also be applied in other RNA-seq data analyses in the future.

Tools

Linux system

Wynton HPC (https://wynton.ucsf.edu/hpc/index.html) - UCSF Research Computing at Scale, based upon the CentOS 7.

Pipeline tool

The nf-core/rnaseq pipeline (https://nf-co.re/rnaseq/3.10.1)[8], based upon Nextflow (https://www.nextflow.io/)[5].

R

R version: 4.1.3 or 4.2.2
RStudio version: 2023.09.0 Build 463
R packages versions:

```
1. package.version("dplyr")   # "1.1.0"[10]
2. package.version("tidyr")   # "1.3.0"[10]
3. package.version("DESeq2")   # "1.34.0"[11]
4. package.version("tibble")   # "3.1.8"[10]
5. package.version("ggplot2")   # "3.4.1"[10]
6. package.version("RColorBrewer")   # "1.1-3"
7. package.version("pheatmap")   # "1.0.12"
8. package.version("Rtsne")   # "0.16"[12, 13]
9. package.version("readr")   # "2.1.4"[10]
10. package.version("clusterProfiler")   # "4.2.2"[14]
11. package.version("org.Mm.eg.db")   # "3.14.0"
12. package.version("EnhancedVolcano")   # "1.12.0"
```

Dataset

Here, I used our in-house RNA-seq data as a demo. Sequencing type: SE. Library Kit: Nugen Universal Plus. Machine: HiSeq4000. Because our in-house RNA-seq dataset is unpublished and the application of this pipeline is not restricted to this dataset, the detailed meta information is hidden. In brief, this dataset was obtained from mice with a gene mutation (mut) and their wild-type littermates (con), both of whom received treatment of two hormones singly (E2, P4) or combinedly (E2P4) or placebo (Plac). In total, this dataset contains eight groups of samples: Plac_con, Plac_mut, E2_con, E2_mut, P4_con, P4_mut, E2P4_con, and E2P4_mut. This project did not intend to describe the gene mutation within this dataset or to explain the definition of different hormone treatments explicitly. When we apply this pipeline in other projects, mut can represent any other gene mutation, and E2 or P4 can represent any other internal or external stimuli or disease states.

Workflow

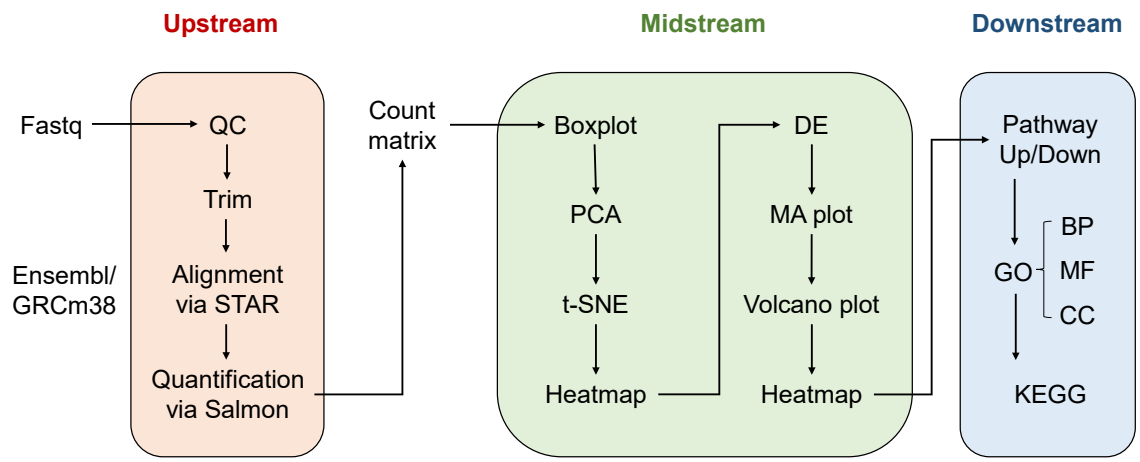


Figure 1. Workflow of RNA-seq data analysis. Upstream analysis of fastq files via Nextflow, midstream analysis of count matrix and differentially expressed genes in R, and downstream analysis of functional enrichment.

Based on the Nextflow nf-core rnaseq pipeline, I established my pipeline (Figure 1) consisting of three stages: (1) upstream analysis containing quality control of fastq files before and after trimming, trimming, alignment, and quantification; (2) midstream analysis containing count normalization, boxplot, PCA, t-SNE, sample distance heatmap, differentially expressed genes analysis, MA plot, volcano plot, and gene expression heatmap; and (3) downstream analysis containing functional enrichments of KEGG pathways and GO terms.

Upstream analysis with Bash and Nextflow

1. Create a conda environment.

The virtual environment technique is an ideal way to manage systems and software, so I recommended creating a virtual environment via conda (https://docs.conda.io/en/latest/).

```
cd ~
mkdir 01-Nextflow
cd ~/01-Nextflow
module load Sali anaconda
conda create -n NextFlow-nf-core
source activate NextFlow-nf-core
python -V     # Python 3.8.10
```

2. Install Nextflow, nf-core, and rnaseq pipeline.

Singularity is a scientific container for the mobility of comput[15], so I recommend downloading a singularity image of nf-core/rnaseq program to reproduce the analysis.

```
# install nextflow
conda install nextflow -y
conda list | grep nextflow   # version: 22.10.6
```

```
# install nf-core
conda search nf-core
conda install nf-core -y
conda list | grep nf-core   # 2.7.2

# install rnaseq pipeline
nf-core download nf-core/rnaseq
# select version: 3.10.1
# download singularity
```

3. Prepare the reference genome.

Because the data were obtained from mice, the GRCm38 reference genome and its accompanied annotation were downloaded from iGenome (https://support.illumina.com/sequencing/sequencing_software/igenome.html) and used in this project.

```
cd ~/01-Nextflow
mkdir -p reference_genome/mouse/Ensembl/GRCm38
cd reference_genome/mouse/Ensembl/GRCm38
nohup aws s3 --no-sign-request --region eu-west-1 sync s3://ngi-igenomes/igenomes/Mus_musculus/Ensembl/GRCm38/Annotation/Genes/ . --exclude "*" --include "genes.gtf" &
nohup aws s3 --no-sign-request --region eu-west-1 sync s3://ngi-igenomes/igenomes/Mus_musculus/Ensembl/GRCm38/Sequence/WholeGenomeFasta/ ./ &
```

4. Prepare fastq files.

```
cd ~/01-Nextflow
mkdir data
```

Copy fastq files that need to be analyzed into the folder ~/01-Nextflow/data.

5. Make a sample sheet.

Download the python script to make a sample sheet for the subsequent analysis.

```
cd ~/01-Nextflow/data
wget -L https://raw.githubusercontent.com/nf-core/rnaseq/master/bin/fastq_dir_to_samplesheet.py
chmod a+x fastq_dir_to_samplesheet.py
./fastq_dir_to_samplesheet.py -r1 _R1_001.fastq.gz ~/data/fastq samplesheet.csv
```

6. Customize a config file.

```
cd ~/01-Nextflow
mkdir analysis
mkdir output
cd ~/01-Nextflow/analysis
vim nextflow.config
```

Customized analysis includes:

- FastQC – Raw read QC
- TrimGalore – Adapter and quality trimming
- STAR – Alignment
- Salmon – Quantification
- RSeQC – Various RNA-seq QC metrics
- DESeq2 – PCA plot and sample pairwise distance heatmap and dendrogram
- MultiQC – Present QC for raw reads, alignment, read counting and sample similarity

In addition, in the config file, we specified the paths to the input sample sheet of fastq files to be analyzed, reference genome files, and folder to save the output.

```
// Global default params, used in configs
params {

    // Input options
    input = '~/data/fastq/samplesheet.csv'

    // References
    fasta = '~/reference_genome/mouse/Ensembl/GRCm38/genome.fa'
    gtf = '~/reference_genome/mouse/Ensembl/GRCm38/genes.gtf'
    gencode = false
    save_reference = true

    // QC
    skip_bigwig = true
    skip_stringtie = true
    skip_preseq = true
    skip_qualimap = true
    deseq2_vst = true

    // Boilerplate options
    outdir = '~/output'
    email = "jiahua.qu@hotmail.com"

    // Max resource options
    // Defaults only, expecting to be overwritten
    max_memory = '500.GB'
    max_cpus = 12
    max_time = '240.h'
}
```

7. Make a bash script.

```
cd ~/01-Nextflow/analysis
vim my_script.sh
```

In the bash script, we specified: (1) required computation resources including core number, memory and running time; (2) paths to input files including log file that records

progress, Nextflow workflow program, singularity profile, and config file with customized settings.

```
#! /usr/bin/env bash
#
#$ -S /bin/bash
#$ -cwd
#$ -r y
#$ -j y
#$ -pe smp 6
#$ -l mem_free=45G
#$ -l scratch=270G
#$ -l h_rt=24:00:00

module load Sali anaconda
source activate NextFlow-nf-core

# set path
log=~/01-Nextflow/analysis/my.log
workflow=~/01-Nextflow/nf-core-rnaseq-3.10.1/workflow
profile=singularity
config=~/01-Nextflow/analysis/nextflow.config

# run pipeline
nextflow -log $log run $workflow -profile $profile -resume -c $config

conda deactivate

[[ -n "$JOB_ID" ]] && qstat -j "$JOB_ID"
```

8. Submit a job to run in the background.

```
qsub my_script.sh
```

9. Output.

The output contains two major parts: (1) star_salmon output including count matrix and metadata, and (2) MultiQC report showing quality control of fastq files before and after trimming (Figure 2).

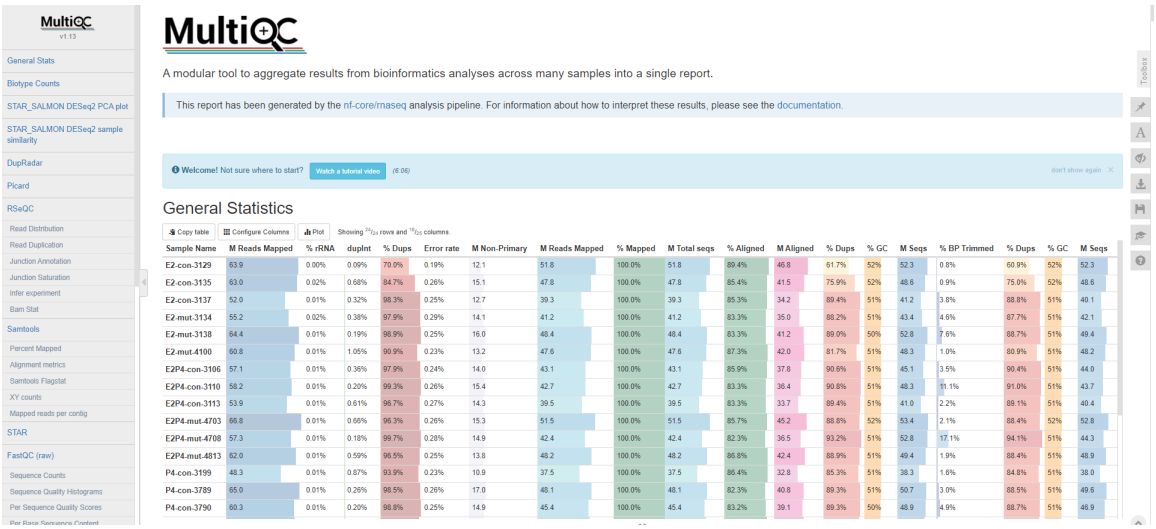


Figure 2. Screenshot of MultiQC report. The report summarizes quality control results of fastq files before and after each process.

Midstream analysis with R

10. Read in rds object generated in the last step.

```
### create a folder
setwd("~/")
if (!file.exists("02-QC")){
  dir.create("02-QC")
}
setwd("~/02-QC")

### read in rds
# use the salmon.merged.gene_counts.rds
gene_counts <- readRDS("~/01-Nextflow/output/star_salmon/salmon.merged.gene_counts.rds")

# extract and process count matrix
counts <- gene_counts@assays@data$counts
# round non-integer to integer for loading into DESeq2
counts_round <- round(counts, digits = 0)
# remove rows whose values are all zero
library(dplyr)
counts_round <- counts_round %>%
  filter(rowSums(.) > 0)

# extract meta_data
library(tidyr)
meta_data <- data.frame(sample=gene_counts@colData@rownames) %>%
```

```
  separate(sample, into=c("treatment","genotype","num"), sep = "\\.", remove = FALSE) %>%
  unite(col = "condition", treatment,genotype, remove = FALSE)
meta_data$condition <- factor(meta_data$condition)
meta_data$treatment <- factor(meta_data$treatment)
meta_data$genotype <- factor(meta_data$genotype, levels = c("con", "mut"))
saveRDS(meta_data, "meta_data.rds", compress = T)
```

11. Generate a dds object with the DESeq2 R package.

```
library(DESeq2)
# Load in Salmon output
dds_new <- DESeqDataSetFromMatrix(countData = counts_round,
                                  colData = meta_data,
                                  design= ~ condition)

# Normalization
dds_new <- estimateSizeFactors(dds_new)
dds_rlog <- rlog(dds_new)
dds_vst <- varianceStabilizingTransformation(dds_new)
assay(dds_new, "rlog") <- assay(dds_rlog)
assay(dds_new, "vst") <- assay(dds_vst)
saveRDS(dds_new, "dds_new.rds", compress = T)
saveRDS(dds_rlog, "dds_rlog.rds", compress = T)
saveRDS(dds_vst, "dds_vst.rds", compress = T)
```

12. Plot to visualize the read counts.

```
### create a folder
setwd("~/02-QC")
if (!file.exists("QCplots")){
  dir.create("QCplots")
}

### 1. boxplot
library(DESeq2)
## 1.1. counts_vst
counts_vst <- dds_new@assays@data$vst

library(tibble)
dat <- counts_vst %>%
  as.data.frame() %>%
  rownames_to_column(var="gene") %>%
  gather(key = "sample", value = "counts_vst", -gene) %>%
  left_join(meta_data, by="sample")

library(ggplot2)
```

```
p_1 <- ggplot(data=dat,aes(x=sample,y=counts_vst,fill=condition)) +
  geom_boxplot()+
  labs(x="Sample",y="VST Normalized Read Counts")+
  ggtitle("VST Normalized Read Count by Sample") +
  theme(axis.text.x=element_text(angle = 90, vjust = 0.5, hjust = 1))
ggsave(filename = "QCplots/boxplot-counts_vst.pdf", plot = p_1, width = 10, height = 5)

## 1.2. counts_rlog
counts_rlog <- dds_new@assays@data$rlog

library(tibble)
dat <- counts_rlog %>%
  as.data.frame() %>%
  rownames_to_column(var="gene") %>%
  gather(key = "sample", value = "counts_rlog", -gene) %>%
  left_join(meta_data, by="sample")

library(ggplot2)
p_2 <- ggplot(data=dat,aes(x=sample,y=counts_rlog,fill=condition)) +
  geom_boxplot()+
  labs(x="Sample",y="rlog Normalized Read Counts")+
  ggtitle("rlog Normalized Read Count by Sample") +
  theme(axis.text.x=element_text(angle = 90, vjust = 0.5, hjust = 1))
ggsave(filename = "QCplots/boxplot-counts_rlog.pdf", plot = p_2, width = 10, height = 5)

### 2. sample correlation heatmap
sampleDists <- dist(t(counts_vst))
sampleDistMatrix <- as.matrix(sampleDists)

library(RColorBrewer)
colors <- colorRampPalette( rev(brewer.pal(9, "Oranges")) )(255)
mydata_col = data.frame(condition=meta_data$condition, row.names = meta_data$sample)

library(pheatmap)
p_3 <- pheatmap(
  sampleDistMatrix,
  clustering_distance_rows=sampleDists,
  clustering_distance_cols=sampleDists,
  col=colors,
  annotation_col = mydata_col,
```

```
  main="Euclidean distance between samples"
)

### make my customized function to save heatmap
save_pheatmap_pdf <- function(x, filename, width, height) {
  stopifnot(!missing(x))
  stopifnot(!missing(filename))
  pdf(filename, width=width, height=height)
  grid::grid.newpage()
  grid::grid.draw(x$gtable)
  dev.off()
}
save_pheatmap_pdf(p_3, "QCplots/sample_heatmap.pdf", 7, 6.5)

### 3. gene expression heatmap of top 1000 variable genes
counts_vst_var <- as.data.frame(counts_vst)
counts_vst_var$var <- apply(counts_vst,1,var)
counts_vst_var <- counts_vst_var %>%
saveRDS(counts_vst_var, "counts_vst_var.rds", compress=T)
counts_vst_var <- counts_vst_var %>%
  slice(1:1000) %>%
  as.matrix()

p_4 <- pheatmap(counts_vst_var[,1:24],
              scale="row",
              color = colorRampPalette(c("darkblue", "white", "darkred"))(1000),
              annotation_col = mydata_col,
              main="Gene Expression (VST) of Top 1000 Variable Genes",
              show_rownames = F)
save_pheatmap_pdf(p_4, "QCplots/gene_heatmap.pdf", 7, 6.5)

### 4. PCA plots
# make my customized function to plot PCA
plotPCA_vst <- function (object, assay, ntop = 1000) {
  rv <- rowVars(assay(object, assay))
  select <- order(rv, decreasing = TRUE)[seq_len(min(ntop, length(rv)))]
  pca <- prcomp(t(assay(object, assay)[select, ]), center=TRUE, scale=FALSE)
  percentVar <- pca$sdev^2/sum(pca$sdev^2)
  df <- cbind( as.data.frame(colData(object)), pca$x)
  # order points so extreme samples are more likely to get label
        ord <- order(abs(rank(df$PC1)-median(df$PC1)), abs(rank(df$PC2)-median(df$PC2)))
        df <- df[ord,]
```

```
      attr(df, "percentVar") <- data.frame(PC=seq(along=percentVar), percentVar=100*percentVar)
      return(df)
    }

    pca.data <- plotPCA_vst(dds_new, assay="vst", ntop=1000)
    saveRDS(pca.data, "pca.data.rds", compress = T)

    for (i in c(1:4)) {
      for (j in c((i+1):5)) {
        percentVar <- round(attr(pca.data, "percentVar")$percentVar)
        p <- ggplot(pca.data, aes_string(x=paste0("PC",i), y=paste0("PC",j), color="condition")) +
          geom_point(size=3) +
          xlab(paste0("PC",i,": ",percentVar[i],"% variance")) +
          ylab(paste0("PC",j,": ",percentVar[j],"% variance")) +
          theme_bw()
        ggsave(filename = paste0("QCplots/PCAplot.","PC",i,"vs","PC",j,".pdf"), plot = p, width=6, height=5)
      }
    }

    ### 5. t-SNE plots
    library(Rtsne)
    # make my customized function to plot t-SNE
    plottsne_vst <- function (object, assay, perpl, ntop = 1000) {
      rv <- rowVars(assay(object, assay))
      select <- order(rv, decreasing = TRUE)[seq_len(min(ntop, length(rv)))]
      set.seed(123)
      tsne <- Rtsne(t(assay(object, assay)[select, ]), perplexity=perpl)
      df <- as.data.frame(tsne$Y)
      names(df) <- c("Dimention1","Dimention2")
      df <- cbind(df, meta_data)
      return(df)
    }

    for (i in 1:6) {
      tsne.data <- plottsne_vst(dds_new, assay="vst", perpl=i, ntop=1000)
      saveRDS(tsne.data, paste0("tsne.data-perplexity_",i,".rds"), compress = T)

```

```
        p <- ggplot(tsne.data, aes_string(x="Dimention1", y="Dimention2", color="condition")) +
          geom_point(size=3) +
          xlab("Dimention 1") +
          ylab("Dimention 2") +
          theme_bw()
        ggsave(filename = paste0("QCplots/tSNEplot-perplexity_",i,".pdf"), plot = p, width=6, height=5)
      }
```

13. Visualization results of read counts.

The raw data in the count matrix were normalized via two methods, rlog and vst, and were displayed in two box plots (Figure 3 and 4).

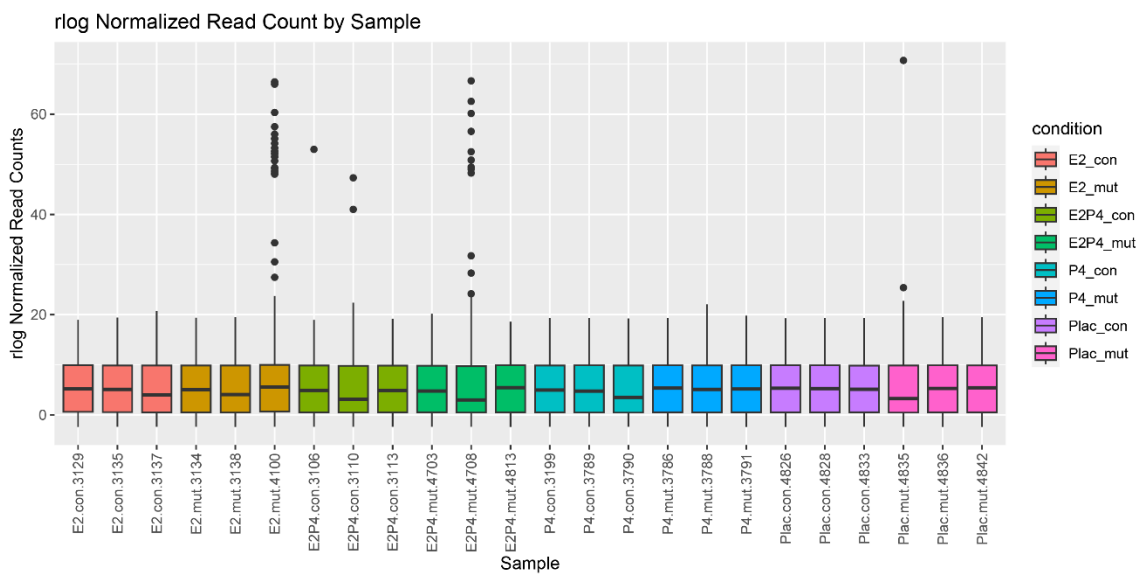


Figure 3. Box plot of read counts normalized by the rlog method. Samples in different groups are in different colors. The X-axis represents each sample. The Y-axis represents read counts normalized by the rlog method.

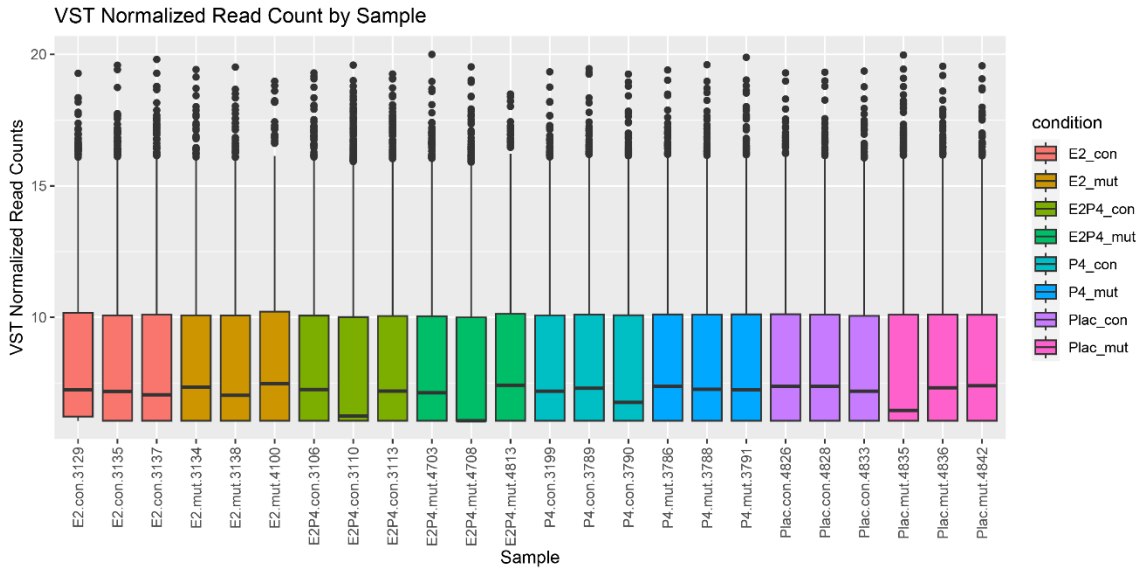


Figure 4. Box plot of read counts normalized by the vst method. Samples in different groups are in different colors. The X-axis represents each sample. The Y-axis represents read counts normalized by the vst method.

To observe the correlation among samples, I first calculated the distance between each pair of samples and then visualized their Euclidean distance in the heatmap (Figure 5).

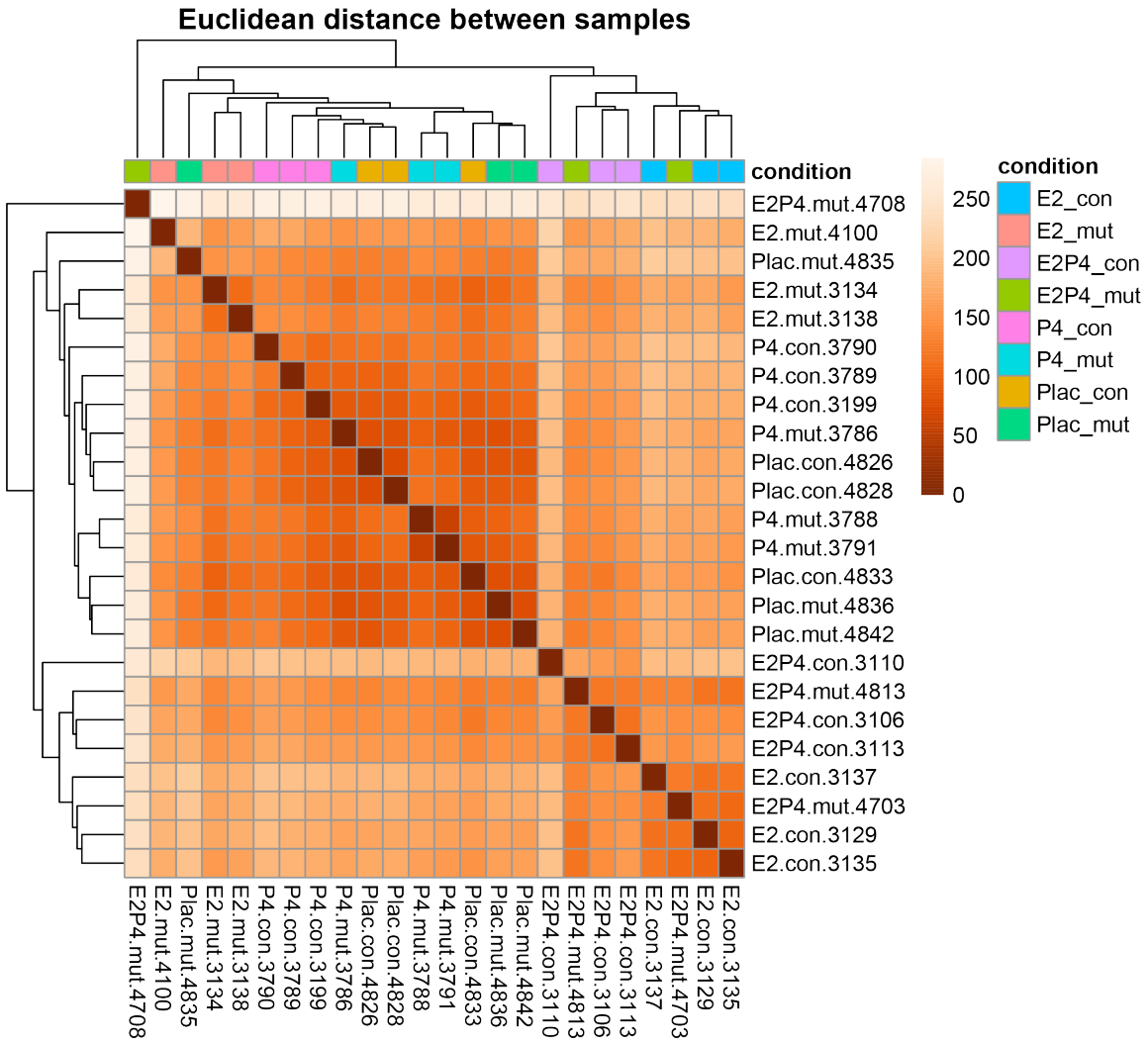


Figure 5. Heatmap of Euclidean distance between samples. Samples in different groups are in different colors. The color of the crossing square between two samples indicates the distance between those two samples. The darker the crossing square is, the smaller distance between those two samples is.

In addition to the correlation at the sample level, I also calculated and visualized the correlation across all samples at the gene expression level (Figure 6).

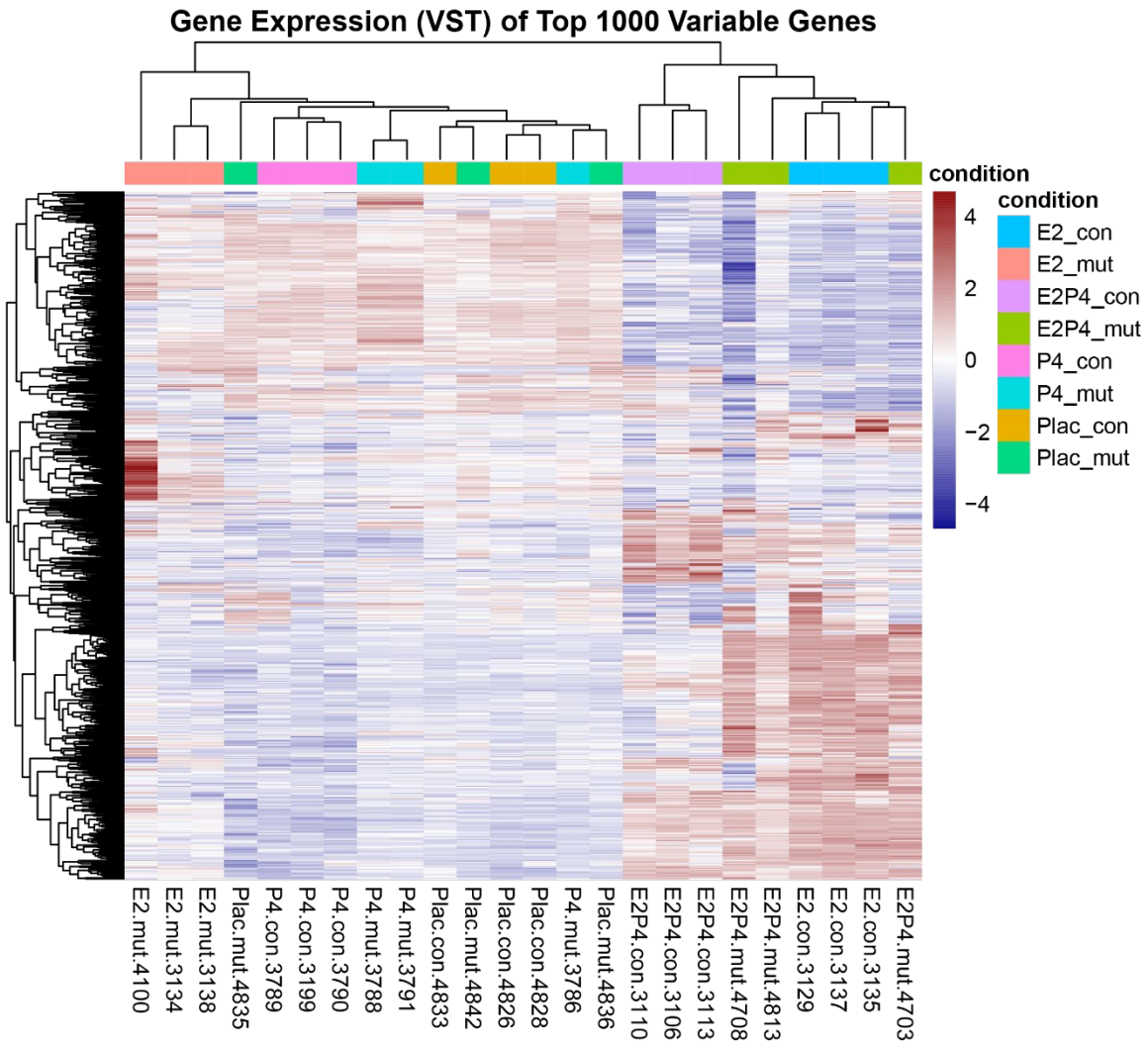


Figure 6. Heatmap of gene expression across samples. Samples in different groups are in different colors. The color of each gene indicates the scaled level of gene expression across samples. The redder indicates the higher level while the bluer indicates the lower level.

To further observe the distribution of samples, I reduced the dimension of read counts via principal component analysis (PCA) and visualized the results within combinations of each two of the first five principal components, i.e., PC1 and PC2, PC1 and PC3, PC1 and PC4, PC1 and PC5, PC2 and PC3, PC2 and PC4, PC2 and PC5, PC3 and PC4, PC3 and PC5, PC4 and PC5. The first two PCA plots were displayed here as examples (Figure 7A and 7B). In addition, we projected the read counts into the t-distributed Stochastic Neighbor Embedding (t-SNE) projection with a perplexity value from 1 to 6. The first and the last t-SNE plots were displayed here as examples (Figure 7C and 7C). Both PCA and t-SNE plots demonstrated the separation of samples by genotype and hormone treatment. Particularly, the E2 treatment affected gene expression the most significantly, and the P4 treatment regulated gene expression in a strikingly different way from E2 treatment, while the inner group heterogeneity could not be ignored. The correlation

observed in PCA and t-SNE plots was consistent with that in heatmaps at the sample distance level (Figure 5) and gene expression level (Figure 6).

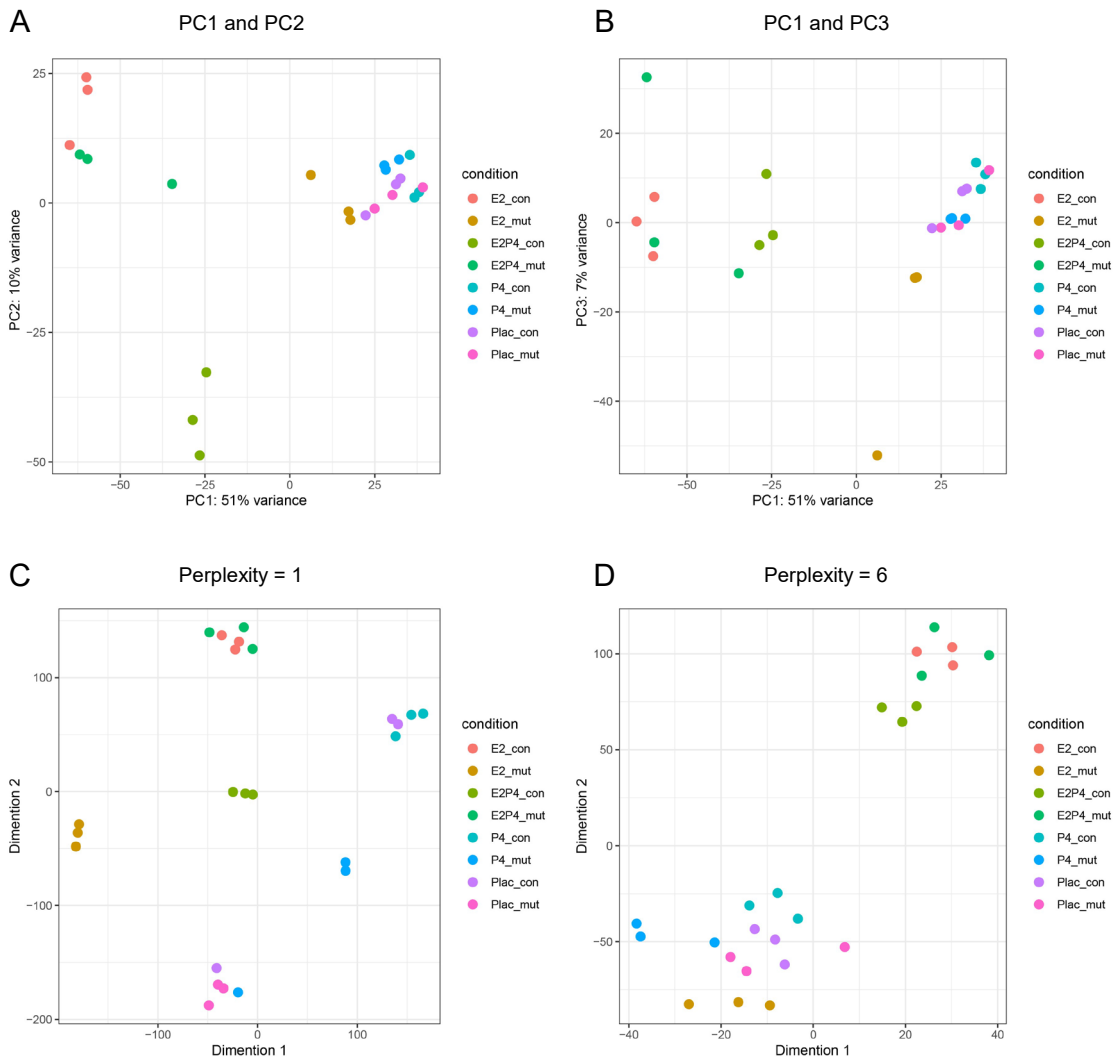


Figure 7. Sample distribution in projections of reduced dimension. A-B. PCA plots with PC1 and PC2 (A), and PC1 and PC3 (B). C-D. t-SNE plots with perplexity = 1 (C) and perplexity = 6 (D).

14. Convert gene ID from gene symbol to ENTREZID.

The default gene IDs in the STAR output are gene symbols that are intuitive and convenient to read and understand. However, ENTREZID is preferred by many advanced bioinformatic software[14] because it is unique and more accurate than gene symbol. Therefore, I converted gene IDs from gene symbols to ENTREZIDs, which were required in subsequent analyses.

```
### create folders
setwd("~/")
if (!file.exists("03-DE")){
```

```
  dir.create("03-DE")
}
setwd("~/03-DE")

### read in gene information obtained from Nextflow
library(readr)
bg_ID <- read_delim("~/01-Nextflow/output/star_salmon/salmon_tx2gene.tsv",
                    delim = "\t", escape_double = FALSE,
                    col_names = FALSE, trim_ws = TRUE)
names(bg_ID) <- c("transcript", "gene", "gene_name")
bg_ID <- bg_ID[,-1] %>% unique()

library(clusterProfiler)
library(org.Mm.eg.db)
#keytypes(org.Mm.eg.db)
bg_ENTREZID <- bitr(bg_ID$gene_name,
                    fromType = "SYMBOL",
                    toType = "ENTREZID",
                    OrgDb = "org.Mm.eg.db")
bg_ID <- left_join(bg_ID, bg_ENTREZID, by=c("gene_name"="SYMBOL"))
saveRDS(bg_ID, "bg_ID.rds", compress = T)
```

15. Differentially expressed gene (DEG) analysis.

In this project, we aimed to compare the difference between mutant mice (mut) and their wild-type littermates (con) in each condition (hormone treatment or placebo) separately, but the same analysis codes can be used to compare every two groups of samples with minor adjustments.

```
### create folders
setwd("~/03-DE")

if (!file.exists("DEfiles")){
  dir.create("DEfiles")
}

if (!file.exists("MAplots")){
  dir.create("MAplots")
}

if (!file.exists("Volcanoplots")){
  dir.create("Volcanoplots")
}

if (!file.exists("Heatmaps")){
  dir.create("Heatmaps")
}
```

```r

### generate FC and FDR via DESeq2
library(DESeq2)
library(EnhancedVolcano)
# read in dds object and metadata
dds_new <- readRDS("~/02-QC/dds_new.rds")
meta_data <- readRDS("~/02-QC/meta_data.rds")

### run DESeq to get result objects in loop
library(ggplot2)
library(readr)
library(tibble)
library(pheatmap)
counts_vst <- dds_new@assays@data$vst
dds_new_res <- DESeq(dds_new)
saveRDS(dds_new_res, "dds_new_res.rds", compress = T)

for (i in levels(meta_data$treatment)) {

  ### 1. DE results
  sampleA <- paste(i, "con", sep = "_")
  sampleB <- paste(i, "mut", sep = "_")
  condition <- "condition"
  contrastV <- c(condition, sampleA, sampleB)
  res <- results(dds_new_res,  contrast=contrastV)
  saveRDS(res, paste0("DEfiles/DE-",sampleB,"vs",sampleA,".rds"), compress = T)

  # join gene_name
  res_df <- as.data.frame(res)
  res_symbol <- res_df %>%
    as.data.frame() %>%
    rownames_to_column(var="gene") %>%
    left_join(bg_ID, by="gene")

  ### 2. MA plots
  pdf(file = paste0("MAplots/MAplot-",sampleB,"vs",sampleA,".pdf"),
      width=7, height=5)
  MAplot <- plotMA(object=res, alpha=0.05,
                   main=paste0(sampleB,"vs",sampleA),
                   colNonSig="black",
                   colSig="red",
                   colLine="blue")
  dev.off()

  ### 3. volcano plots
  vol <- EnhancedVolcano(res_symbol,
```

```
                  lab = NA,
                  x = 'log2FoldChange',
                  y = 'padj',
                  ylab = bquote(~-Log[10] ~ italic(FDR)),
                  pCutoff = 0.05,
                  FCcutoff = 2,
                  title = paste0(sampleB,"vs",sampleA),
                  legendLabels=c('Not DE &\nabsolute FC < 2',
                                 'Not DE &\nabsolute FC > 2',
                                 'FDR < 0.05 &\nabsolute FC < 2',
                                 'FDR < 0.05 &\nabsolute FC > 2'),
                  legendPosition = 'right')
  ggsave(filename = paste0("Volcanoplots/Volcanoplot-",sampleB,"vs",sampleA,".pdf"), plot=vol, width = 7, height = 7)

  vol_lab <- EnhancedVolcano(res_symbol,
                             lab = res_symbol$gene_name,
                             x = 'log2FoldChange',
                             y = 'padj',
                             ylab = bquote(~-Log[10] ~ italic(FDR)),
                             pCutoff = 0.05,
                             FCcutoff = 2,
                             title = paste0(sampleB,"vs",sampleA),
                             legendLabels=c('Not DE &\nabsolute FC < 2',
                                            'Not DE &\nabsolute FC > 2',
                                            'FDR < 0.05 &\nabsolute FC < 2',
                                            'FDR < 0.05 &\nabsolute FC > 2'),
                             legendPosition = 'right')
  ggsave(filename = paste0("Volcanoplots/Volcanoplot_gene_name-",sampleB,"vs",sampleA,".pdf"), plot=vol_lab, width = 7, height = 7)

  ### 4. heatmaps
  res_symbol <- res_symbol %>%
    mutate(change=case_when(padj < 0.05 ~ "DE",
                            TRUE ~ "Not")) %>%
    mutate(direction=case_when(change == "DE" & log2FoldChange > 0 ~ "Up",
```

```
                                change == "DE" & log2FoldChange < 0 ~ "Down",
                                      TRUE ~ "Not"))
        write.csv(res_symbol, paste0("DEfiles/DE-",sampleB,"vs",sampleA,".csv"))

        counts_vst_DE <- counts_vst[res_symbol$change=="DE",]
        mydata_col <- data.frame(condition=meta_data$condition, row.names = meta_data$sample)
        heat_DE_1 <- pheatmap(counts_vst_DE,
                              scale="row",
                              color = colorRampPalette(c("darkblue", "white", "darkred"))(1000),
                              annotation_col = mydata_col,
                              main=paste0(sampleB,"vs",sampleA," FDR < 0.05"),
                              show_rownames = F)
        save_pheatmap_pdf(heat_DE_1, paste0("Heatmaps/Heatmap-",sampleB,"vs",sampleA,".pdf"), 7, 6.5)

        # row names
        tree_row <- data.frame(gene = rownames(counts_vst_DE[heat_DE_1$tree_row[["order"]],])) %>%
          left_join(bg_ID, by="gene")
        saveRDS(tree_row, paste0("Heatmaps/Heatmap-",sampleB,"vs",sampleA,".rds"), compress = T)
        write.csv(tree_row, paste0("Heatmaps/Heatmap-",sampleB,"vs",sampleA,".csv"))

        meta_data_2 <- meta_data[meta_data$treatment==i,]
        mydata_col_2 = data.frame(condition=meta_data_2$condition, row.names = meta_data_2$sample)
        counts_vst_DE_2 <- counts_vst_DE[,meta_data_2[,1]]
        heat_DE_2 <- pheatmap(counts_vst_DE_2,
                              scale="row",
                              color = colorRampPalette(c("darkblue", "white", "darkred"))(1000),
                              annotation_col = mydata_col_2,
                              main=paste0(sampleB,"vs",sampleA," FDR < 0.05"),
                              show_rownames = F)
        save_pheatmap_pdf(heat_DE_2, paste0("Heatmaps/Heatmap2-",sampleB,"vs",sampleA,".pdf"), 7, 6.5)

        # row names
        tree_row_2 <- data.frame(gene = rownames(counts_vst_DE[heat_DE_2$tree_row[["order"]],])) %>%
          left_join(bg_ID, by="gene")
```

```
        saveRDS(tree_row_2, paste0("Heatmaps/Heatmap2-",sampleB,"vs",sampleA,".rds"), compress = T)
        write.csv(tree_row_2, paste0("Heatmaps/Heatmap2-",sampleB,"vs",sampleA,".csv"))
      }
```

16. Visualization results of DEG analysis.

Four sub-folders, i.e., Plac_mutvPlac_con, E2_mutvsE2_con, P4_mutvsP4_con, and E2P4_mutvsE2P4_con, were created inside the folder of ~/03-DE/DEfiles containing results of DEG analysis. In addition, the results were visualized in various plots that were saved into sub-folders.

MA plots suggested that the fold changes of genes between mut and con were distributed evenly and independent of mean values of normalized counts (Figure 8).

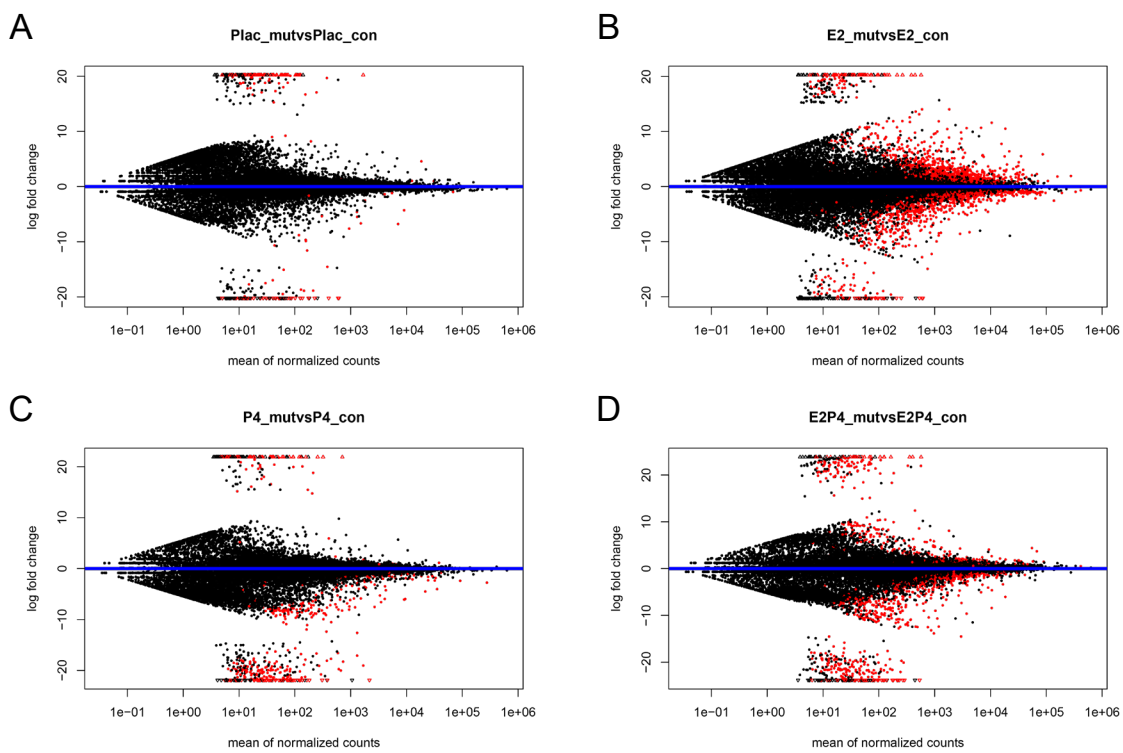


Figure 8. MA plots of DEG analysis results. X-axis represents the mean values of vst normalized counts. The Y-axis represents the log-transformed fold change in mut vs. con in each condition of hormone treatment.

In addition to MA plots, volcano plots were also generated to observe the significant changes between mut and con. Because the above analyses suggested that E2 treatment had the most obvious effect on mut vs. con, here I displayed the volcano plots of DEGs in mut vs. con under the E2 treatment as an example (Figure 9). The panel on the right also labeled symbols of the most significantly changed genes (Figure 9B).

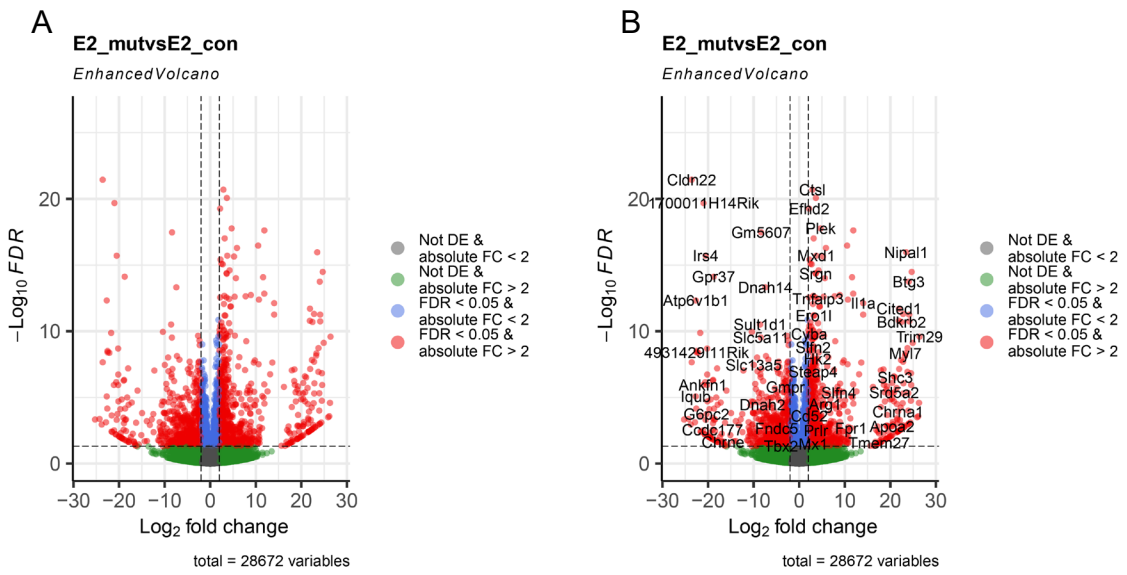


Figure 9. Volcano plots of DEGs in mut vs. con under the E2 treatment. The X-axis represents the log2 transformed fold change in mut vs. con. The Y-axis represents the minus log10 transformed FDR. Each dot represents one gene and is colored corresponding to the cutoffs shown in the legend.

To observe the gene expression across samples, heatmaps were also generated for two groups of samples (Figure 10A) and all groups of samples (Figure 10B). The heatmap showed the two distinct clusters of genes that were upregulated or downregulated in E2_mut vs. E2_con (Figure 10A).

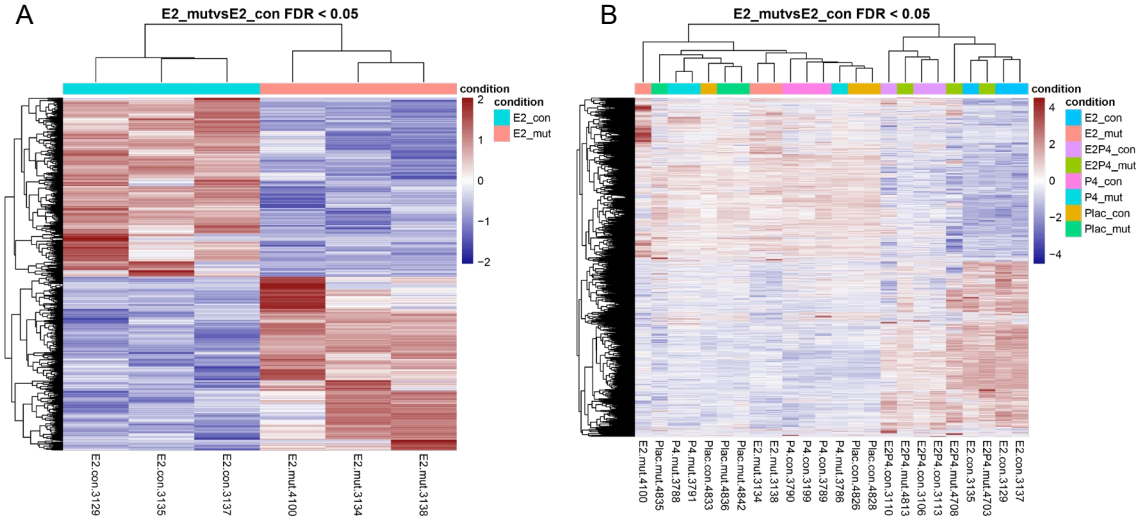


Figure 10. Heatmap of expression of significantly changed genes in mut vs. con under the E2 treatment. A. Heatmaps of comparison between two groups of samples. B. Heatmap across all samples. Samples in different groups are in different colors. The color of each gene indicates the scaled level of gene expression across samples. The redder indicates the higher level while the bluer indicates the lower level.

Downstream analysis with R

17. Functional analyses including Gene Ontology (GO) – Biological Process (BP), Cellular Component (CC), and Molecular Function (MF), and KEGG analyses.

```
### Create folders
setwd("~")
if (!file.exists("04-Pathway")){
  dir.create("04-Pathway")
}
setwd("~/04-Pathway")

if (!file.exists("Pathways_tables")){
  dir.create("Pathways_tables")
}

if (!file.exists("Pathways_figures")){
  dir.create("Pathways_figures")
}

library(clusterProfiler)
library(org.Mm.eg.db)
library(ggplot2)
library(readr)

bg_ID <- readRDS("../03-DE/bg_ID.rds")
bg <- bg_ID$ENTREZID
meta_data <- readRDS("../02-QC/meta_data.rds")
treatment <- levels(meta_data$treatment)

### Loops
for (i in treatment) {
  sampleA <- paste(i, "con", sep = "_")
  sampleB <- paste(i, "mut", sep = "_")
  # Read in DE table
  res_symbol <- read.csv(paste0("../03-DE/DEfiles/DE-",sampleB,"vs",sampleA,".csv"))

  ## Inner loop for Up or Down
  for (j in c("Up", "Down")){
    DEG <- res_symbol[res_symbol$direction == j,]$ENTREZID %>%
      as.character()

    # KEGG
    KEGG <- enrichKEGG(gene = DEG,
                       universe = bg,
```

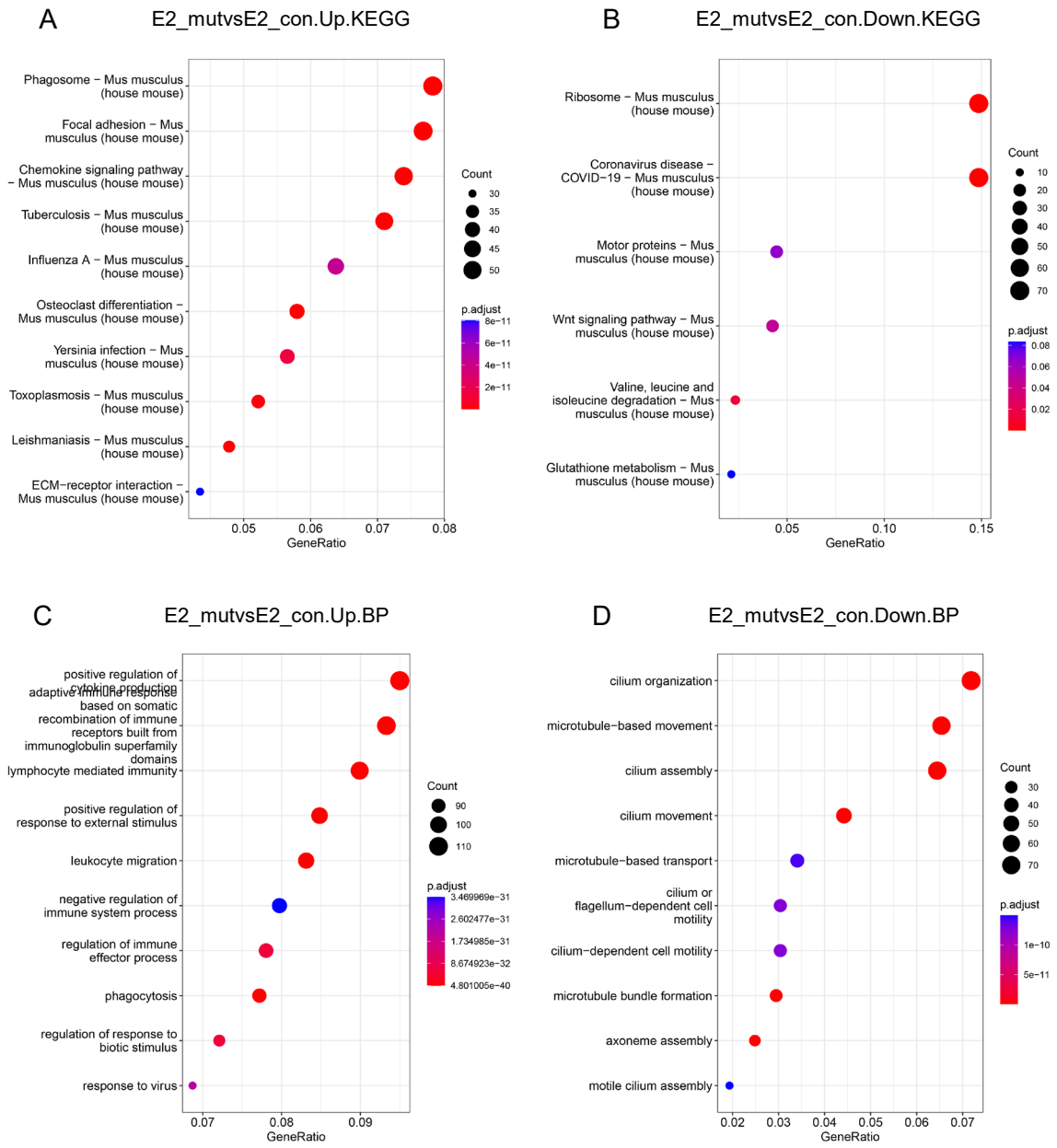
```
                        organism ='mmu',
                        pvalueCutoff = 0.1,
                        qvalueCutoff = 0.1,
                        use_internal_data =FALSE)
    KEGG <- setReadable(KEGG, OrgDb = org.Mm.eg.db, keyType="ENTREZID")
    write_csv(as.data.frame(KEGG@result),paste0("Pathways_tables/",sampleB,"vs",sampleA,".",j,".KEGG.csv"))
    saveRDS(KEGG,paste0("Pathways_tables/",sampleB,"vs",sampleA,".",j,".KEGG.rds"),compress=T)
    dot_KEGG <- dotplot(KEGG)
    ggsave(filename = paste0("Pathways_figures/",sampleB,"vs",sampleA,".",j,".KEGG.pdf"), plot = dot_KEGG, width = 7, height = 7)

    # GO sub-categories
    for (GO in c("CC", "MF", "BP")) {
      GO_enrich <- enrichGO(gene = DEG,
                            universe = bg,
                            OrgDb = "org.Mm.eg.db",
                            keyType = "ENTREZID",
                            ont = GO,
                            pvalueCutoff  = 0.1,
                            pAdjustMethod = "BH",
                            qvalueCutoff  = 0.1,
                            readable=T)
      GO_enrich <- simplify(GO_enrich,cutoff=0.7,
                            by="p.adjust",
                            select_fun=min,
                            measure = "Wang")
      write_csv(as.data.frame(GO_enrich@result),paste0("Pathways_tables/",sampleB,"vs",sampleA,".",j,".",GO,".csv"))
      saveRDS(GO_enrich,paste0("Pathways_tables/",sampleB,"vs",sampleA,".",j,".",GO,".rds"),compress=T)
      dot_GO <- dotplot(GO_enrich)
      ggsave(filename = paste0("Pathways_figures/",sampleB,"vs",sampleA,".",j,".",GO,".pdf"), plot = dot_GO, width = 7, height = 7)
      }
  }
}
```

18. Visualization of results of functional enrichment analysis.

DEGs were divided into two parts, i.e., upregulated (Up) and downregulated (Down) genes in mut vs. con under individual hormone treatment. The two parts were separately loaded into the clusterProfiler R package for functional enrichment of KEGG pathways and GO terms including BP, MF and CC sub-categories. Here, I displayed the bubble

charts of KEEG pathways and GO terms enriched from DEGs in mut vs. con under the E2 treatment as examples (Figure 11).



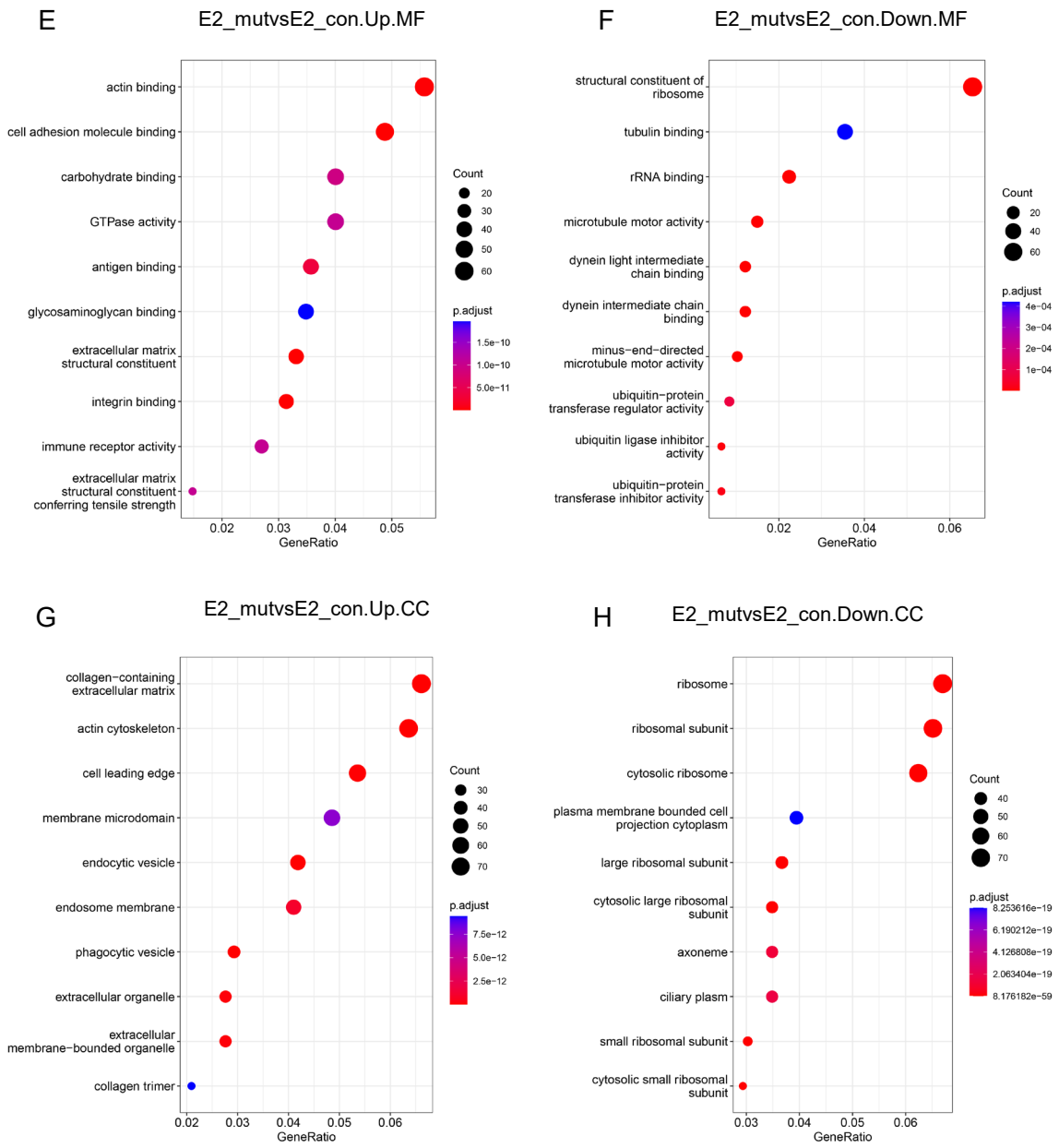


Figure 11. Functional enrichments. IDs of significantly upregulated and downregulated genes in E2_mut vs. E2_con were converted into the ENTREZ IDs and imported into the clusterProfiler R package to conduct functional enrichment. The top ten terms with adjusted p-values < 0.05 were displayed in bubble charts. Four categories were KEGG pathways from KEGG pathway database (A and B), GO_BP terms from Biological Process (Gene Ontology) database (C and D), GO_MF terms from Molecular Function (Gene Ontology) database (E and F), and GO_CC from Cellular Component (Gene Ontology) database (G and H). Color represents significance: the redder, the more significant.

Discussion

This pipeline started from processing raw fastq files obtained from the Illumina HiSeq4000 sequencing machine, aligned reads to the Ensembl/GRCm38 reference genome via STAR, quantified to count matrix via Salmon, visualized normalized counts, analyzed and visualized DEGs, enriched and visualized KEGG pathways and GO terms to deduce changed functions in R.

This pipeline combined the advantages of Nextflow and R from upstream to midstream, and to downstream analyses. In the upstream analysis, different reference genomes can be downloaded from the igenomes repository (https://support.illumina.com/sequencing/sequencing_software/igenome.html) provided by Illumina. In addition to STAR and Salmon, other alignment and quantification software can be selected as a substitution. Addition parameters can be adjusted in the Nextflow config file. Therefore, this pipeline that was built based on Nextflow can be extended to analyze RNA-seq data generated from other species and with different qualities.

In the midstream and downstream analyses, I took advantage of R in statistics and visualization. In addition to the pairs of comparisons that were demonstrated here as examples, samples from any two groups can be compared via DESeq2 by this pipeline. I am also considering expanding this pipeline to provide other options such as EdgeR and Limma which are also widely used. Of note, this pipeline inherited one of the best advantages of R, i.e., the customized and fast visualization. I provided codes to observe the data and results in many most popular plots. The generated figures can be combined and trimmed flexibly. All figures can be exported to PDF files which are editable and can be transformed into any other formats in graph edition software like Illustrator.

In summary, this pipeline provides an easy and convenient way to analyze RNA-seq data and reproduce bioinformatics analysis. Particularly, the enriched KEGG pathways and GO terms are very useful in the interpretation of biological functions and the discovery of regulatory mechanisms.

References

1. Qu, J.H., et al., *Characterization of diverse populations of sinoatrial node cells and their proliferation potential at single nucleus resolution.* Heliyon, 2023. **9**(1): p. e12708.
2. Qu, J.H., et al., *Transcriptome of left ventricle and sinoatrial node in young and old C57 mice.* Fortune Journal of Health Sciences, 2023. **6**: p. 332 - 356.
3. Qu, J.H., et al., *Proteomic Landscape and Deduced Functions of the Cardiac 14-3-3 Protein Interactome.* Cells, 2022. **11**(21).
4. Agrimi, J., et al., *Cardiac AC8 Over-Expression Increases Locomotion by Altering Heart-Brain Communication.* JACC Clin Electrophysiol, 2023.

5. Di Tommaso, P., et al., *Nextflow enables reproducible computational workflows.* Nat Biotechnol, 2017. **35**(4): p. 316-319.
6. Köster, J. and S. Rahmann, *Snakemake-a scalable bioinformatics workflow engine.* Bioinformatics, 2018. **34**(20): p. 3600.
7. Voss, K., G. Van der Auwera, and J. Gentry, *Full-stack genomics pipelining with GATK4 + WDL + Cromwell* F1000Research, 2017. **6**: p. 1381.
8. Ewels, P.A., et al., *The nf-core framework for community-curated bioinformatics pipelines.* Nat Biotechnol, 2020. **38**(3): p. 276-278.
9. Tarasov, K.V., et al., *A remarkable adaptive paradigm of heart performance and protection emerges in response to marked cardiac-specific overexpression of ADCY8.* Elife, 2022. **11**.
10. Wickham, H., et al., *Welcome to the Tidyverse.* Journal of Open Source Software, 2019. **4**(43).
11. Love, M.I., W. Huber, and S. Anders, *Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2.* Genome Biol, 2014. **15**(12): p. 550.
12. Van der Maaten, L. and G. Hinton, *Visualizing data using t-SNE.* Journal of machine learning research, 2008. **9**(11).
13. Van Der Maaten, L., *Accelerating t-SNE using tree-based algorithms.* The journal of machine learning research, 2014. **15**(1): p. 3221-3245.
14. Wu, T., et al., *clusterProfiler 4.0: A universal enrichment tool for interpreting omics data.* Innovation (Camb), 2021. **2**(3): p. 100141.
15. Kurtzer, G.M., V. Sochat, and M.W. Bauer, *Singularity: Scientific containers for mobility of compute.* PLoS One, 2017. **12**(5): p. e0177459.