Next Generation Sequencing Project A

Liliana Pamasa

2024-04-30

Introduction

Differential gene expression, otherwise known as DGE, analysis is one of the most widely used applications of RNA-sequencing data (1). This procedure is frequently utilized in various RNA-seq data analysis applications because it enables the identification of genes that are differentially expressed across two or more conditions. In order to facilitate specific hypothesis-driven studies, DEGs are widely employed to identify genotypic differences between two or more conditions of cells (1). A commonly used data set for DGE analysis is cancer cell lines. It is still difficult to effectively translate cancer genetic data into comprehensive knowledge regarding tumor biology and treatment options (6). Human cancer cell lines are the main representative of tumor biology and drug discovery. By facilitating experimental manipulation, detained mechanical studies, and various high-throughput applications, the understanding of systematic cell lines is promising (6). In this project, a DGE analysis was conducted comparing gene expression in control breast cancer cell lines to treatment lines with a gene, NRDE2, that has been silenced. The overall goal is to characterize the differentially expressed genes that may be impacted by knocking down NRDE2 (7).

Methods

Download FastQs

To begin, the fastQ samples were first download by executing a slurm script using the `wget` command. The exit statuses were also checked during this to make sure they were properly downloaded. The code is as follows:

```
#!/bin/bash
#
#SBATCH --nodes=1
#SBATCH --tasks-per-node=1
#SBATCH --cpus-per-task=1
#SBATCH --time=24:00:00
#SBATCH --mem=8GB
#SBATCH --job-name=download_fastqs
#SBATCH --mail-type=FAIL,END
#SBATCH --mail-user=llp293@nyu.edu


module purge

wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR781/000/SRR7819990/SRR7819990.fastq.gz
echo _ESTATUS_ [ wget SRR7819990 ]: $?
wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR781/001/SRR7819991/SRR7819991.fastq.gz
echo _ESTATUS_ [ wget SRR7819991 ]: $?
wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR781/002/SRR7819992/SRR7819992.fastq.gz
```

```
echo _ESTATUS_ [ wget SRR7819992 ]: $?
wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR781/003/SRR7819993/SRR7819993.fastq.gz
echo _ESTATUS_ [ wget SRR7819993 ]: $?
wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR781/004/SRR7819994/SRR7819994.fastq.gz
echo _ESTATUS_ [ wget SRR7819994 ]: $?
wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR781/005/SRR7819995/SRR7819995.fastq.gz
echo _ESTATUS_ [ wget SRR7819995 ]: $?
echo _END_ [ download.slurm ]: $(date)
```

Following the FastQ downloads, the following was executed to load the latest reference files for human from Ensemb with the fasta and gtf parameters:

```
latest_release=$(curl -s 'http://rest.ensembl.org/info/software?content-type=application/json' | grep -
wget -L ftp://ftp.ensembl.org/pub/release-${latest_release}/fasta/homo_sapiens/dna/Homo_sapiens.GRCh38.
wget -L ftp://ftp.ensembl.org/pub/release-${latest_release}/gtf/homo_sapiens/Homo_sapiens.GRCh38.${late
```

nf-core/rnaseq

After the fastQ files were downloaded, the nextflow module (v.23.04.1) was downloaded so that a nf-core/rnaseq pipeline (v.3.14) could be executed to trim reads and Salmon (v.1.10.1) could create quant.sf files for each sample. Within this slurm script, the input was created to reflect the sample names, corresponding fastq files, and their strandness. This sample sheet is used by the pipeline to auto-detect the samples as single-end. However, the strand-specificity is unknown, so auto is used. This allows the pipeline to sub-sample the input FastQ files to 1 million reads, use Salmon Quant to infer the strandedness automatically and then propagate this information to the remainder of the pipeline. The output directory was set to 'res' so that all the outputs were in their own directory for easy navigation. the '–fasta' and '–gtf' parameters were the downloaded human reference files in their respective formats. The '–extra_salmon_quant_args' was passed the `--gcBias` option so that Salmon corrected for GC bias. Next, the '-profile' argument was set to configure to the NYU HPC profile. Lastly, since the documentation for nf-core/rnaseq says that under DSL2, "parameters" (variables defined in a nextflow "parameters" block) must be defined in a YAML or JSON file, the JSON file from Week 9 was used. Within this file, the 'skip_trimming' parameter was set to false so that the default Trim Galore! wrapper tool would be used to perform quality trimming on the FastQ files. The alignment of these fastQ files were skipped and Salmon was used as the pseudo alinger. The '-input' and '-params-file' parameter files can be viewed in the appendix. The following is the batch script that was executed with requests a 24 hour run time and 4 GB of RAM:

```
#!/bin/bash
#
#SBATCH --nodes=1
#SBATCH --tasks-per-node=1
#SBATCH --cpus-per-task=1
#SBATCH --time=24:00:00
#SBATCH --mem=4GB
#SBATCH --job-name=nf_core
#SBATCH --mail-type=FAIL,END
#SBATCH --mail-user=llp293@nyu.edu

module purge

module load nextflow/23.04.1

nextflow run nf-core/rnaseq -r 3.14.0 \
--input /scratch/llp293/projectA/samplesheet.csv \
--outdir res \
--fasta /scratch/llp293/projectA/Homo_sapiens.GRCh38.dna_sm.primary_assembly.fa.gz \
```

```
--gtf /scratch/llp293/projectA/Homo_sapiens.GRCh38.111.gtf.gz \
--extra_salmon_quant_args "--gcBias " \
-profile nyu_hpc \
-params-file /scratch/work/courses/BI7653/hw9.2024/rnaseq.json
```

Convert Salmon TPMs to gene-level counts with tximport

To convert the Salmon TPMs produced in the nf-core/rnaseq pipeline to gene-level counts with tximports we first needed to install the appropriate packages – `tximport` (v.1.30.0). Once downloaded, the library was loaded and the Salmon quantification files were assigned to the "files" variable with the appropriate path to my local directory. Once assigned the quant.sf files, the sample names were assigned the "sample_names" variable. Next, the mapping file that is required for tximport, "tx2gene", is read via the file path on the local directory as well. Finally, the TPM values from the Salmon quantification files are imported using the 'tximport' function. The "type" arguement set to salmon so that the abundances are generated and "tx2gene" links transcript IDs to gene IDs for summarization. Once the 'txi' variable is created, a DESeq2 object can be created using the gene-level counts.

```
library(tximport)
patient_ids <- c('control1','control2','control3','treated1', 'treated2', 'treated3')
sample_names <- c(paste(patient_ids,sep=''))
sample_condition <- c(rep('control',3),rep('RNAi',3))

files <- file.path("/Users/lilianapamasa/Downloads/salmon",sample_names,"quant.sf")

names(files) <- sample_names

tx2gene <- read.table(file.path("/Users/lilianapamasa/Downloads/salmon/tx2gene.tsv"),
                      header=F,sep="\t")

# tx2gene links transcript IDs to gene IDs for summarization
txi <- tximport(files, type="salmon", tx2gene=tx2gene)
```

Conduct DGE with DESeq2

To conduct DGE with DESeq2, a DESeq2 object was created from the gene-level counts using 'DESeqDataSet-FromTximport()'. First, the DESeq2 (v.1.42.1) was loaded via libary. A variable 'dds' was assigned to this object. After creating the object, the lowly expressed gene counts were removed if they were >= 10 for each ESEMBL row id. Next, the `DESeq()` wrapper was executed and the produced DESeqDataSet results. The 'metadata.df' that was used for colData and the design arguments can be found in the appendix.

```
library(DESeq2)
library(magrittr)

metadata.df <- data.frame(sample = factor(sample_names),
                          condition = factor(sample_condition,levels = c('control','RNAi')))

row.names(metadata.df) <- sample_names

dds <- DESeqDataSetFromTximport(txi,
                                colData = metadata.df,
                                design = ~ condition)
keep <- rowSums(counts(dds)) >= 10
dds <- dds[keep,]
counts(dds) %>%
  dim()
```

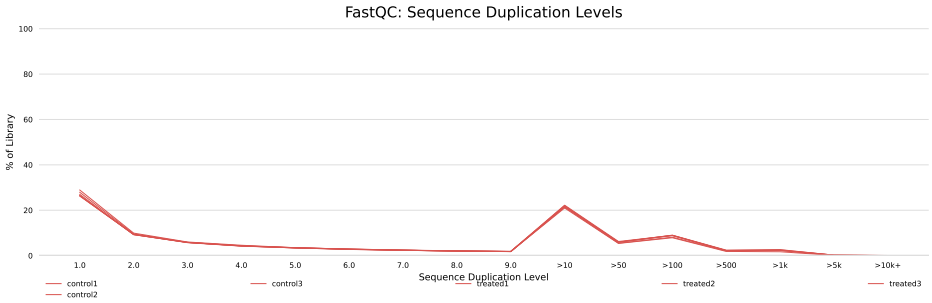
```
## [1] 22832     6
```

```
dds <- DESeq(dds) # execute DESeq wrapper

# Extract results table with unshrunken LFCs with alpha set to 0.05
res <- results(dds, alpha = 0.05)
```

MultiQC Report

Next, the generated MultiQC report from the nf-core/rnaseq pipeline was downloaded and examined. The MultiQC report indicated that for Sequence Duplication Levels all 6 samples failed as well as the Per Base Sequence Count. This means the proportion of each base position for which each of the four normal DNA bases could not be called and the relative level of duplication found for each sequence could not be found. The Sequence Duplication Levels plot is shown, however, the Per Base Sequence Count was unable to be downloaded from the MultiQC Report.



In addition, 4 of the 6 samples had warnings for containing over-represented sequences. Shown in the table below is the top 5 over-represented sequences based on the highest number of all reads in at least one sequence.

ID	Sequence
1	GATCGGAAGAGCACACGTCTGAACTCCAGTCACCCGTCCCGATCTCGTAT
2	GATCGGAAGAGCACACGTCTGAACTCCAGTCACATGTCAGAATCTCGTAT
3	GATCGGAAGAGCACACGTCTGAACTCCAGTCACAGTTCCGTATCTCGTAT
4	GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG
5	GATCGGAAGAGCACACGTCTGAACTCCAGTCACAGTCAACAATCTCGTAT

ID	Number_of_Occurances	Percentage_of_All_Reads
1	115467	0.0335%
2	107283	0.0311%
3	77394	0.0225%
4	76580	0.0222%
5	76303	0.0221%

Shurnken LFCs & Multiple Test Correction

The `lfcShrink` function adds shrunken log2 fold changes (LFC) and SE to the DESeq object. The "padj" column contains p-values that have had a multiple test correction applied. The approach applied by default is the false discovery rate (FDR) and is calculated using the Benjamin-Hochberg ("BH") method. Since the 'type' specifies apeglm, it passes along the DESeq2 MLE log2 fold changes and standard errors to the apeglm function in the apeglm package, and re-estimates posterior LFCs for the coefficient specified by coef. The final DESeq results object along with the shrunken LFCs were used in downstream analysis to generate statistically significant genes, a PCA Plot, a MA Plot, a P-value Histogram, and a Dispersion-By-Mean Plot. All of these results are found in the results section.

```r
library(tibble)
# Create a results object from res with updated shrunken LFC estimates.
res.lfcShrink <- lfcShrink(dds,
                           res = res, # here we provide the results object with alpha = 0.05
                           coef = 'condition_RNAi_vs_control',type = 'apeglm')

res.lfcShrink.df <- res.lfcShrink %>%
  as.data.frame() %>%
  rownames_to_column(var = "feature_id") %>%
  as_tibble()
```

Results

Table of Total Number of Reads and Mapping Rate for Each Sample

Sample	Number_of_Reads	Map_Rate
SRR7819990	61376100	90.90073440405693
SRR7819991	63865536	92.08544122651703
SRR7819992	56138272	93.05537822873117
SRR7819993	57824745	92.39376166783192
SRR7819994	58946312	92.68125391697545
SRR7819995	36412431	92.65945411973398

Table of Top 10 Highly Significant DGE Genes

```
## # A tibble: 10 x 6
##    feature_id      baseMean log2FoldChange  lfcSE    pvalue      padj
##    <chr>              <dbl>          <dbl>  <dbl>     <dbl>     <dbl>
##  1 ENSG00000175334    6423.          1.66  0.0655 2.70e-142 4.25e-138
##  2 ENSG00000163041    7972.          1.66  0.0717 1.46e-120 1.15e-116
##  3 ENSG00000196396    6618.          1.15  0.0526 4.36e-107 2.29e-103
##  4 ENSG00000105976    9566.          1.57  0.0758 2.48e- 96 9.75e- 93
##  5 ENSG00000128595   22967.          1.48  0.0719 1.49e- 95 4.69e- 92
##  6 ENSG00000101384   11784.          1.31  0.0660 5.01e- 89 1.31e- 85
##  7 ENSG00000124333    2742.          1.48  0.0747 7.62e- 89 1.71e- 85
##  8 ENSG00000117632   16768.          1.34  0.0703 3.53e- 82 6.94e- 79
##  9 ENSG00000180398   20588.          0.909 0.0514 5.99e- 71 1.05e- 67
## 10 ENSG00000213281    6962.          1.18  0.0675 1.12e- 69 1.76e- 66
```

Number of Statistically Significant Genes at FDR of 0.05

Using a 5% cutoff for determination if genes are statistically significant shows that 3,608 of the genes within this set are false rejections of the null hypothesis, but its impossible to tell exactly which ones are the false rejection.

FDR < 0.05
3608

Number of Significant Higher & Lower Expression in RNAi vs Control

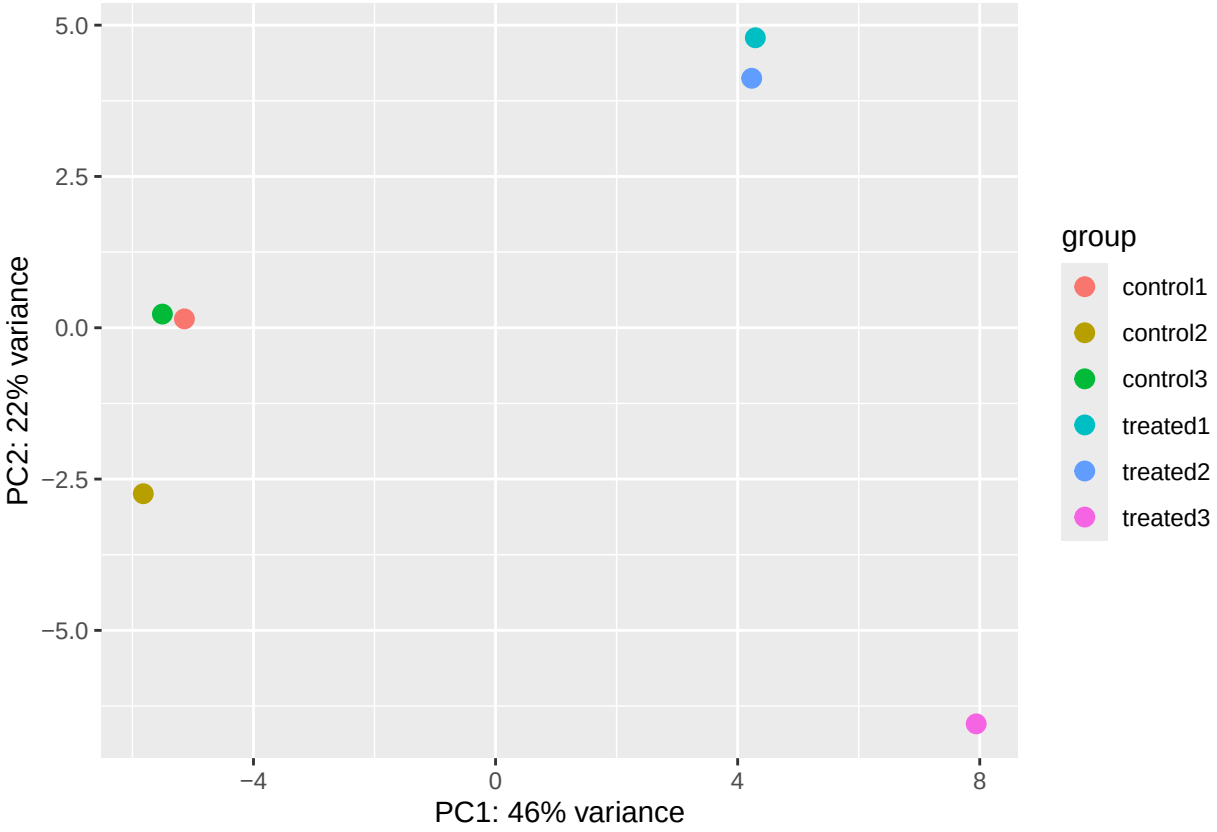
Now, by controlling the log2FoldChange, we can determine which of those significant genes are up-regulated and which are down-regulated. According to the chart, there are 1,930 up-regulated genes and 1,678

down-regulated genes.

Up-Regulated	Down-Regulated
1930	1678

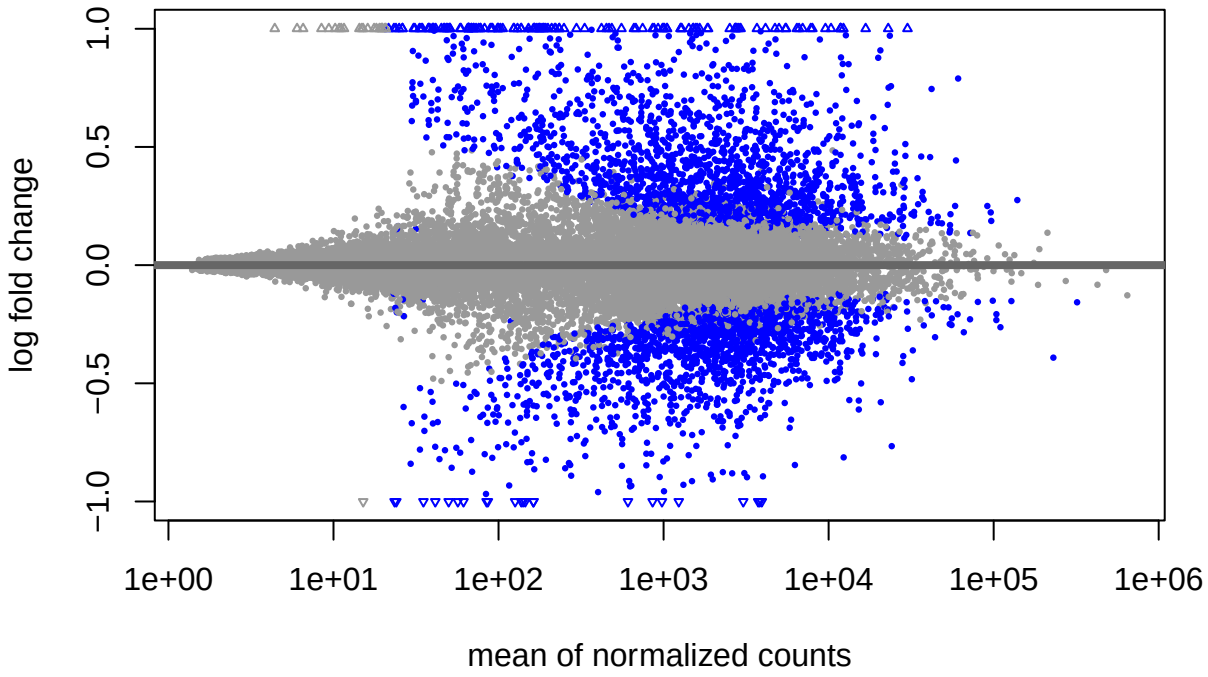
PCA Plot

A PCA plot shows sample clusters based on their similarity and reduces the overwhelming number of dimensions by constructing principal components (2). Based on this PCA plot, treated3 is an outlier and the treated group clusters high around the x-axis at x = 4 which is PC1: 46% variance. On the other hand, the control group clusters around the y-axis which is PC2: 22% variance. Before conducting PCA, the raw count matrix in DESeq2 was first transformed with "regularized log" or `rlog`.



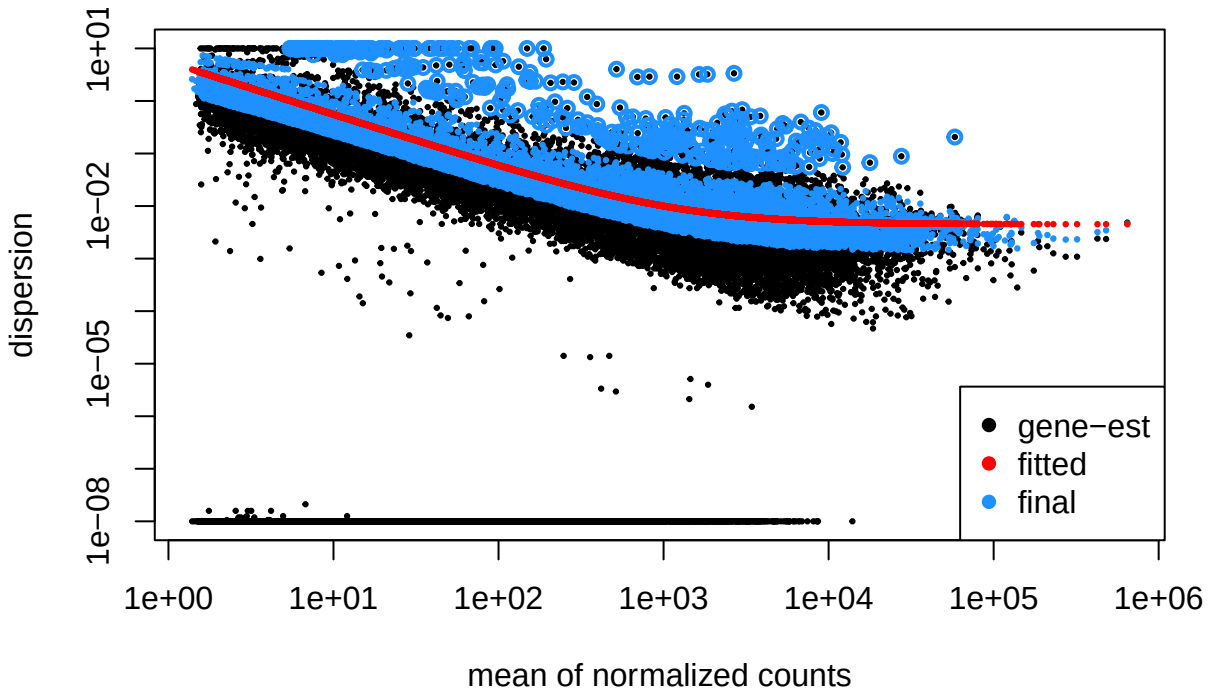
MA Plot after LFC Shrinkage

In DESeq2, the function plotMA shows the log2 fold changes attributable to a given variable over the mean of normalized counts for all the samples in the DESeqDataSet. Points will be colored blue if the adjusted p value is less than 0.1. Points which fall out of the window are plotted as open triangles pointing either up or down. The log fold-change threshold is set at -4 to 4. This MA plot shows a fairly even distribution relative to the y-axis, which tightens with an increase along the x-axis. Sometimes, biological significance may indicate an expected spread higher or lower on the y-axis (3). However, in this plot, all or most of the data points fall close to 0 along the y-axis, which indicates that the two groups would be highly similar in expression pattern. In this plot, blue points indicate statistically significant DEGs, gray points are not significantly different between control and RNAi tissues.



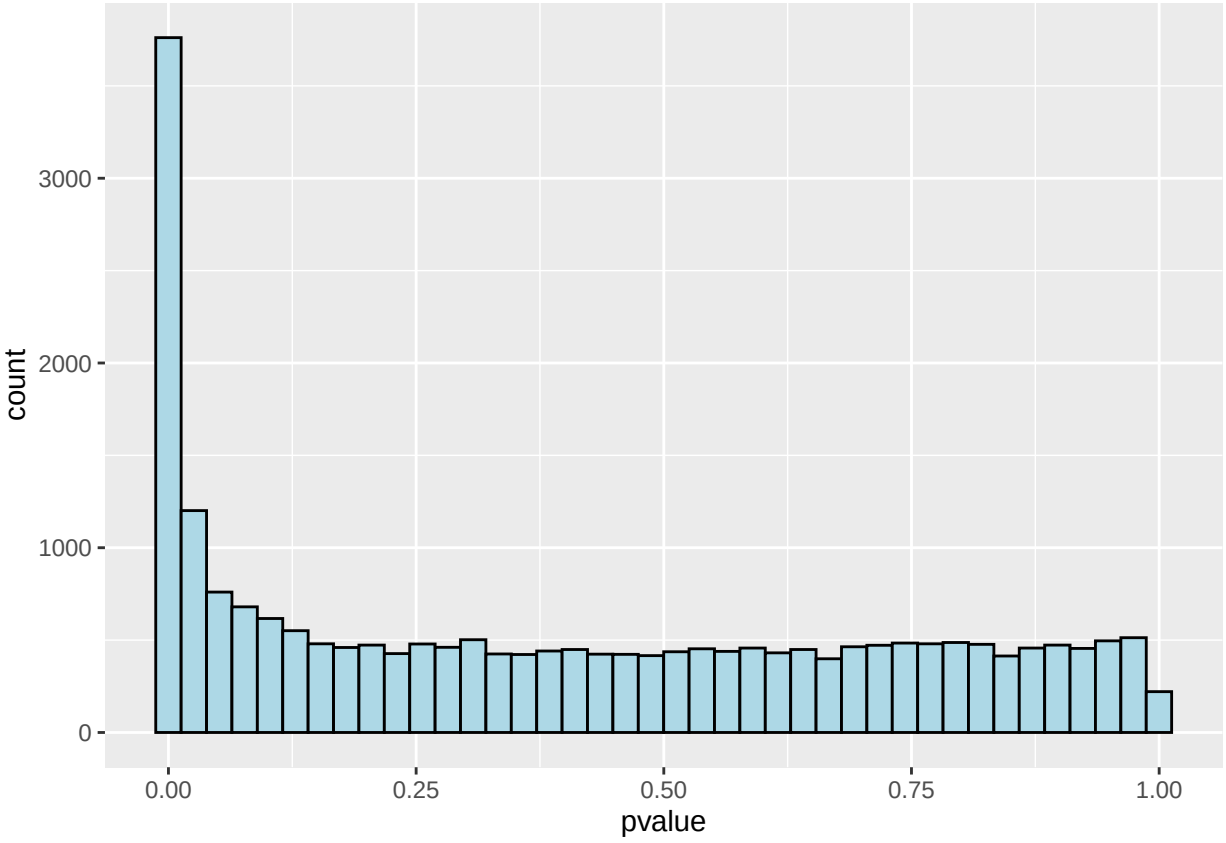
Dispersion By Mean Plot

Te plot below shows the dispersion-by-mean plot. A typical dispersion-by-mean plot has the final estimates shrunk from the gene-wise estimates towards the fitted estimates. From this generated plot, the parametric curve does seem to fit the observed dispersion-mean-relationship well, but there is over expression shown above the fitted line in the final dispersion estimates. A parametric fit, which is the default fitType, on a dispersion-mean relation of the form dispersion is equal to asymptDisp plus extraPois divided by the mean via a robust gamma-family GLM (4).



Raw P-value Histogram

This histogram shows the raw p-value counts. On the surface is a set of well-behaved p-values. The peak grows taller and closer to 0 and indicates that more p-values are close to 0 and therefore are significant. This is where the alternative hypothesis resides, along with some false positives (5). This also shows an enrichment of low p-values, which is the expected result if there is a large class of differentially expressed genes between treatment and control. Thus, the reason for removing lowly expressed genes before performing `DESeq()`.



Discussion

After analyzing the results, we are able to characterize differentially expressed genes that may be impacted by knocking down NRDE2. By identifying the significant DGEs and allowing for multiple test corrections, 3,608 genes passed the threshold and can be used in downstream analysis to try to understand the biological functions, regulatory networks, metabolic pathways, etc. After DEGs have been identified, the most common next step is to conduct a GO enrichment analysis on those genes. This comprehensive framework describes the function of genes with a set of terms that describe the function of genes with some terms nested within others in a hierarchical fashion.

Appendix

samplesheet.csv content

```
sample,fastq_1,strandedness
control1,SRR7819990.fastq.gz,auto
control2,SRR7819991.fastq.gz,auto
control3,SRR7819992.fastq.gz,auto
treated1,SRR7819993.fastq.gz,auto
```

```
treated2,SRR7819994.fastq.gz,auto
treated3,SRR7819995.fastq.gz,auto
```

rnaseq.json content

```
{
    "max_memory": "22.GB",
    "max_cpus": 4,
        "max_time": "4.h",
    "skip_trimming": false,
    "skip_alignment": true,
    "pseudo_aligner": "salmon",
    "save_reference": true
}
```

metadata.df content

	sample	condition
control1	control1	control
control2	control2	control
control3	control3	control
treated1	treated1	RNAi
treated2	treated2	RNAi
treated3	treated3	RNAi

versions

Process Name	Software	Version
CUSTOM_DUMPSOFTWAREVERSIONS	python	3.11.7
	yaml	5.4.1
CUSTOM_GETCHROMSIZES	getchromsizes	1.16.1
DESEQ2_QC_PSEUDO	bioconductor-deseq2	1.28.0
	r-base	4.0.3
FASTQC	fastqc	0.12.1
FQ_SUBSAMPLE	fq	0.9.1 (2022-02-22)
GTF2BED	perl	5.26.2
GTF_FILTER	python	3.9.5
GUNZIP_FASTA	gunzip	1.10
GUNZIP_GTF	gunzip	1.10
MAKE_TRANSCRIPTS_FASTA	rsem	1.3.1
	star	2.7.10a
SALMON_INDEX	salmon	1.10.1
SALMON_QUANT	salmon	1.10.1
SE_GENE	bioconductor-summarizedexperiment	1.24.0
	r-base	4.1.1
TRIMGALORE	cutadapt	3.4
	trimgalore	0.6.7
TX2GENE	python	3.9.5
TXIMPORT	bioconductor-tximeta	1.12.0
	r-base	4.1.1
Workflow	Nextflow	23.04.1
	nf-core/rnaseq	3.14.0

References:

1. Adam McDermaid, Brandon Monier, Jing Zhao, Bingqiang Liu, Qin Ma, Interpretation of differential gene

2. Team, BioTuring. "How to Read PCA Biplots and Scree Plots." Medium, 18 Sept. 2018, bioturing.medium.

3. McDermaid A, Monier B, Zhao J, Liu B, Ma Q. Interpretation of differential gene expression results o

4. Deshaies, Vivien, and Laurent Jourdren. Eoulsan - Differential Analysis Step, www.outils.genomique.bi

5. How to interpret a p-value histogram was published on December 15, 2014 and last modified on Decembe

6. Barretina J, Caponigro G, Stransky N, Venkatesan K, Margolin AA, Kim S, Wilson CJ, Lehár J, Kryukov

7. file:///Users/lilianapamasa/Downloads/project_optionA_2024%20(1).html


```
## Code Used
library(tximport)
patient_ids <- c('control1','control2','control3','treated1', 'treated2', 'treated3')
sample_names <- c(paste(patient_ids,sep=''))
sample_condition <- c(rep('control',3),rep('RNAi',3))

files <- file.path("/Users/lilianapamasa/Downloads/salmon",sample_names,"quant.sf")

names(files) <- sample_names

tx2gene <- read.table(file.path("/Users/lilianapamasa/Downloads/salmon/tx2gene.tsv"),
                      header=F,sep="\t")

# tx2gene links transcript IDs to gene IDs for summarization
txi <- tximport(files, type="salmon", tx2gene=tx2gene)

library(DESeq2)
library(magrittr)

metadata.df <- data.frame(sample = factor(sample_names),
                          condition = factor(sample_condition,levels = c('control','RNAi')))

row.names(metadata.df) <- sample_names

dds <- DESeqDataSetFromTximport(txi,
                                colData = metadata.df,
                                design = ~ condition)
keep <- rowSums(counts(dds)) >= 10
dds <- dds[keep,]
counts(dds) %>%
  dim()

dds <- DESeq(dds) # execute DESeq wrapper

# Extract results table with unshrunken LFCs with alpha set to 0.05
res <- results(dds, alpha = 0.05)

library(tibble)
```

```r
# Create a results object from res with updated shrunken LFC estimates.
res.lfcShrink <- lfcShrink(dds,
                           res = res, # here we provide the results object with alpha = 0.05
                           coef = 'condition_RNAi_vs_control',type = 'apeglm')

res.lfcShrink.df <- res.lfcShrink %>%
  as.data.frame() %>%
  rownames_to_column(var = "feature_id") %>%
  as_tibble()

library(dplyr)
res.lfcShrink.df %>%
  arrange(pvalue) %>%
  head(n=10)

res.lfcShrink.df %>%
  filter(padj < 0.05) %>%
  arrange(padj) %>%
  summarise(`FDR < 0.05` = sum(padj < 0.05,na.rm = T)) %>%
  kable(align = 'c')

res.lfcShrink.df %>%
   mutate(`LFC > 0` = case_when(log2FoldChange > 0 & padj < 0.05 ~1,
                                TRUE ~ 0)) %>%
  mutate(`LFC < 0` = case_when(log2FoldChange < 0 & padj < 0.05 ~ 1,
                                TRUE ~ 0)) %>%
  summarise(`Up-Regulated`= sum(`LFC > 0`),
            `Down-Regulated` = sum( `LFC < 0` )) %>%
  kable(align = "cc")

rld <- rlog(dds)

plotPCA(rld, intgroup = 'sample')

plotMA(res.lfcShrink) # LFC shrinkage was applied to this object

dds <- estimateSizeFactors(dds)
dds <- estimateDispersions(dds)
plotDispEsts(dds, fitType = "parametric")

library(ggplot2)
res.lfcShrink %>%
  as_tibble() %>% # coerce to tibble
  ggplot(aes(pvalue)) +
  geom_histogram(fill="light blue",color='black',bins = 40)
```