BIOINFORMATICS REPLICATION OF RNA-SEQ ANALYSIS

Project7BBG1002

Introduction

Reproducibility is a fundamental principle in scientific research, as it underpins the creation of reliable and trustworthy knowledge (Goodman et al., 2016). Over the past two decades, the issue of reproducible research has gained significant attention within the scientific community. A survey of 1,576 scientists revealed that over 80% believe there is a reproducibility crisis, which raises concerns about the credibility of scientific findings (Baker, 2016). The failure to replicate research outcomes is often attributed to how methods are disseminated (Gandrud, 2018). While most published methods effectively communicate research findings and persuade readers of their validity, they often fall short in fully conveying the context and specific procedures of the research (Mesirov, 2010).

To highlight the importance of reproducibility, and the challenges arising from incomplete methodological descriptions, variations in computational environments, and the availability of datasets, we will try to replicate as much as possible the bioinformatics downstream analysis that was performed in a study by Espinosa et al., (2020).

The study was aiming to expand their insights on CEN.PK 113-5D strain that was previously found to be a superior candidate to the other *S. cerevisiae* strain S288C, for methylotrophy engineering (Espinosa et al., 2019). *YGR067C* was identified as the only gene mutated in all 3 methanol exposed lineages but absent in those grown in just yeast extract (controls), (Espinosa et al., 2020). The mutation was reconstructed in the parental CEN.PK113-5D strain and was named reconstructed EC (ReEC). To characterise the effect of the *YGR067C* truncation in native methanol metabolism in *S. cerevisiae*, parental CEN.PK113-5D (control) and the ReEC strains were both cultured under the same methanol exposed conditions. Global transcript levels were compared to the parental strain.

Methods

Raw Data

The dataset is comprised of two groups of Saccharomyces cerevisiae strains; CEN.PK 113-5D (control) and the ReEC (mutated). Each group (control vs mutated) consists of two replicates that have been grown under the same environmental conditions. RNA-seq data files in fastq.gz format were downloaded from the sequence read archive (SRA) with project number PRJNA612896 to our group project directory in the CREATE HPC. Fasterq-dump tool from the SRA Toolkit (version 3.0.3) was used to fetch the SRA files to the HPC and convert them directly to FASTQ format. The --gzip function is not supported in fasterq-dump and thus, it was run after downloading and conversion had finished. --split-files was used to split the forward from the reverse reads data was in paired-end format. FastQC (version 0.12.1) was performed to confirm the quality scores of the RNA-seq data. A reference genome of S288C with GCF_000146045.2 accession number was downloaded with NCBI Datasets Command Line (CLI) tools (version 16.36.0). To install NCBI Datasets CLI tools, a conda environment was created in our group project directory in the HPC. Only FNA (sequence) and GTF (annotation) files were retained from the reference genome data files.

Alignment & Read Counts

A nano script to perform reference genome indexing and alignment of the sequencing reads using STAR aligner (version 2.7.10b) was created and was run in the HPC (see Appendix 1).

```
STAR --runThreadN 16  --runMode genomeGenerate --genomeDir /scratch_tmp/grp/msc_appbio/group1/ref_gen
--genomeFastaFiles/scratch_tmp/grp/msc_appbio/group1/ref_gen/GCF_000146045.2_R64_genomic.fna --sjdbGTFfile /scratch_tmp/grp/msc_appbio/group1/ref_gen/genomic.gtf
```

The FNA file from the reference genome was used for indexing, while the fastq.gz files for each pair of reads (forward and reverse) served as input for alignment. STAR mapped the reads to the indexed genome and generated a sorted BAM file as output, consolidating the paired reads into a single file.

```
STAR --runThreadN 6 --genomeDir "$ref_genome_index" --readFilesCommand zcat --readFilesIn "$rawDataDir${sample}_1.fastq.gz" "$rawDataDir${sample}_2.fastq.gz"
--outFileNamePrefix "$outputDataDir${sample}_" --outSAMtype BAM SortedByCoordinate
```

Aligned read counts were performed with featureCounts tool from the subread package (version 2.0.2) that was loaded in the HPC (see Appendix 1). featureCounts takes the sorted-BAM files as input and produces a TXT file with read counts for each sample (readCounts2.txt). The TXT file was transferred from the HPC to the local drive using SFTP for analysis in R (version 4.4.1).

```
featureCounts -p -a ${gtf_file}/*.gtf -g $feature_type -o ${output_Dir}/readCounts2.txt ${inputDir}/*.out.bam
```

Differential Gene Expression Analysis

Initial exploration of the data was performed with R build-in functions like *summary()*, *barplot()*, *hist()* and *plot()* for the observation of the distribution of our data. A script to perform differential gene expression (DGE) analysis was created and executed in R using the DESeq2 package (version 1.44.0). Our readCounts2.txt file was initially loaded into R and edited accordingly to rename sample IDs and remove redundant columns (see Appendix 1). *DESeq()* function was used to with default parameters for differential expression analysis (DEA). *results()* function was used to extract the results from DESeq analysis with initially setting an adjusted p-value (padj) cutoff at 0.01 and default lfcThreshold. A second run was performed where we set the lfcThreshold to 0.379 and we preserved our initial padj cutoff (see Appendix 1). The output results were filtered in both cases to remove any not applicable (N/A) values.

```
sigGenes = na.omit(sigGenes)
```

ID Mapping & Comparison

Transcriptome supplementary CSV file was retrieved from the paper and filtered to retrieve all the differentially expressed genes with padj ≤ 0.01. Our results of significantly differentially expressed gene were filtered against org.Sc.sgd.db (*Saccharomyces cerevisiae*) database for common IDs to the total of significantly differentially expressed gene IDs reported by the paper (see Appendix 1). This was performed for comparison purposes, since our gene ID types were different from the ones

```
sigGenes$COMMON = mapIds(org.Sc.sgd.db, keys = rownames(sigGenes), column = "COMMON", keytype = "REFSEQ", multiVals = "first")
```

reported by the paper.

Data Visualization

A subset of our significantly differentially expressed genes was created with the top 31 differentially expressed genes to be used for heatmap creation based on padj=0.01 and log2FoldChange=1.6 thresholds.

```
df.top = res.df[(abs(res.df$log2FoldChange) > 1.6) & (res.df$padj < 0.01),]
```

A heatmap was created with ComplexHeatmap (version 2.20.0) and the addition of RColorBrewer (version 1.1-3) for access to a wide colour palette, and circlize (0.4.16) for displaying relationships between variables in a circular layout, since the heatmap consisted of three columns. Volcano and PCA plots were produced with ggplot2 (version 3.5.1) and ggrepel (version 0.9.6) for customisation purposes (see Appendix 1).

GO Term Enrichment Analysis

Enrichment analysis for Biological Process (BP), Molecular Function (MF) and Cellular Components (CC) was performed in all our significantly differentially expressed genes with *enrichGO()* function from clusterProfiler (version 4.12.6), p-value cutoff of 0.05 and the Holm correction for multiple comparisons (see Appendix 1).

```
Upregulated_genesBP = enrichGO(gene = Upregulated_genes, OrgDb = "org.Sc.sgd.db", keyType = "ENTREZID", ont = "BP", pvalueCutoff = 0.05, pAdjustMethod = "holm")
Downregulated_genesBP = enrichGO(gene = Downregulated_genes, OrgDb = "org.Sc.sgd.db", keyType = "ENTREZID", ont = "BP", pvalueCutoff = 0.05, pAdjustMethod = "holm")
```

```
Upregulated_genesMF = enrichGO(gene = Upregulated_genes, OrgDb = "org.Sc.sgd.db", keyType = "ENTREZID", ont = "MF", pvalueCutoff = 0.05, pAdjustMethod = "holm")
Downregulated_genesMF = enrichGO(gene = Downregulated_genes, OrgDb = "org.Sc.sgd.db", keyType = "ENTREZID", ont = "MF", pvalueCutoff = 0.05, pAdjustMethod = "holm")
```

```
Upregulated_genesCC = enrichGO(gene = Upregulated_genes, OrgDb = "org.Sc.sgd.db", keyType = "ENTREZID", ont = "CC", pvalueCutoff = 0.05, pAdjustMethod = "holm")
Downregulated_genesCC = enrichGO(gene = Downregulated_genes, OrgDb = "org.Sc.sgd.db", keyType = "ENTREZID", ont = "CC", pvalueCutoff = 0.05, pAdjustMethod = "holm")
```

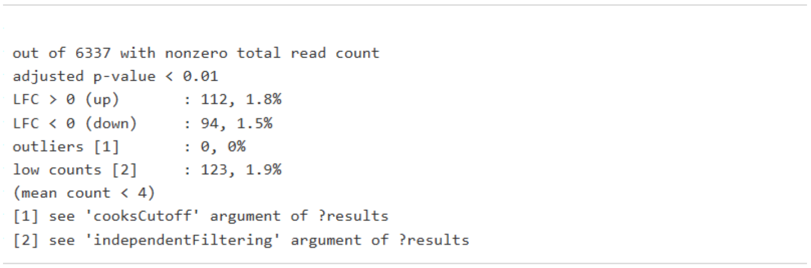
Pathway Enrichment Analysis

KEGG pathway enrichment analysis was performed for all our significantly differentially expressed genes against the KEGG pathway database for *S. cerevisiae* (organism = "sce"); with *enrichKEGG()* function from clusterProfiler, a p-value cutoff of 0.05 and the Holm correction for multiple comparisons (see Appendix 1).

```
Path_Upregulated_genes = enrichKEGG(gene = Upregulated_genes, organism = "sce", keyType = "ncbi-geneid", pvalueCutoff = 0.05, pAdjustMethod = "holm", minGSSize = 5)
Path_Downregulated_genes = enrichKEGG(gene = Downregulated_genes, organism = "sce", keyType = "ncbi-geneid", pvalueCutoff = 0.05, pAdjustMethod = "holm", minGSSize = 5)
```

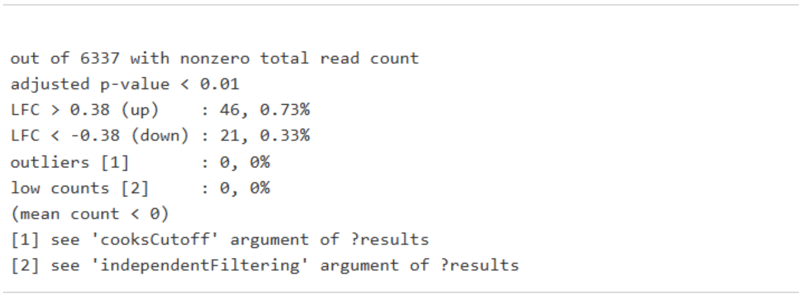
Results

DEA based on adjusted p-value cutoff of ≤0.01 and default log2FoldChange = 0 (log2FC) threshold, indicated 206 significantly differentially expressed genes of which 112 were upregulated and 94 were downregulated (Figure 1, 3). The same analysis but this time with a log2FC threshold of 0.379 or 1.3-Fold Change (FC), resulted in a total of 67 differentially expressed genes (padj≤0.01) of which 46 were upregulated (log2FC >0.379) and 21 were downregulated (log2FC <0.379), (see Figure 2, 4).



```
out of 6337 with nonzero total read count
adjusted p-value < 0.01
LFC > 0 (up)       : 112, 1.8%
LFC < 0 (down)     : 94, 1.5%
outliers [1]       : 0, 0%
low counts [2]     : 123, 1.9%
(mean count < 4)
[1] see 'cooksCutoff' argument of ?results
[2] see 'independentFiltering' argument of ?results
```

Figure 1: Differential expression results with padj = 0



```
out of 6337 with nonzero total read count
adjusted p-value < 0.01
LFC > 0.38 (up)     : 46, 0.73%
LFC < -0.38 (down) : 21, 0.33%
outliers [1]        : 0, 0%
low counts [2]      : 0, 0%
(mean count < 0)
[1] see 'cooksCutoff' argument of ?results
[2] see 'independentFiltering' argument of ?results
```

Figure 2: Differential expression results with padj = 0 and log2FC = 0.379

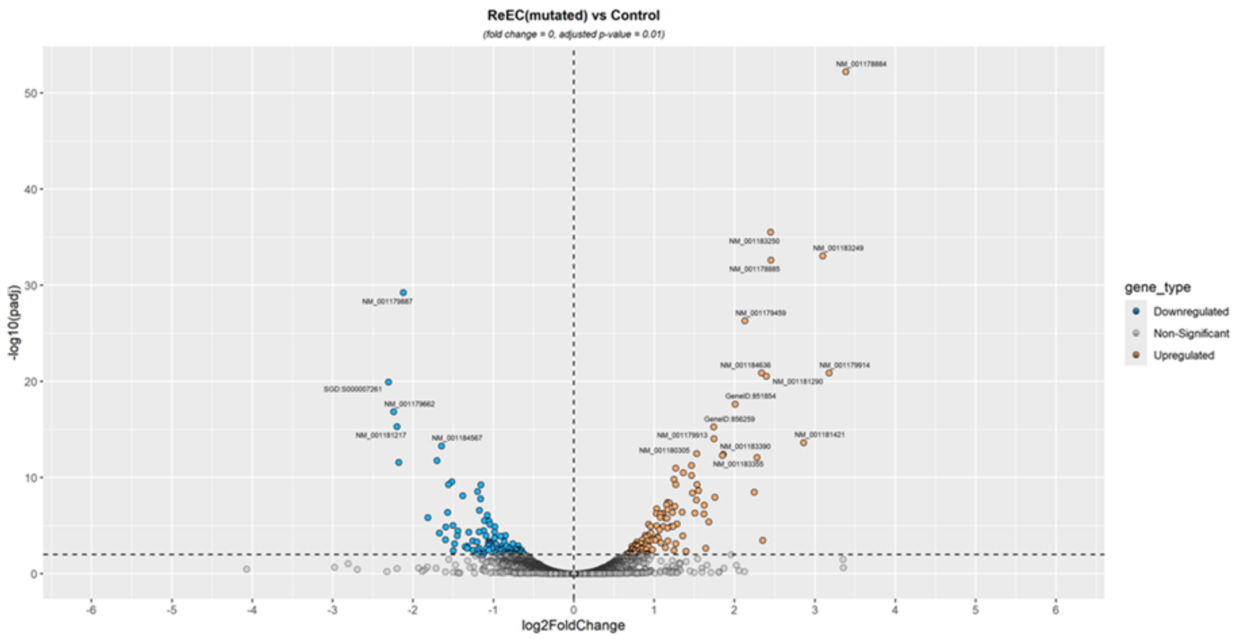


Figure 3: Volcano plot showing differentially expressed genes (coloured dots) that meet the threshold of ≤ -log10padj (horizontal dashed line) with padj=0.01. Not significantly differentially expressed genes are not coloured. Upregulated genes (orange dots) present a positive log2FC >0 and Downregulated genes (blue dots) present a negative log2FC <0. The vertical dashed line represents the log2FC =0 threshold.

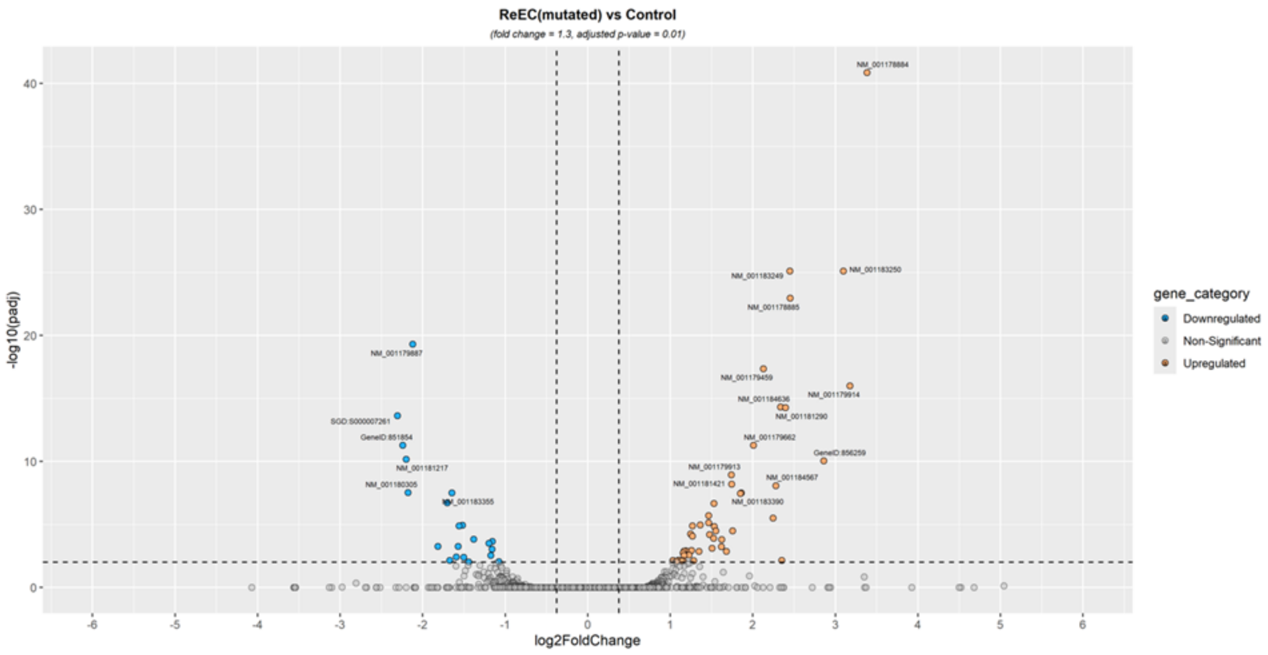


Figure 4: Volcano plot showing differentially expressed genes (coloured dots) that meet the threshold of ≤ -log10padj (horizontal dashed line) with padj=0.01. Not significantly differentially expressed genes are not coloured. Upregulated genes (orange dots) present a positive log2FC >0.379 or 1.3FC and Downregulated genes (blue dots) present a negative log2FC <0.379 or 1.3FC. The vertical dashed lines represent the log2FC = ±0.379 or ±1.3FC thresholds.

A heatmap was created of the top 31 differentially expressed genes (significant) to depict their Z-scores, log2FC and their average expression across all 4 samples (Figure 5). Notably, NM_001179914, NM_001178884 and NM_001183249 are highly upregulated genes with log2FC > 3. In contrast, S000007261, NM_001179662 and NM_001181217 are highly downregulated with log2FC > -2.

GO term enrichment analysis of the upregulated genes for biological processes indicated significant enrichment (p-value < 0.05) for ‘carbohydrate metabolic process’, ‘carbohydrate transmembrane transport’, ‘pyruvate metabolic process’ and ‘glycolytic process’ (Figure 6). GO term enrichment analysis of the downregulated genes for biological processes indicated significant enrichment (p-value < 0.05) for ‘cell wall organisation biogenesis’, ‘external encapsulating structure organisation’, ‘cellular respiration’ and ‘cell division’ (Figure 7).
KEGG pathway enrichment analysis for the upregulated genes showed significant enrichment (p-value < 0.05) for only three pathways namely ‘Fructose and mannose metabolism’, ‘Galactose metabolism’ and ‘Starch and sucrose metabolism’ (Figure 8). The same pathway enrichment analysis for downregulated genes indicted more significantly enriched pathways (p-value < 0.05) including ‘Carbon metabolism’, ‘Biosynthesis of secondary metabolites’, ‘TCA cycle’ and ‘Glyoxylate and dicarboxylate metabolism’ (Figure 9).

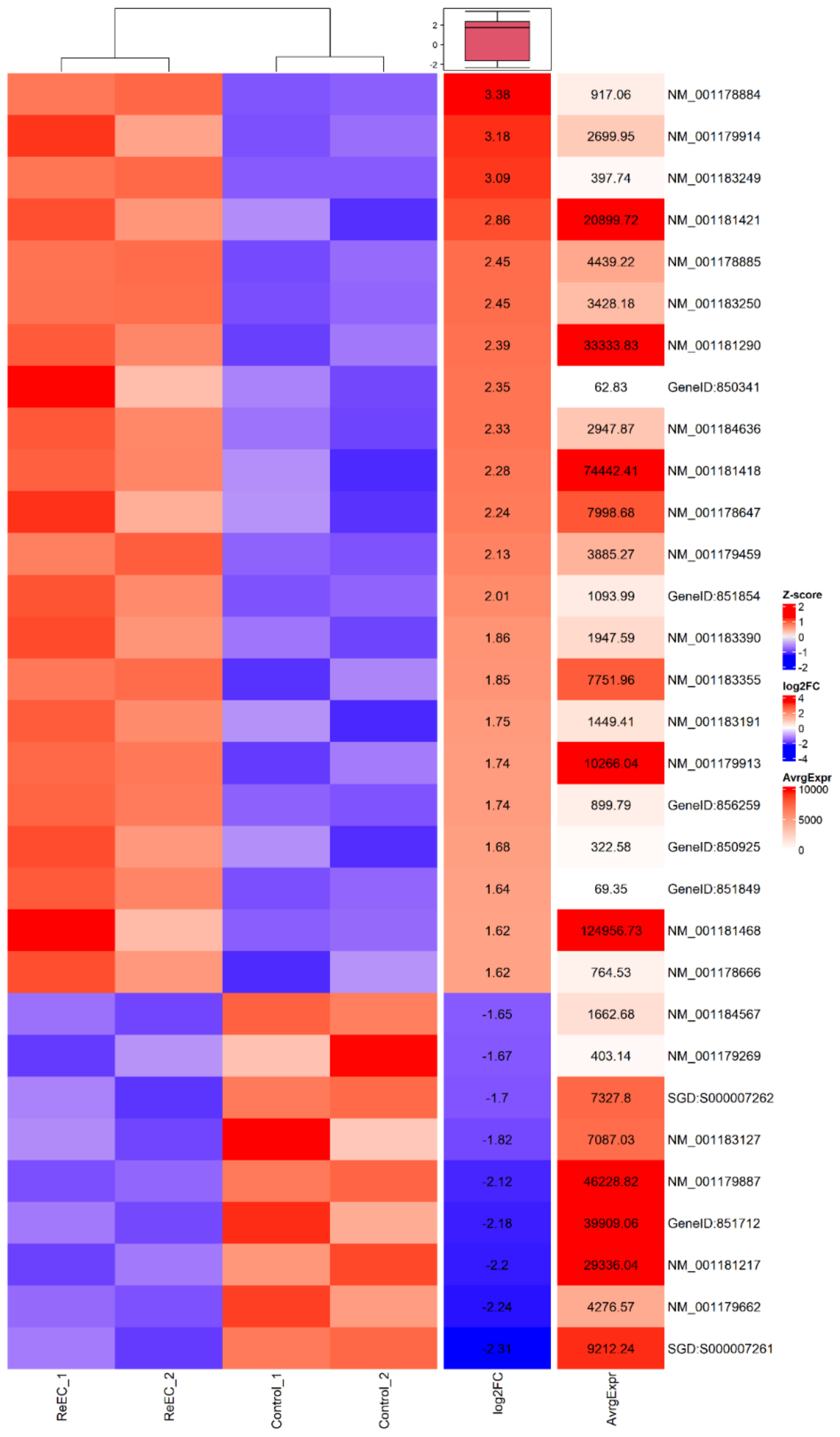


Figure 5: Heatmap displaying the expression levels of genes in ReEC (mutated) and Control groups. The cluster and colour coding are based on their Z-scores, log2FC and Average Expression across all 4 samples.

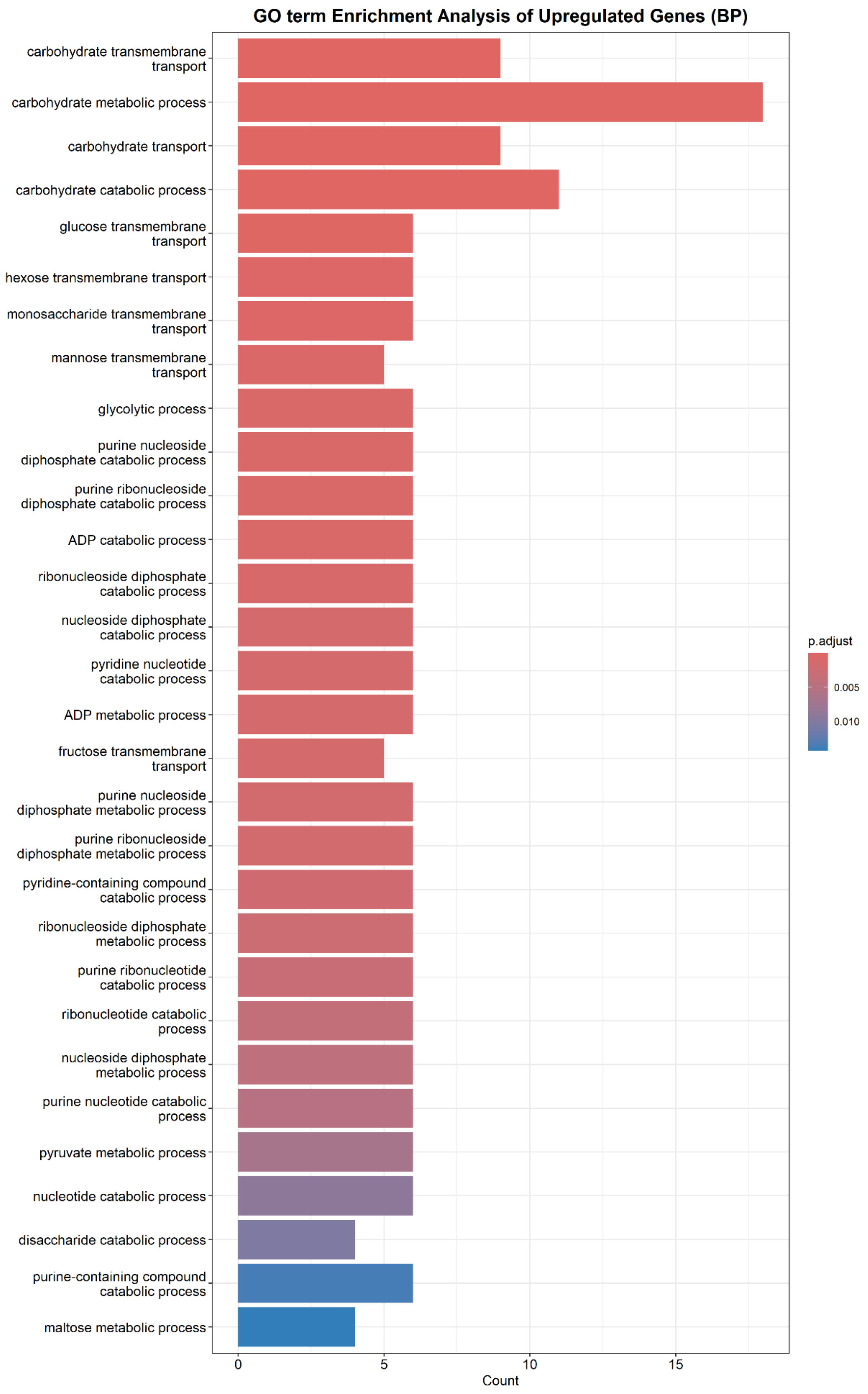


Figure 6: Bar plot indicating significantly (p-value < 0.05) enriched upregulated genes for biological processes.

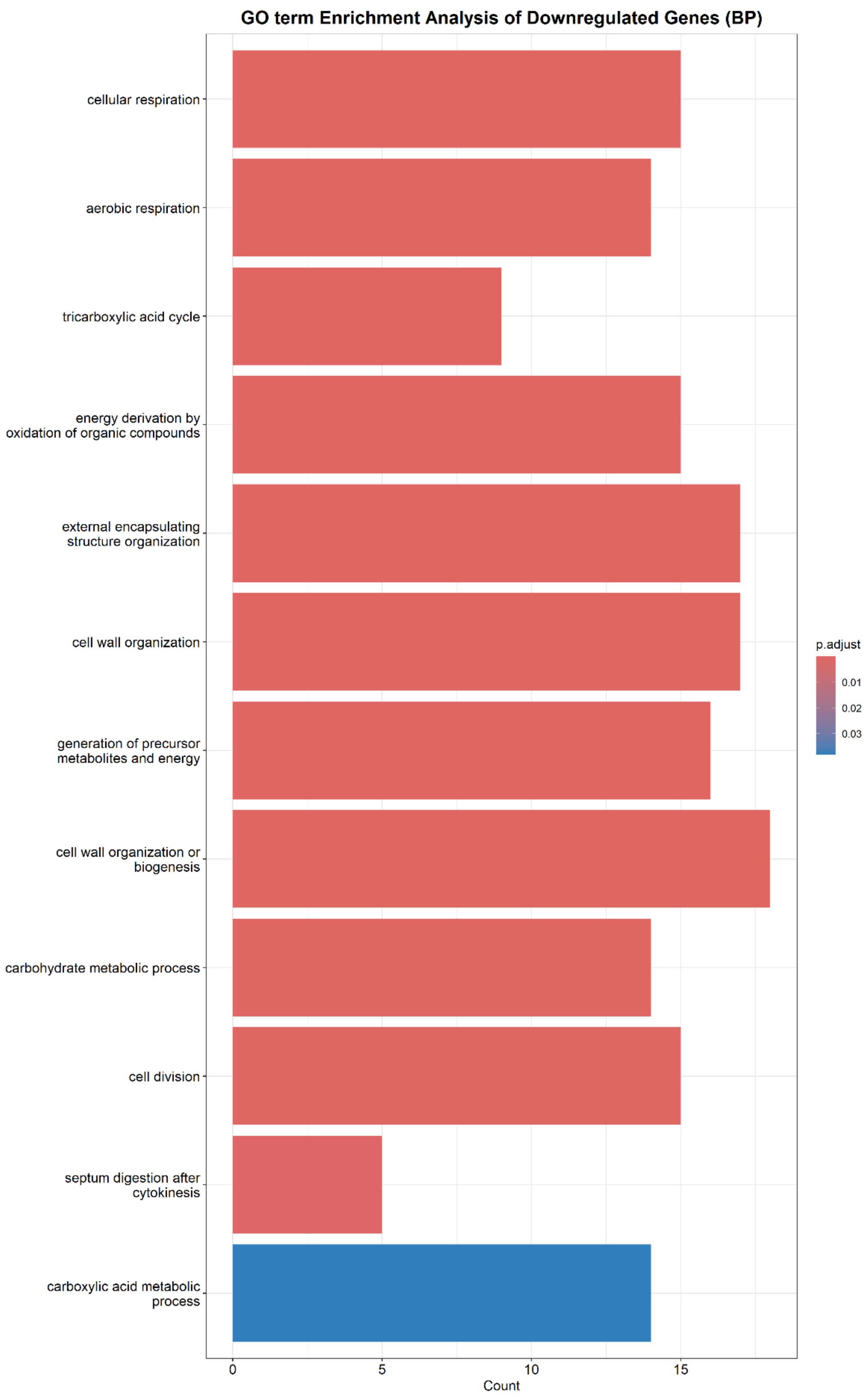


Figure 7: Bar plot indicating significantly (p-value < 0.05) enriched downregulated genes for biological processes.

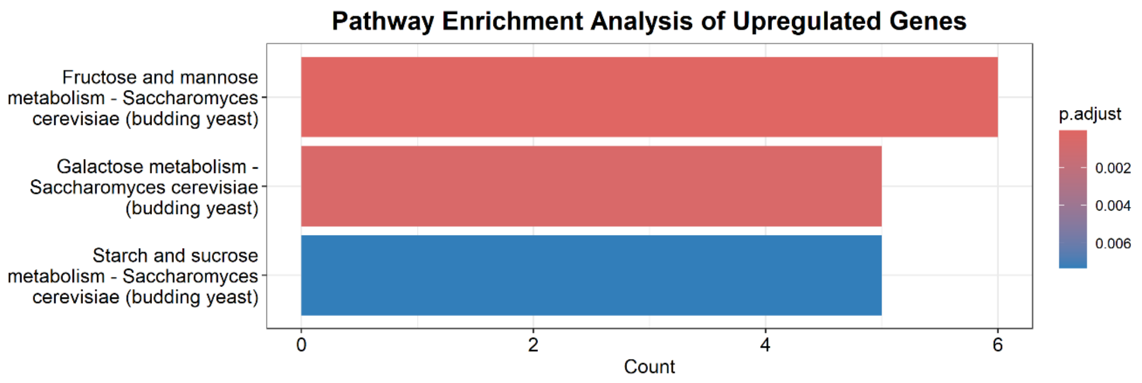


Figure 8: Bar plot indicating significantly (p-value < 0.05) enriched pathways of the upregulated genes based on KEGG pathway enrichment analysis.

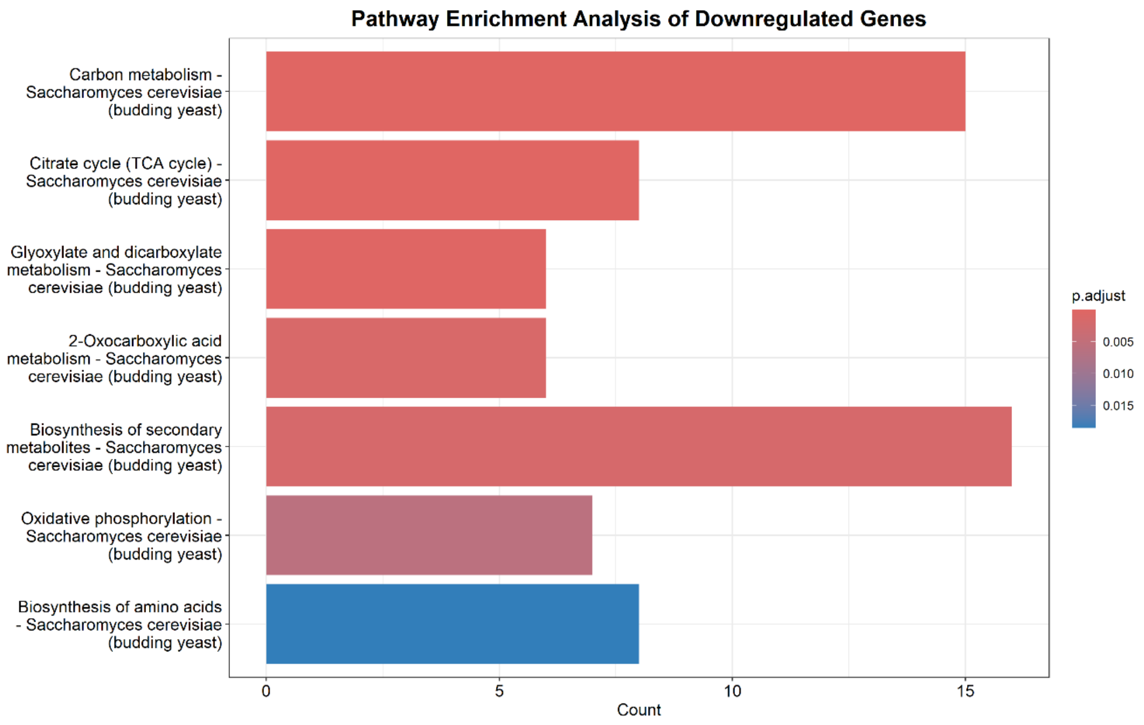


Figure 9: Bar plot indicating significantly (p-value < 0.05) enriched pathways of the downregulated genes based on KEGG pathway enrichment analysis.

Discussion

This study utilized Geneious Pro (version 11), and although the authors specified the use of default parameters, the availability of multiple updates within version 11 creates ambiguity regarding the precise configuration employed. The proprietary nature of this software further restricted access to its RNA algorithm. As a result, we opted to use STAR for alignment due to its open-source nature and comprehensive documentation. However, like any alignment tool, STAR is subject to its own inherent biases, which can significantly impact downstream analyses, raising concerns about the reproducibility of this workflow. Additionally, the inability to access the YeastMine database, coupled with minimal workflow documentation, undermines the principles of accessibility and reusability outlined in the FAIR guidelines (Wilkinson et al., 2016), further impacting the reproducibility of our findings

In terms of the DEA, the study reported 243 significantly differentially expressed genes, with 111 downregulated and 132 upregulated genes. However, analysis of their provided supplementary transcriptome file indicated that only 194 genes were actually meeting the padj=0.01 threshold that was set by the study for significantly differentially expressed genes. Our results of total significantly differentially expressed genes (206) indicated a much closer number to the ones in the supplementary file but the reason for this discrepancy remains unclear.

We are unsure if additional cutoffs were used in the study for DEA, since we are unaware of Geneious default settings. Nonetheless, based on what is typically followed by most studies when DEA is performed, a cutoff of at least 1.3FC is considered alongside the padj threshold for a more accurate prediction of the differentially expressed genes (McCarthy et al., 2009). Therefore, we also performed an additional DEA run with a cutoff of 1.3FC alongside the same padj threshold that indicated even less differentially expressed genes, as expected. Specifically, a total of 67 differentially expressed genes with 46 upregulated and 21 downregulated was observed.

Our pathway enrichment analysis of the upregulated genes revealed three significantly (p-value < 0.05) enriched pathways (Figure 8) unlike with the study where they reported none. However, this discrepancy might have been caused by the utilisation of different pathway databases for the enrichment analysis, since we used KEGG instead of YeastMine at the Saccharomyces Genome Database that was used in the study. This is because we were unable to access YeastMine as the platform was discontinued on July 15, 2024, due to funding cuts at the Saccharomyces Genome Database (SGD). Nonetheless, our results of biological process enrichment analysis for the upregulated genes agree with study's reported 'carbohydrate metabolic process', 'carbohydrate transmembrane transport', 'pyruvate metabolic process' and 'glycolytic process' as significantly enriched (p-value < 0.05), (Figure 6). KEGG pathway enrichment analysis for the downregulated genes also agrees to some extent with what was reported in the study even though we reported more significantly enriched pathways (Figure 9). Specifically, both TCA and Glyoxylate cycles were identified by KEGG enrichment analysis but not respiration.

Our study underscores the importance of reproducibility in computational biology and bioinformatics, while highlighting challenges posed by inaccessible proprietary software, incomplete methodological descriptions, and technical discrepancies. The inability to replicate the exact parameters of Geneious Pro significantly impacted DEA results, leading to discrepancies with the original study's findings. However, pathway enrichment analysis corroborated some of the biological insights from the study, affirming the robustness of alternative databases for pathway enrichment like KEGG. These results emphasize the need for transparent methods, accessible tools, and consistent standards to ensure reproducibility in bioinformatics research.

Supplementary material

Our documents were all can be accessed at:
https://github.com/NikolaosSamperis/group1_project7BBG1002.git

Contribution statement

Madiha Khan was responsible for downloading all the raw data and performing FastQC. She was also in charge of creating a script for alignments to the reference genome. Nikolaos Samperis was responsible for the production and visualisation of read counts from the mapped sequence reads. Yixin Huang contributed to the formatting of the final report. All authors contributed equally to the writing of the final report.

References

1. Baker M., (2016). 1,500 scientists lift the lid on reproducibility. Nature; vol. 533, pp. 452-454.
2. Cokelaer T., Cohen-Boulakia S. and Lemoine F., (2023). Reprohackathons: promoting reproducibility in bioinformatics through training. Bioinformatics; vol. 39(Supplement_1), pp. i11-i20.
3. Espinosa M.I., Gonzalez-Garcia R.A., Valgepea K., Plan M.R., et al., (2020). Adaptive laboratory evolution of native methanol assimilation in Saccharomyces cerevisiae. Nature Communications; vol. 11(1), art. no. 5564.
4. Espinosa M.I., Williams T.C., Pretorius I.S. and Paulsen I.T., (2019). Benchmarking two Saccharomyces cerevisiae laboratory strains for growth and transcriptional response to methanol. Synthetic and systems biotechnology; vol. 4(4), pp.180-188.
5. Gandrud C., (2018). Reproducible research with R and R studio. 2nd edn. New York: Chapman & Hall /CRC Press.
6. Goodman S.N., Fanelli D., & Ioannidis J.P., (2016). What does research reproducibility mean? Science translational medicine; vol. 8(341), p. 341ps12.
7. McCarthy D.J. and Smyth G.K., (2009). Testing significance relative to a fold-change threshold is a TREAT. Bioinformatics; vol. 25(6), pp. 765-771.
8. Mesirov J.P., (2010). Accessible reproducible research. Science; vol. 327(5964), pp. 415-416.
9. Wilkinson M.D., Dumontier M., Aalbersberg I.J., Appleton G., et al., (2016). The FAIR Guiding Principles for scientific data management and stewardship. Scientific data; vol. 3(1), pp. 1-9.

Appendix 1

Preparation for alignment

Downloading all the files, uploading the genome files from the local terminal, and setting up the Conda environment.

```
cd /scratch_tmp/grp/msc_appbio/group1/raw_reads2

wget -nc ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR113/068/SRR11318268/SRR11318268_2
.fastq.gz
wget -nc ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR113/070/SRR11318270/SRR11318270_1
.fastq.gz
wget -nc ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR113/070/SRR11318270/SRR11318270_2
.fastq.gz
wget -nc ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR113/071/SRR11318271/SRR11318271_2
.fastq.gz
wget -nc ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR113/069/SRR11318269/SRR11318269_2
.fastq.gz
wget -nc ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR113/071/SRR11318271/SRR11318271_1
.fastq.gz
wget -nc ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR113/068/SRR11318268/SRR11318268_1
.fastq.gz
wget -nc ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR113/069/SRR11318269/SRR11318269_1
.fastq.gz
```

```
sftp -i~/.ssh/panda k24090847@hpc.create.kcl.ac.uk
lcd /Users/madihakhan/Desktop/Group1project #Change the local directory to Desktop
lls #list files in directory

#Upload ref genome files (fna and GTF) from local directory to HPC directory
put /Users/madihakhan/Desktop/ref_genome/GCF_000146045.2/GCF_000146045
.2_R64_genomic.fna /scratch_tmp/grp/msc_appbio/group1
put /Users/madihakhan/Desktop/ref_genome/GCF_000146045.2/genomic.gtf /scratch_tmp
/grp/msc_appbio/group1
```

```
#Install tools from Bioconda
create conda --name myenv
conda activate myenv
conda install -c bioconda samtools fastqc star subread py-multiqc
```

The script for quality analysis of fastq.gz files

Doing the quality analysis of fastq.gz files to check the accuracy and reliability of downstream analyses, including alignment, variant detection, and gene expression analysis.

```
nano fastqc.sh

#!/bin/bash
echo "start of pipeline"

module load fastqc #use if the conda env does not activate

#Specify base and results directories
baseDirectory="/scratch_tmp/grp/msc_appbio/group1/raw_reads"
resultsDirectory="/scratch_tmp/grp/msc_appbio/group1/outputs/fastqc_output"

#create results directory  if it does not exist
mkdir -p "$resultsDirectory"

#Run FastQC on all .fastq files in baseDirectory
fastqc -o "$resultsDirectory" -t 4 "$baseDirectory"/*.fastq.gz

# Run multiqc
multiqc "$output_Dir"

echo "end of fastqc pipeline"

conda activate myenv
srun msc_appbio --pty /bin/bash
sbatch fastqc.sh
squeue -u k24090847

cd outputs
ls
cd fastqc_output/
```

The script for reference genome index

```
nano star_indexing.sh

#!/bin/bash
module load star #use if the conda env does not activate

STAR --runThreadN 16  --runMode genomeGenerate --genomeDir /scratch_tmp/grp
/msc_appbio/group1/ref_gen --genomeFastaFiles/scratch_tmp/grp/msc_appbio/group1
/ref_gen/GCF_000146045.2_R64_genomic.fna --sjdbGTFfile /scratch_tmp/grp/msc_appbio
/group1/ref_gen/genomic.gtf
#ensure line 131 is run on a single line for alignment to work
```

The STAR alignment script for paired-end RNA reads

```
#!/bin/bash

echo "start of pipeline"
module load star #use if the conda env does not activate

# Specify path to raw data
baseDir="/scratch_tmp/grp/msc_appbio/group1"

# Define path to STAR indexed reference genome
ref_genome_index="${baseDir}/ref_gen/starindex"

rawDataDir="${baseDir}/raw_reads2/"
outputDataDir="${baseDir}/outputs/star_outs/"

# Create output directory if it doesn't exist
mkdir -p "$outputDataDir"

samples=$(ls "$rawDataDir"/*_1.fastq.gz | sed 's/_1.fastq.gz//' | xargs -n 1
basename)

#loop where samples are aligned using STAR and the genome index, saving the outputs
as sorted BAM file.
for sample in $samples; do #iterates through each item in the samples variable
        echo "Processing sample: $sample" #prints a message indicating which sample
is being processed

STAR --runThreadN 6 --genomeDir "$ref_genome_index" --readFilesCommand zcat
--readFilesIn "$rawDataDir${sample}_1.fastq.gz" "$rawDataDir${sample}_2.fastq.gz"
--outFileNamePrefix "$outputDataDir${sample}_" --outSAMtype BAM SortedByCoordinate
#ensure line 160 is run on a single line for alignment to work

done
echo "STAR alignment complete! YAY"
```

The script for quantification of aligned reads using feature counts

```
#!/bin/bash

module load subread #use if the conda env does not activate
module load py-multiqc #use if the conda env does not activate

# Define variables and paths
baseDir="/scratch_tmp/grp/msc_appbio/group1"
inputDir="${baseDir}/outputs/star_outs"
gtf_file="${baseDir}/ref_gen/raw_refgen/GCF_000146045.2"
output_Dir="${baseDir}/outputs/readCounts"
feature_type="db_xref"

# Create output directory if it doesn't exist
mkdir -p "$output_Dir"

# Run featureCounts tool

featureCounts -p -a ${gtf_file}/*.gtf -g $feature_type -o ${output_Dir}/readCounts2
.txt ${inputDir}/*.out.bam

# Check if featureCounts ran successfully
if [[ $? -eq 0 ]]; then
  echo "featureCounts completed successfully. Results are in $output_Dir"
else
  echo "Error: featureCounts failed. Check your input files and parameters."
exit 1
fi

# Run multiqc
multiqc "$output_Dir"

# Check if MultiQC ran successfully
if [[ $? -eq 0 ]]; then
    echo "multiqc completed successfully"
else
    echo "Error: multiqc failed"
exit 1
fi
```

Preparation for DGE

Installing and library all the packages.

```
{r eval=FALSE, include=FALSE}
install.packages(c('BiocManager', 'dplyr', 'gplots', 'ggplot2', 'ggrepel'))
```

```
BiocManager::install(c('limma', 'DESeq2', 'AnnotationDbi', 'ReportingTools', 'GO.db', 'GOstats', 'pathview', 'gage',
'gageData', 'Select', 'ComplexHeatmap', 'EnhancedVolcano', 'clusterProfiler', 'org.Sc.sgd.db'))
```

```r
# Loading libraries.

# For differential gene expression analysis of RNA-seq data.
library(DESeq2)

# A package for data manipulation and transformation.
library(dplyr)

# A popular package for data visualization in R.
library(ggplot2)

# Provides various plotting functions, with a focus on heatmaps and other complex visualizations.
library(gplots)

 # Specifically designed for creating complex heatmaps with rich annotations.
library(ComplexHeatmap)

# Provides a collection of color palettes that are suitable for various types of data visualizations.
library(RColorBrewer)

# Used for circular visualizations, especially for displaying relationships between variables in a circular layout
library(circlize)

# Provides methods for performing Gene Ontology (GO) statistical tests and enrichment analysis.
library(GOstats)

# A database package providing GO (Gene Ontology) annotations.
library(GO.db)

# Used for statistical analysis and visualization of functional profiles (e.g., GO, KEGG) of genes, proteins, or
metabolites.
library(clusterProfiler)
```

```r
# A framework for storing, querying, and retrieving annotation data, such as gene symbols, IDs, and GO terms.
library(AnnotationDbi)

# An organism-specific annotation database, in this case, for *Saccharomyces cerevisiae* (yeast).
library(org.Sc.sgd.db)
```

Data Exploration

Draw the figures for visualization to find out the quality of data.

Create barplot.

```r
# Fix margins of our output.
par(mar=c(8,5,4,2)+0.1)

# Create barplot.
barplot(colSums(countData)/1e6, main = "Total Read Counts", xlab = "Samples", ylab = "Counts per Million")
```

Create histogram.

```r
# Fix margins of our output.
par(mar=c(7,5,4,4)+0.2)

# Create histogram.
hist(countData$ReEC_2, xlim = c(0,120000), main = "Histogram of read counts for ReEC_2", xlab = "Number of read counts",
br=1000)
```

Apply log2FoldChange for our sample read counts in order to get better visualization.

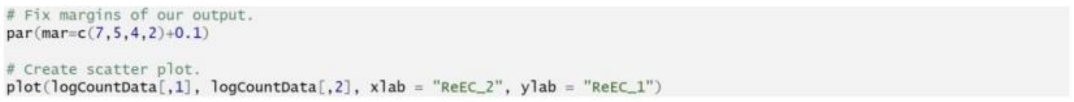
```r
logCountData = log2(1 + countData)
```

```r
# Fix margins of our output.
par(mar=c(7,5,4,4)+0.2)

# Create histogram with the log2 read counts
hist(logCountData$ReEC_2, xlim = c(0,20), main = "Histogram of read counts for ReEC_2", xlab = "Number of read counts",
br=100)
```

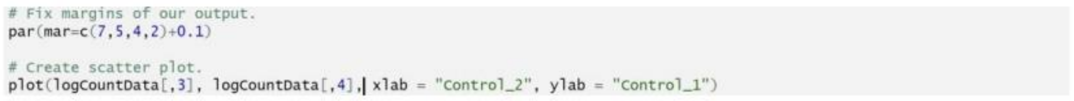
Do the barplot again with the log2FoldChange data.

```r
# Fix margins of our output.
par(mar=c(7,5,4,4)+0.2)

# Create histogram with the log2 read counts
hist(logCountData$ReEC_2, xlim = c(0,20), main = "Histogram of read counts for ReEC_2", xlab = "Number of read counts",
br=100)
```

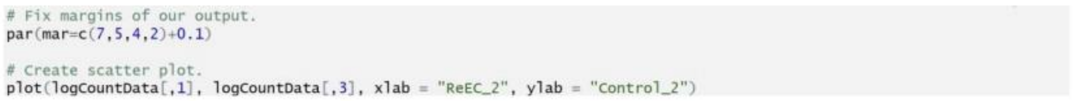
Plotting Scatter Plot between experimental groups, control groups and both of them seperatly.



```r
# Fix margins of our output.
par(mar=c(7,5,4,2)+0.1)

# Create scatter plot.
plot(logCountData[,1], logCountData[,2], xlab = "ReEC_2", ylab = "ReEC_1")
```



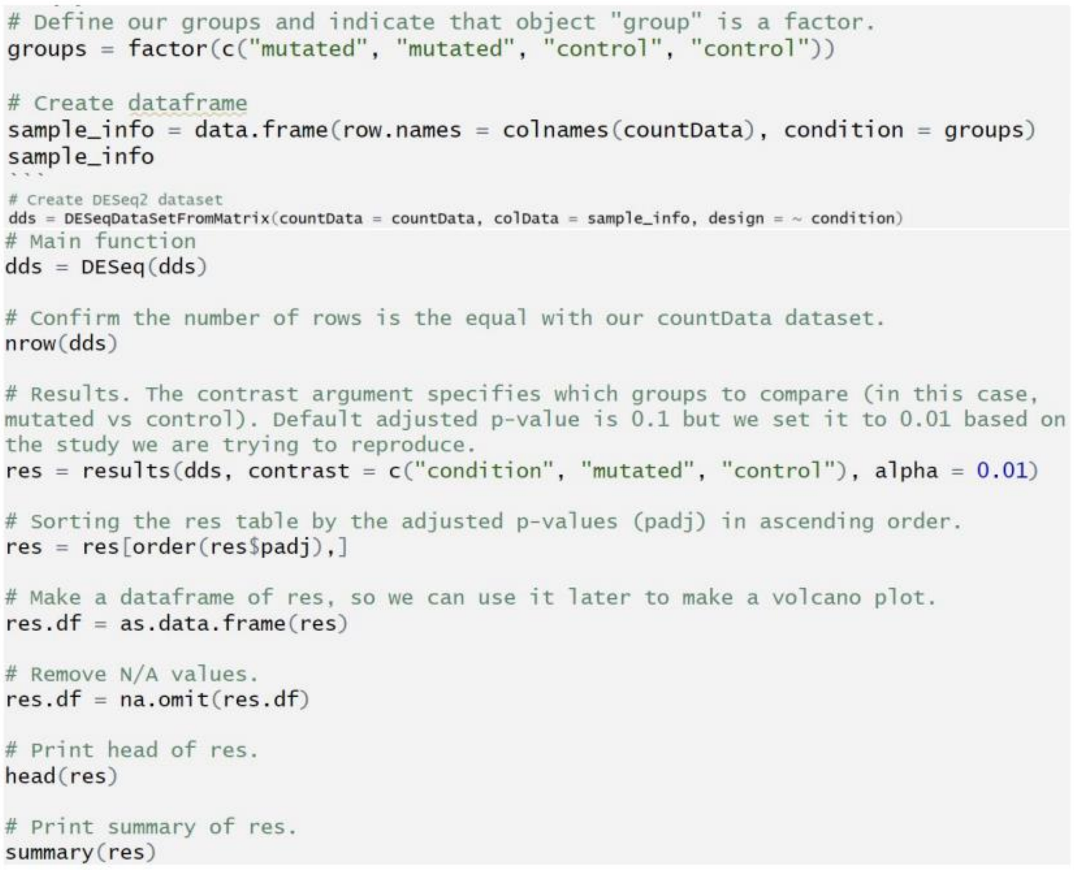
```r
# Fix margins of our output.
par(mar=c(7,5,4,2)+0.1)

# Create scatter plot.
plot(logCountData[,3], logCountData[,4], xlab = "Control_2", ylab = "Control_1")
```



```r
# Fix margins of our output.
par(mar=c(7,5,4,2)+0.1)

# Create scatter plot.
plot(logCountData[,1], logCountData[,3], xlab = "ReEC_2", ylab = "Control_2")
```

DESeq Normalization



~~~~r
# Define our groups and indicate that object "group" is a factor.
groups = factor(c("mutated", "mutated", "control", "control"))

# Create dataframe
sample_info = data.frame(row.names = colnames(countData), condition = groups)
sample_info
```
# Create DESeq2 dataset
dds = DESeqDataSetFromMatrix(countData = countData, colData = sample_info, design = ~ condition)
# Main function
dds = DESeq(dds)

# Confirm the number of rows is the equal with our countData dataset.
nrow(dds)

# Results. The contrast argument specifies which groups to compare (in this case,
mutated vs control). Default adjusted p-value is 0.1 but we set it to 0.01 based on
the study we are trying to reproduce.
res = results(dds, contrast = c("condition", "mutated", "control"), alpha = 0.01)

# Sorting the res table by the adjusted p-values (padj) in ascending order.
res = res[order(res$padj),]

# Make a dataframe of res, so we can use it later to make a volcano plot.
res.df = as.data.frame(res)

# Remove N/A values.
res.df = na.omit(res.df)

# Print head of res.
head(res)

# Print summary of res.
summary(res)
~~~~

Visualization of DGE results

Generate volcano plots

```r
# Create vectors of custom colors and alphas (transparency) to volcano plot.
cols2 = c("Upregulated" = "#ffad73", "Downregulated" = "#26b3ff", "Non-Significant" = "grey")
alphas2 = c("Upregulated" = 1, "Downregulated" = 1, "Non-Significant" = 0.5)

# Create a column in res.strict.df2 that takes the rownames of our 20 most significant differentially expressed genes that will be used later for labeling
purposes. This is possible as res.strict.df2 is sorted in ascending order of the padj values.
res.strict.df2 = res.strict.df2 %>% mutate(top_20genes = ifelse(row_number() <= 20, rownames(res.df2), NA))

# Create Volcano plot in the same way we did in the previous steps.
VolcanoPlot_strict = ggplot( res.strict.df2, aes(x = log2FoldChange,
             y = -log10(padj),
             fill = gene_category,
             size = gene_category,
             alpha = gene_category)) +
  geom_point(shape = 21,
             colour = "black", size = 2) +  # Plot the points and modify color and size.
  geom_hline(yintercept = -log10(0.01),
             linetype = "dashed") + # Add dashed line in the y-axis at -log(0.01).
  geom_vline(xintercept = c(log2(1.3), -log2(1.3)),
             linetype = "dashed") + # Add dashed line in the x-axis at log2(1.3) and -log2(1.3) fold change respectively.
  scale_fill_manual(values = cols2) + # Modify point color
  scale_alpha_manual(values = alphas2) + # Modify point transparency
  scale_x_continuous(breaks = c(seq(-6, 6, 1)), # Modify x-axis tick intervals.
                 limits = c(-6, 6)) + # Modify x-axis range from -6 to 6.
  ggtitle("ReEC(mutated) vs Control") + # Add title.
  theme(plot.title = element_text(size = 11, face = "bold", hjust = 0.5)) + # Format title.
  labs(subtitle = "(fold change = 1.3, adjusted p-value = 0.01)") + # Add subtitle.
  theme(plot.subtitle = element_text(size = 8, face = "italic", hjust = 0.5)) + # Formt subtitle.
  geom_text_repel(aes(label = top_20genes), # Add labels to the top 20 most significant differentially expressed genes.
              size = 2,
              max.overlaps = Inf)

# Extracting VolcanoPlot_strict into a PNG file and saving it to our current directory.
png("./VolcanoPlot_strict.png", res = 300, width = 4100, height = 2100)

# Renders the VolcanoPlot_strict and writes it to the PNG file.
print(VolcanoPlot_strict)

# "dev.off()" closes the graphics device, freeing up memory.
dev.off()
```

Generate PCA plot

```r
# Perform PCA to the regularized log transformed data.
pca.object = prcomp(t(assay(rld)))


 # Data preparation for plotting.

# Select the first two columns of pca.object which contains the principal component scores.
pcaData = as.data.frame(pca.object$x[,1:2]);

# Add groups as a "Type" column in pcaData
pcaData = cbind(pcaData, detectGroups(colnames(assay(rld)) ))

# Add column names
colnames(pcaData) = c("PC1", "PC2", "Type")

# Calculate the percentage of variance explained by the first two principal components (PC1 and PC2). Use the summary(pca.object)$importance table to get
the proportion of variance explained for each principal component and convert it to a percentage.
percentVar=round(100*summary(pca.object)$importance[2,1:2],0)


 # Plotting the PCA Results

# Initialize the ggplot object with the PCA data. Color and shape the points by the Type variable (which indicates the sample group).
p=ggplot(pcaData, aes(PC1, PC2, color=Type, shape = Type)) + geom_point(size=3)

# Label the x-axis and y-axis respectively with the percentages of variance explained by PC1 and PC2.
p=p+xlab(paste0("PC1: ",percentVar[1],"% variance"))
p=p+ylab(paste0("PC2: ",percentVar[2],"% variance"))

# Add title and ensure that the aspect ratio of the plot is equal, so the axes have the same scale.
p=p+ggtitle("Principal component analysis (PCA)") + coord_fixed(ratio=1.0) +
    theme(plot.title = element_text(size = 12,hjust = 0.5)) + theme(aspect.ratio = 1) +
    theme(axis.text.x = element_text( size = 10),
    axis.text.y = element_text( size = 10),
    axis.title.x = element_text( size = 11),
    axis.title.y = element_text( size = 11) ) +
  theme(legend.text=element_text(size=11))

# Display plot
print(p)
```

Preparation for GO Term Enrichment Analysis and Pathway Enrichment Analysis

Adding four extra columns in sigGenes for **ENTREZ**, **GENENAME**, **ENSEMBL**, and **GO IDs** to ensure compatibility with different databases and tools. These IDs can be mapped to our row names (RefSeq ID type) using org.Sc.sgd.db.

```r
# Make ENTREZ column and map RefSeq IDs to ENTREZ from org.Sc.sgd.db.
sigGenes$ENTREZ = mapIds(org.Sc.sgd.db, keys = sigGenes$RefSeq, column = "ENTREZID", keytype = "REFSEQ", multiVals = "first")

# Make GENENAME column and map RefSeq IDs to GENENAME from org.Sc.sgd.db.
sigGenes$GENENAME = mapIds(org.Sc.sgd.db, keys = sigGenes$RefSeq, column = "GENENAME", keytype = "REFSEQ", multiVals = "first")

# Make ENSEMBL column and map RefSeq IDs to ENSEMBL from org.Sc.sgd.db.
sigGenes$ENSEMBL = mapIds(org.Sc.sgd.db, keys = sigGenes$RefSeq, column = "ENSEMBL", keytype = "REFSEQ", multiVals = "first")

# Make GO column and map RefSeq IDs to GO from org.Sc.sgd.db.
sigGenes$GO = mapIds(org.Sc.sgd.db, keys = sigGenes$RefSeq, column = "GO", keytype = "REFSEQ", multiVals = "first")
```

Identifying upregulated and downregulated genes based on the log2FoldChange thresholds of (-0.379, 0.379), in order to intuitively represent the relative changes in gene expression. Extracting the corresponding ENTREZ ID aims to perform enrichment analysis including GO analysis and KEGG pathway analysis to ensure compatibility across the different databases and tools.

```r
# Filter upregulated genes from sigGenes if log2FoldChange >= 0.379 and select the ENTREZ column from sigGenes. Store this to Upregulated_genes object.
Upregulated_genes = sigGenes[sigGenes$log2FoldChange >= 0.379, "ENTREZ"]

# Remove NA values.
Upregulated_genes = na.omit(Upregulated_genes)

# Filter downregulated genes from sigGenes if log2FoldChange <= -0.379 and select the ENTREZ column from sigGenes. Store this to Downregulated_genes object
.
Downregulated_genes = sigGenes[sigGenes$log2FoldChange <= (-0.379), "ENTREZ"]

# Remove NA values.
Downregulated_genes = na.omit(Downregulated_genes)
```

Visualization of GO Term Enrichment results

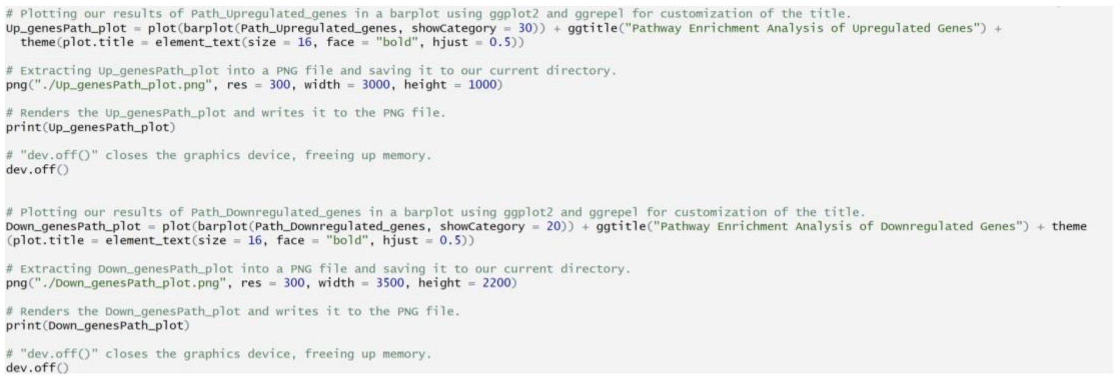
Plotting all three figures (BP, MF, and CC) with the ggplot2 and ggrepel.

```r
# Plotting our results of Upregulated_genesBP in a barplot using ggplot2 and ggrepel for customization of the title.
Up_genesBP.plot = plot(barplot(Upregulated_genesBP, showCategory = 30)) + ggtitle("GO term Enrichment Analysis of Upregulated Genes (BP)") +
  theme(plot.title = element_text(size = 16, face = "bold", hjust = 0.5))

# Extracting Up_genesBP.plot into a PNG file and saving it to our current directory.
png("./Up_genesBP.png", res = 300, width = 3200, height = 5100)

# Renders the Up_genesBP.plot and writes it to the PNG file.
print(Up_genesBP.plot)

# "dev.off()" closes the graphics device, freeing up memory.
dev.off()

# Plotting our results of Downregulated_genesBP in a barplot using ggplot2 and ggrepel for customization of the title.
Down_genesBP.plot = plot(barplot(Downregulated_genesBP, showCategory = 20)) + ggtitle("GO term Enrichment Analysis of Downregulated Genes (BP)") +
  theme(plot.title = element_text(size = 16, face = "bold", hjust = 0.5))

# Extracting Down_genesBP.plot into a PNG file and saving it to our current directory.
png("./Down_genesBP.png", res = 300, width = 3200, height = 5100)

# Renders the Down_genesBP.plot and writes it to the PNG file.
print(Down_genesBP.plot)

# "dev.off()" closes the graphics device, freeing up memory.
dev.off()
```

```r
# Plotting our results of Upregulated_genesMF in a barplot using ggplot2 and ggrepel for customization of the title.
Up_genesMF.plot = plot(barplot(Upregulated_genesMF, showCategory = 30)) + ggtitle("GO term Enrichment Analysis of Upregulated Genes (MF)") +
  theme(plot.title = element_text(size = 16, face = "bold", hjust = 0.5))

# Extracting Up_genesMF.plot into a PNG file and saving it to our current directory.
png("./Up_genesMF.png", res = 300, width = 3200, height = 5100)

# Renders the Up_genesMF.plot and writes it to the PNG file.
print(Up_genesMF.plot)

# "dev.off()" closes the graphics device, freeing up memory.
dev.off()

# Plotting our results of Downregulated_genesMF in a barplot using ggplot2 and ggrepel for customization of the title.
Down_genesMF.plot = plot(barplot(Downregulated_genesMF, showCategory = 20)) + ggtitle("GO term Enrichment Analysis of Downregulated Genes (MF)") +
  theme(plot.title = element_text(size = 16, face = "bold", hjust = 0.5))

# Extracting Down_genesMF.plot into a PNG file and saving it to our current directory.
png("./Down_genesMF.png", res = 300, width = 3200, height = 5100)

# Renders the Down_genesMF.plot and writes it to the PNG file.
print(Down_genesMF.plot)

# "dev.off()" closes the graphics device, freeing up memory.
dev.off()
Down_genesCC.plot = plot(barplot(Downregulated_genesCC, showCategory = 20)) + ggtitle("GO term Enrichment Analysis of Downregulated Genes (CC)") +
  theme(plot.title = element_text(size = 16, face = "bold", hjust = 0.5))

# Extracting Down_genesCC.plot into a PNG file and saving it to our current directory.
png("./Down_genesCC.png", res = 300, width = 3200, height = 5100)

# Renders the Down_genesCC.plot and writes it to the PNG file.
print(Down_genesCC.plot)

# "dev.off()" closes the graphics device, freeing up memory.
dev.off()
```

Visualization of KEGG pathway enrichment results

Plotting both upregulated genes and downregulated genes with the ggplot2 and ggrepel.



```r
# Plotting our results of Path_Upregulated_genes in a barplot using ggplot2 and ggrepel for customization of the title.
Up_genesPath_plot = plot(barplot(Path_Upregulated_genes, showCategory = 30)) + ggtitle("Pathway Enrichment Analysis of Upregulated Genes") +
  theme(plot.title = element_text(size = 16, face = "bold", hjust = 0.5))

# Extracting Up_genesPath_plot into a PNG file and saving it to our current directory.
png("./Up_genesPath_plot.png", res = 300, width = 3000, height = 1000)

# Renders the Up_genesPath_plot and writes it to the PNG file.
print(Up_genesPath_plot)

# "dev.off()" closes the graphics device, freeing up memory.
dev.off()


# Plotting our results of Path_Downregulated_genes in a barplot using ggplot2 and ggrepel for customization of the title.
Down_genesPath_plot = plot(barplot(Path_Downregulated_genes, showCategory = 20)) + ggtitle("Pathway Enrichment Analysis of Downregulated Genes") + theme
(plot.title = element_text(size = 16, face = "bold", hjust = 0.5))

# Extracting Down_genesPath_plot into a PNG file and saving it to our current directory.
png("./Down_genesPath_plot.png", res = 300, width = 3500, height = 2200)

# Renders the Down_genesPath_plot and writes it to the PNG file.
print(Down_genesPath_plot)

# "dev.off()" closes the graphics device, freeing up memory.
dev.off()
```