RNAseq Analysis - Final Project Report

Bioinformatics I

Alessandro Giulivo

12/7/2021

Table of Contents

Intro

This project of RNAseq Analysis is based on RNA sequencing data coming from the Regulation of transcription elongation in response to osmostress paper published by PLOS GENETICS on November 17th 2017.

The sequencing was performed on 12 samples of **Saccharomyces Cerevisiae yeast**, in particular there were 3 replicates for each of four different conditions:

- *Wild-type strains in normal conditions*;
- *Wild-type strains in osmostress conditions;*
- *Mutant strains in normal conditions;*
- *Mutant strains in osmostress conditions.*

The programs used for this bioinformatics analysis are RStudio (R and RMarkdown) and a Xubuntu machine installed within Oracle VirtualBox.
A **conda "RNAseq" environment** was created and activated in the latter and these softwares were installed:

- FastQC;
- MultiQC;
- Trim Galore;
- Salmon.

1 - Downloading the data

SRA Accession List

The first file to be downloaded is the list of the **Run IDs** of our 12 samples. It is found in the **Accession List** page inside the Gene Expression Omnibus (GEO) database, where all the data from the study have been deposited.

This is our `SRR_Acc_List.txt` file:

```
cat(readLines('SRR_Acc_List.txt'), sep = '\n')
```

```
## SRR5486478
## SRR5486479
## SRR5486480
## SRR5486481
## SRR5486482
## SRR5486483
## SRR5486484
## SRR5486485
## SRR5486486
## SRR5486487
## SRR5486488
## SRR5486489
```

Data from SRA

After having downloaded the Accession List, we can use it in our Linux machine in order to download the files of our 12 samples from SRA.
First we download each sample using this `sra-toolkit` command:

```
prefetch SRRxxx -O .
```

Then we need to convert the downloaded `sra` files to `fastq`, using:

```
fastq-dump --gzip SRRxxx.sra
```

If we want to download and convert all the 12 files only in one step, the commands we need to use are:

```
cat SRR_Acc_List.txt | xargs prefetch -O . #for download
```

and

```
fastq-dump --gzip *.sra #for conversion
```

Already processed data

We also need to download the raw counts of *already processed data*, which will be useful in the 7th Step of our analysis. This `"GSE98352_DESeq2_raw_counts.tsv"` file can be downloaded from the same GEO database page we used for the download of the SRA Accession List

2 - FastQC

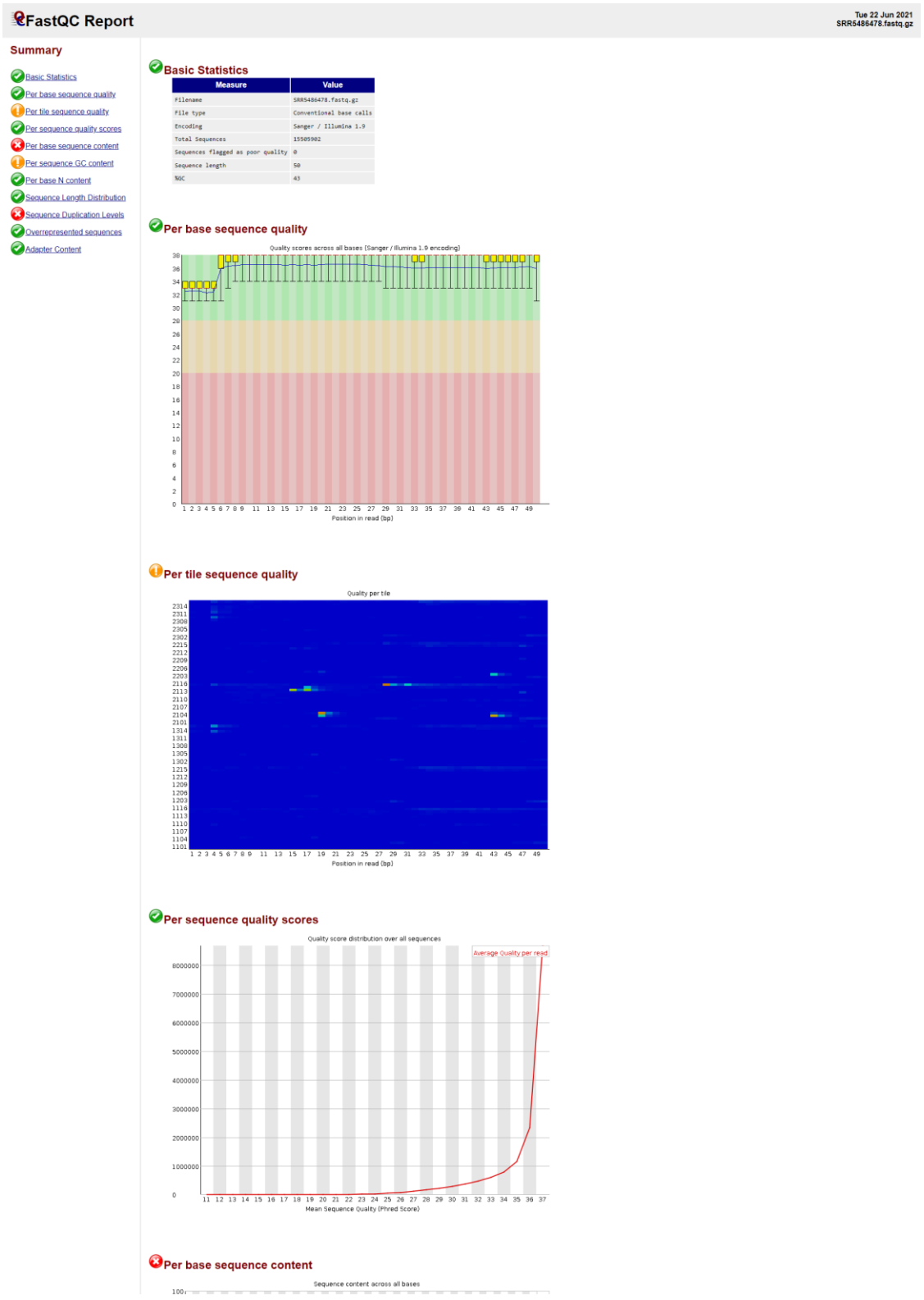
Now that we have the data we can start our RNA-seq analysis with a **quality control** step of our reads. We can do that by using **FastQC**, a tool which will analyse the quality of our files and output a report containing graphs and statistics for each of them.

So first we need to create a dedicated folder for the output reports using the command `mkdir fastqc_reports`, and then we can run FastQC on all our files:

```
fastqc -o fastqc_reports *.fastq.gz
```

FastQC Report

This is what the ***FastQC reports*** will look like:



FastQC report for SRR5486478.fastq

3 - MultiQC

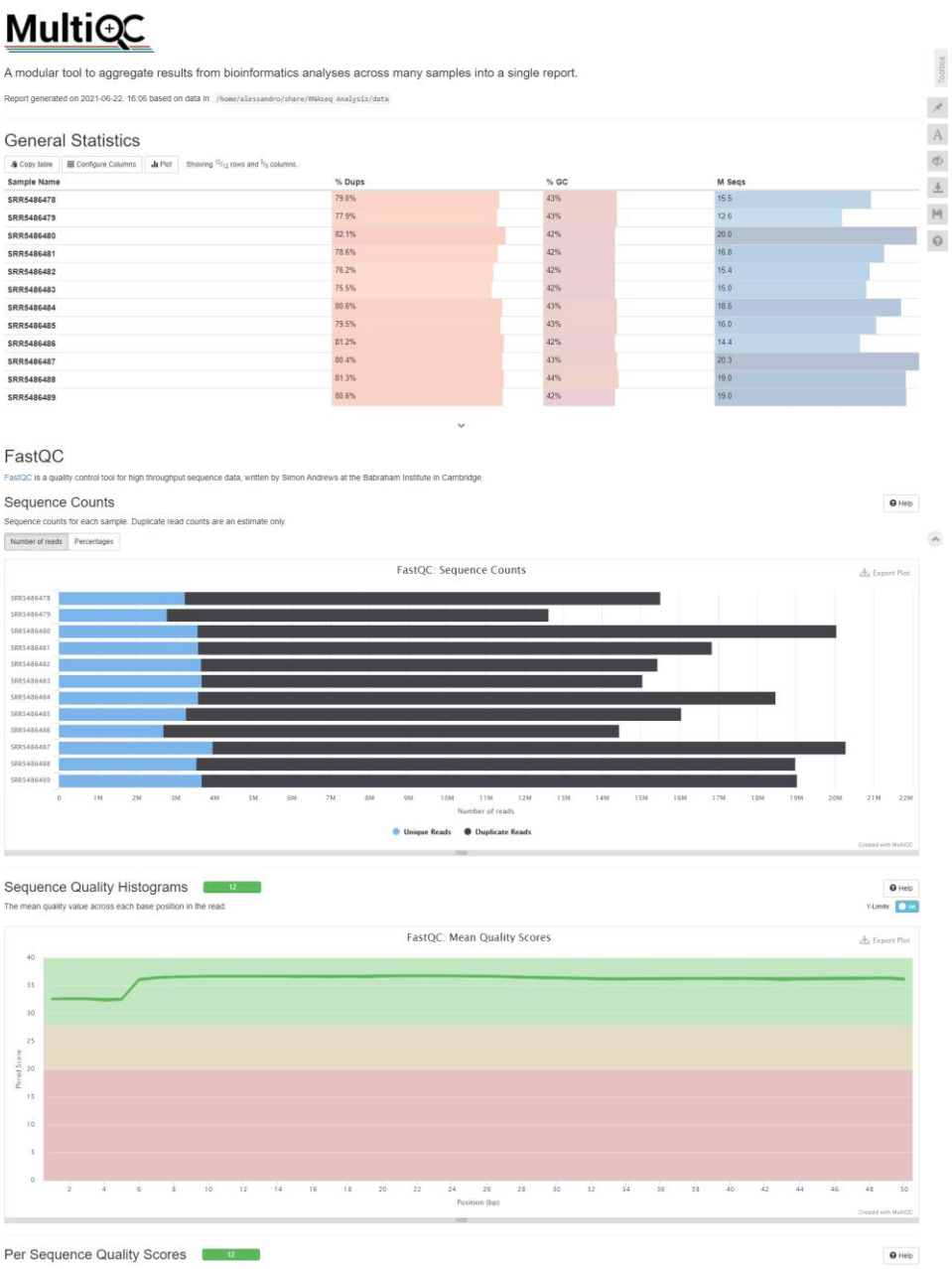
MultiQC is a useful tool capable of collecting all the FastQC output files of *multiple analyzed samples*, and give a summarized result in the form of a **MultiQC Report**

In order to run MultiQC, we simply run this command:

```
multiqc .
```

MultiQC Report

And this is what the **MultiQC report** will look like:



MultiQC Report

TrimGalore?

Taking a look at the FastQC reports or at the MultiQC report we can have an overview of the quality of our files and possibly decide to perform a **quality and adapter trimming step with TrimGalore**.

We usually use *TrimGalore* to trim our sequences in the case of *low-quality ends* (when the **phred score** of the bases is less than 25), in the case of *overlap with adapters* (with a stringency of 5) or directly discard sequences which (after trimming) have a length shorter than 35.
However, our reports clearly show that all our reads always have a quality which is greater than 30, they basically don't contain any adapter sequences and they always have a length of 50 BP.
For this reason, the *TrimGalore* step doesn't appear to be necessary.

4 - Salmon

The next move in our analysis is **sample quantification** using **Salmon**. This software will give us *expression levels* of our samples, which we will then use in the *differential expression analysis*.

Salmon Index

First, we need to create an `index` for the *S. Cerevisiae transcriptome*, in order for Salmon to run faster. We can do that by downloading the `Saccharomyces_cerevisiae.R64-1-1.cdna.all.fa.gz` file from Ensembl ftp and then running this command in the shell:

```
# create salmon index
salmon index -t Saccharomyces_cerevisiae.R64-1-1.cdna.all.fa.gz -i yeast_index
```

Quantification with Salmon

Now that we have our `transcriptome index`, we can run **Salmon** and store the output results in a single directory.
In order to do that, we run the following script:

```
echo "Run salmon..."

mkdir -p salmon_results

for filename in *.fastq.gz
do
   base=$(basename $filename .fastq.gz)

   echo "Align sample ${base}..."

   salmon quant -i yeast_index \
        --libType A \
            -r ${base}.fastq.gz \
            -o salmon_results/${base}
done
```

5 - DESeq2 Analysis

Now that we have our **quantification files** we can switch our analysis to `RStudio` and perform a **Differential Expression analysis** using DESeq2.

Packages Load

First of all, let's download all the packages we will need

```
library(DESeq2)
library(tximport)
library(GenomicFeatures)
library(readr)
library(ggplot2)
```

colData

Now we can start creating the objects which will be necessary for our DESeq Analysis.
The **sampledata data.frame** can be created by using the Run IDs of our `SRR_Acc_List.txt` file as `rownames`. Then, we will ad a `group` column according to the sample characteristics.

```
# create a sapmledata data.frame ----
sampledata <-
  read.csv("SRR_Acc_List.txt", header = FALSE)
colnames(sampledata) <- "runids"

sampledata$group <- rep(c("WT_no_stress", "WT_stress", "Mut_no_stress",
"Mut_stress"), each = 3)

rownames(sampledata) <- sampledata$runids
sampledata[1] <- NULL
sampledata
```

```
##                    group
## SRR5486478  WT_no_stress
## SRR5486479  WT_no_stress
## SRR5486480  WT_no_stress
## SRR5486481     WT_stress
## SRR5486482     WT_stress
## SRR5486483     WT_stress
## SRR5486484 Mut_no_stress
## SRR5486485 Mut_no_stress
## SRR5486486 Mut_no_stress
## SRR5486487    Mut_stress
## SRR5486488    Mut_stress
## SRR5486489    Mut_stress
```

Import data with `tximport`

We will then import our *transcript-level abundances* from the `quant.sf` files using the `tximport` function in a few steps:

- Set the `quant.sf` file paths

```
files <- file.path("data/salmon_results/", row.names(sampledata), "quant.sf")
names(files) <- row.names(colData)
```

- Create a `tx2gene` object after having downloaded the right `GTF` file from Ensembl website

```
txdb <- GenomicFeatures::makeTxDbFromGFF("Saccharomyces_cerevisiae.R64-1-1.104.gtf")
k <- keys(txdb, keytype = "GENEID")
tx2gene <- select(txdb, keys = k, keytype = "GENEID", columns = "TXNAME")
```

- Reorder the columns of `tx2gene`

```
tx2gene <- tx2gene[, c("TXNAME", "GENEID")]
head(tx2gene)
```

```
##         TXNAME GENEID
## 1 ETS1-1_rRNA ETS1-1
## 2 ETS1-2_rRNA ETS1-2
## 3 ETS2-1_rRNA ETS2-1
## 4 ETS2-2_rRNA ETS2-2
## 5  HRA1_ncRNA   HRA1
## 6  ICR1_ncRNA   ICR1
```

- Import **Salmon quantification data** with `tximport` creating a `txi.salmon` object

```
txi.salmon <- tximport(files = files, type = "salmon", tx2gene = tx2gene, ignoreTxVersion = TRUE, dropInfReps = TRUE)
head(txi.salmon$counts)
```

```
##           [,1] [,2] [,3] [,4]    [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
## Q0010 0.000000    0    0    0 0.00000    0    0    0    0     0     0     0
## Q0017 0.000000    0    0    0 0.00000    0    0    0    0     0     0     0
## Q0032 0.000000    0    0    0 0.00000    0    0    0    0     0     0     0
## Q0045 0.999999    2    1    0 1.51721    4    0    2    0     1     2     0
## Q0050 4.000000    1    3    1 3.00000    1    2    4    0     0     1     0
## Q0055 2.000000    0    3    1 4.00000    3    2    0    1     4     1     5
```

- Check that the sample names match between the `rownames` of `sampledata` and the `colnames` of `txi.salmon$counts`

```
identical(x = rownames(colData), y = colnames(txi.salmon$counts))
```

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

DESeq2 Pipeline

Now that we have all the required objects, let's get to the **DESeq2 pipeline**:

- We store the input values in the `dds` object by using the `DDSeqDataSetFromTximport` function

```
dds <- DESeqDataSetFromTximport(txi = txi.salmon, colData = sampledata, design = ~group)
```

- We filter non-expressed genes, re-order the levels of the factor with the `relevel` function, and perform the analysis running the `DESeq` function

```
keep <- rowSums(counts(dds)) > 1
dds <- dds[keep, ]

dds$group <- relevel(x = dds$group, ref = "WT_no_stress")
dds <- DESeq(dds)
```

- We finally store our results in a `res` object and take a look at them

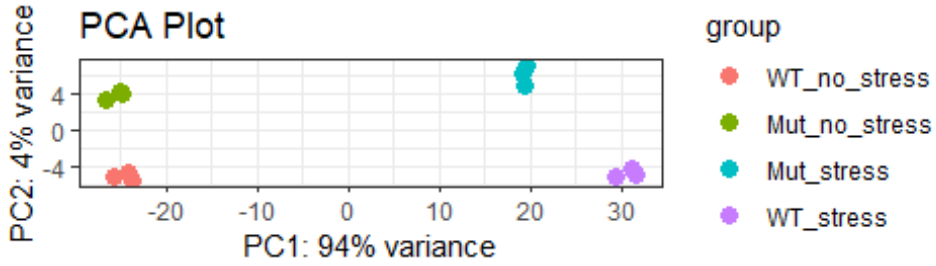
```
res <- results(dds)
summary(res)

##
## out of 6488 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)       : 2276, 35%
## LFC < 0 (down)     : 2227, 34%
## outliers [1]       : 7, 0.11%
## low counts [2]     : 0, 0%
## (mean count < 0)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results

res

## log2 fold change (MLE): group WT stress vs WT no stress
## Wald test p-value: group WT stress vs WT no stress
## DataFrame with 6488 rows and 6 columns
##                  baseMean     log2FoldChange             lfcSE
##                 <numeric>          <numeric>         <numeric>
## Q0045    1.29237738024747  0.151212755349245  1.66092855080181
## Q0050    1.73862295449891 -0.947218027604868  1.36288705586312
## Q0055    1.95356510083966  0.450535862906704  1.27689190222002
## Q0075   0.174549007220374  -1.30864414378148  4.40735632396697
## Q0085   0.559166574240002  -2.01285779358549  2.72456939886744
## ...                   ...                ...               ...
## YPR200C  45.6621706338068  0.694818599132749 0.325927242024541
## YPR201W  98.7122516397809  0.505964473767392 0.202987270320784
## YPR202W   187.49327548858  0.618345996613313 0.373709086461011
## YPR203W  164.743598079909 -0.550594452121245 0.232226515244343
## YPR204W  6072.64978079766 -0.133784137830699 0.116401140780413
##                      stat             pvalue               padj
##                 <numeric>          <numeric>          <numeric>
## Q0045   0.0910410958233261  0.927459936880834  0.946448814838789
## Q0050   -0.695008455418186  0.487050002652271  0.558981949210088
## Q0055    0.352837904385952  0.724209965428483  0.778117504300729
## Q0075   -0.296922700954569  0.766525524465633   0.81453548517163
## Q0085   -0.738780151616692  0.460040506175801  0.533787311458379
## ...                    ...                ...                ...
## YPR200C    2.1318211844361  0.033021549419978 0.0505104228914037
## YPR201W   2.49259213628425 0.0126814424065324 0.0206244487419666
```

```
## YPR202W   1.65461857636107 0.0980018929278219  0.137033499043196
## YPR203W  -2.37093706350424 0.0177430520901809  0.028351262474473
## YPR204W  -1.14933699905123  0.250417047265818  0.315933985464234
```

PCA Plot

Now let's plot the **PCA** of the `rld` normalized values

```
rld <- rlog(dds)
DESeq2::plotPCA(rld, ntop = 500, intgroup = 'group') +
  theme_bw() + labs(title="PCA Plot")
```



Group Comparisons

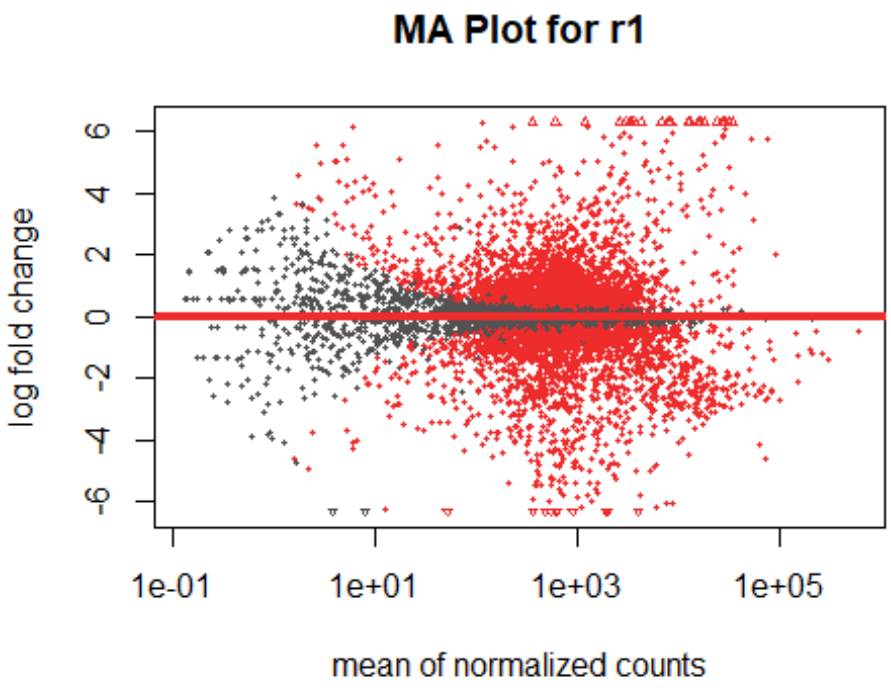
Now we are going to perform **comparisons** between the different groups of samples. These are going to be:

- `r1: WT_stress vs WT_no_stress`
- `r2: Mut_stress vs Mut_no_stress`
- `r3: Mut_stress vs WT_stress`
- `r4: Mut_no_stress vs WT_no_stress`

We are also going to produce an **MA Plot** for each of the four comparisons.

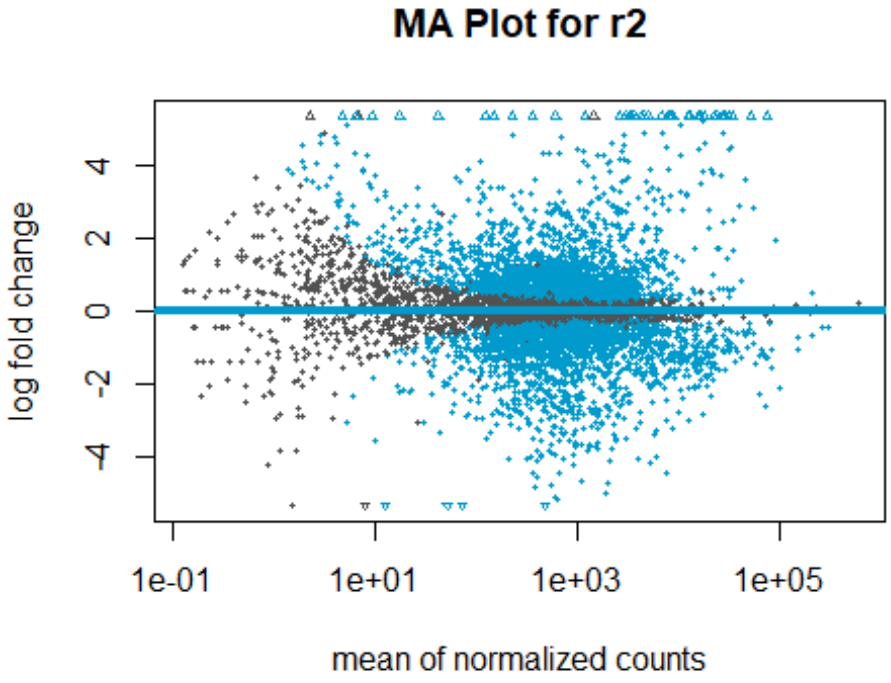
`r1: WT_stress vs WT_no_stress`

```
r1 <- results(dds, contrast = c("group", "WT_stress", "WT_no_stress"))
DESeq2::plotMA(r1, alpha = 0.05, main = "MA Plot for r1", colSig="firebrick2",
colLine="firebrick2")
```



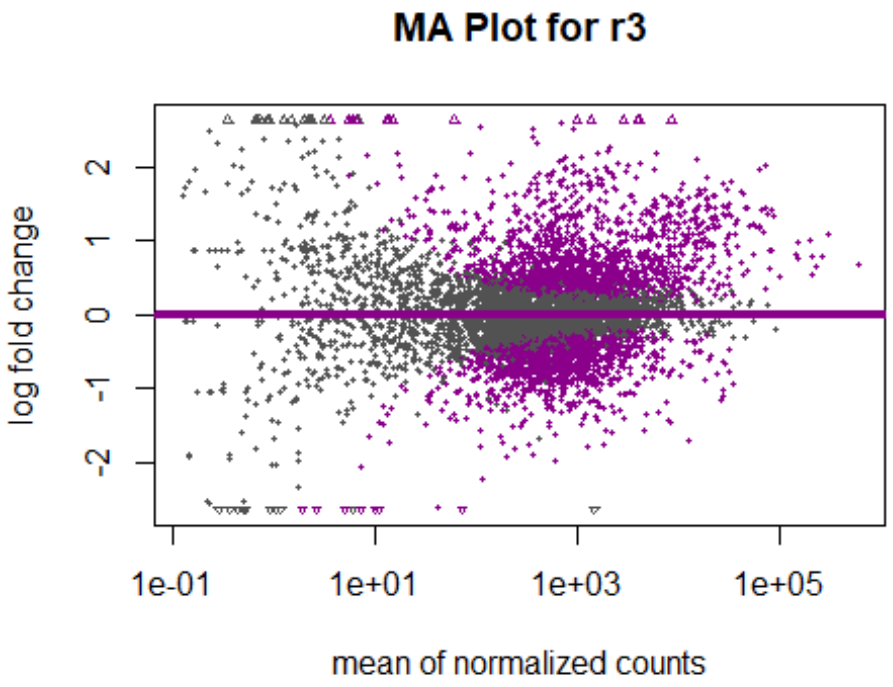
r2: Mut_stress vs Mut_no_stress

```
r2 <- results(dds, contrast = c("group", "Mut_stress","Mut_no_stress"))
DESeq2::plotMA(r2, alpha = 0.05, main = "MA Plot for r2", colSig="deepskyblue3",
colLine="deepskyblue3")
```



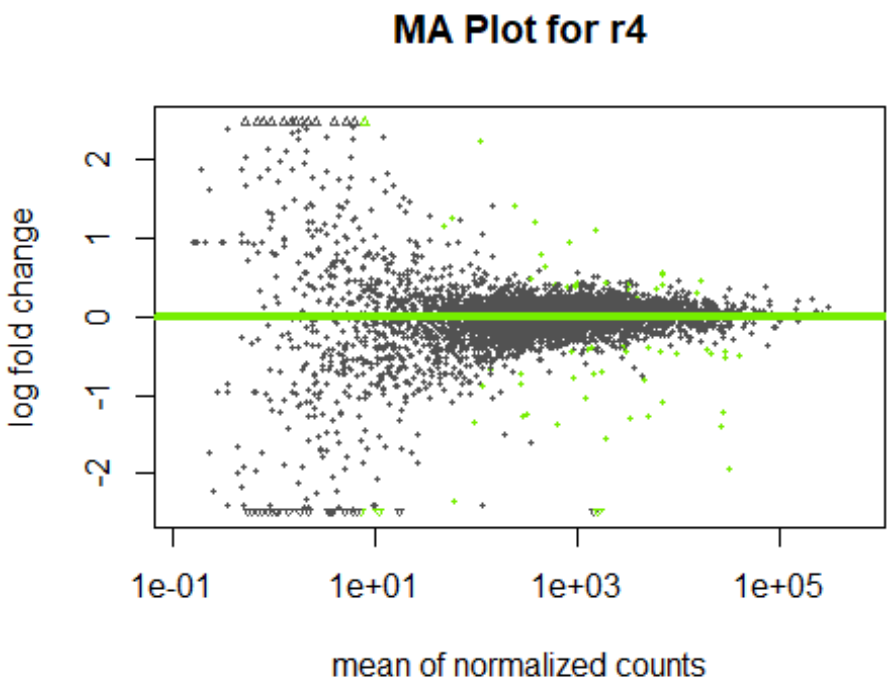
r3: Mut_stress vs WT_stress

```
r3 <- results(dds, contrast = c("group", "Mut_stress", "WT_stress"))
DESeq2::plotMA(r3, alpha = 0.05, main = "MA Plot for r3", colSig="darkmagenta",
colLine="darkmagenta")
```



r4: Mut_no_stress vs WT_no_stress

```
r4 <- results(dds, contrast = c("group", "Mut_no_stress","WT_no_stress"))
DESeq2::plotMA(r4, alpha = 0.05, main = "MA Plot for r4", colSig="chartreuse2",
colLine="chartreuse2")
```



6 - GO Enrichment Analysis

Now we will perform a **Gene Ontology (GO) Enrichment Analysis** on each of the four comparisons we generated in *Step 6*.

Packages Load

In order to perform the analysis, it's going to be necessary to load a few more packages:

```
library("AnnotationDbi")
library("org.Sc.sgd.db")
library("clusterProfiler")
```

Analysis on r1

Let's start with the analysis on the first comparison `r1`.
First of all we need to **remove genes with NA values**, **select genes which have a p-adjusted value smaller than 0.05**, and **select UP regulated genes (log2FoldChange > 1)**

```
upregulated_r1 <- r1[!is.na(r1$padj),]
upregulated_r1 <- upregulated_r1[upregulated_r1$padj < 0.05,]
upregulated_r1 <- upregulated_r1[upregulated_r1$log2FoldChange > 1,]
```

Then we find the Gene Names using this code:

```
upregulated_r1$symbol <- mapIds(org.Sc.sgd.db,
                     keys=rownames(upregulated_r1),
                     column="GENENAME",
```

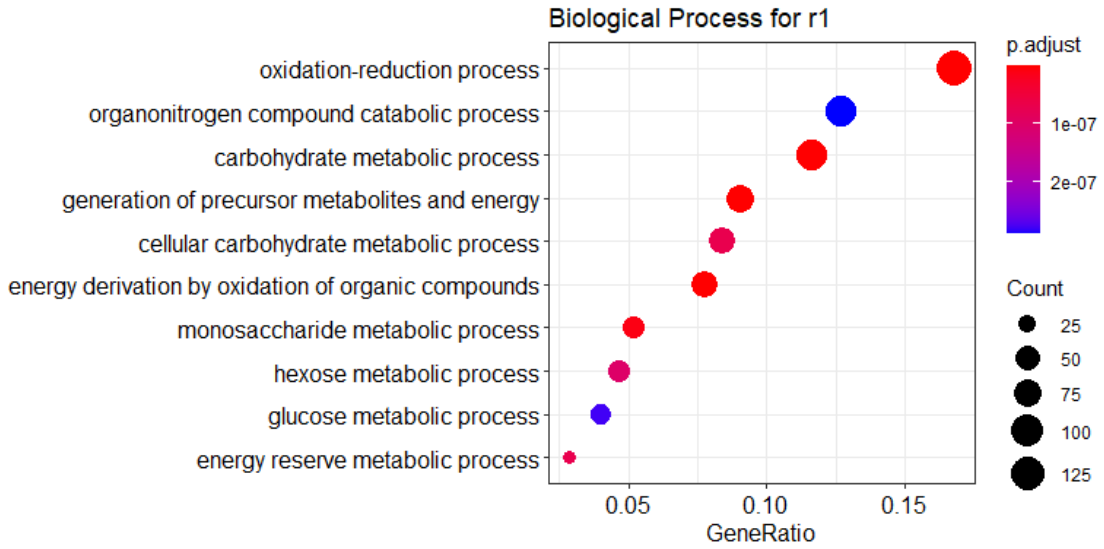
```
                    keytype="ENSEMBL",
                    multiVals="first")
```

Now we use the `enrichGO` function in order to perform the analysis on our `selected_r1` object and choosing the `Biological Process (BP)` ontology

```
GO_BP_1 <- enrichGO(upregulated_r1$symbol, OrgDb = "org.Sc.sgd.db", keyType =
"GENENAME", ont = "BP")
```

We finally plot the results of our *GO Enrichment analysis* using the simple `dotplot` function

```
dotplot(GO_BP_1, title = "Biological Process for r1")
```



We are now going to repeat the same steps over *r2*, *r3* and *r4*.

Analysis on r2

Select **UP regulated** genes (`log2FoldChange > 1` and `padj < 0.05`)

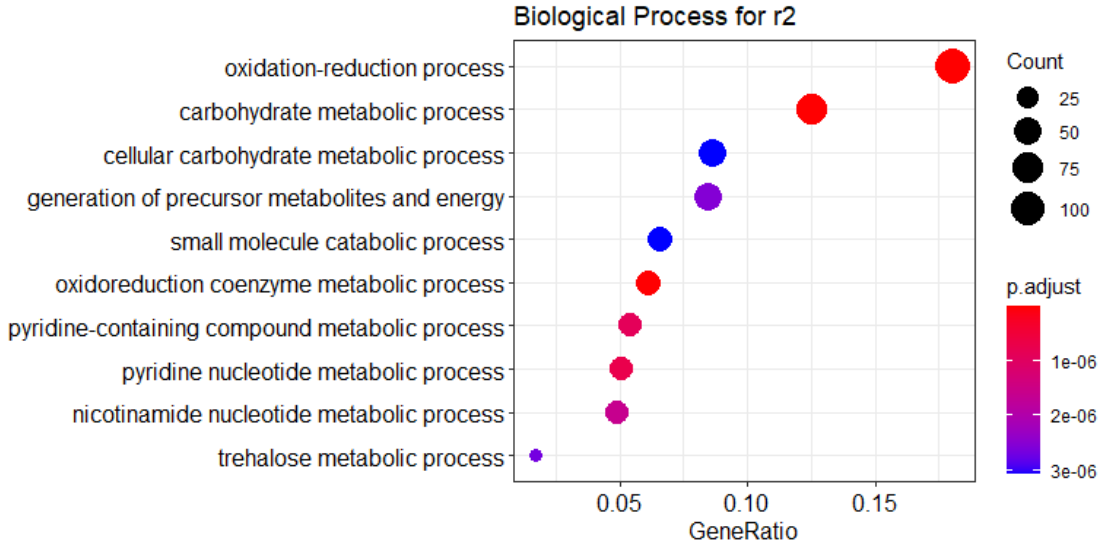
```
upregulated_r2 <- r2[!is.na(r2$padj),]
upregulated_r2 <- upregulated_r2[upregulated_r2$padj < 0.05,]
upregulated_r2 <- upregulated_r2[upregulated_r2$log2FoldChange > 1,]
```

Find the Gene Names

```
upregulated_r2$symbol <- mapIds(org.Sc.sgd.db,
                    keys=rownames(upregulated_r2),
                    column="GENENAME",
                    keytype="ENSEMBL",
                    multiVals="first")
```

Use the `enrichGO` function with `Biological Process (BP)` ontology and plot the results with `dotplot`

```
GO_BP_2 <- enrichGO(upregulated_r2$symbol, OrgDb = "org.Sc.sgd.db", keyType =
"GENENAME", ont = "BP")
dotplot(GO_BP_2, title = "Biological Process for r2")
```



Analysis on r3

Select **UP regulated** genes (`log2FoldChange > 1` and `padj < 0.05`)

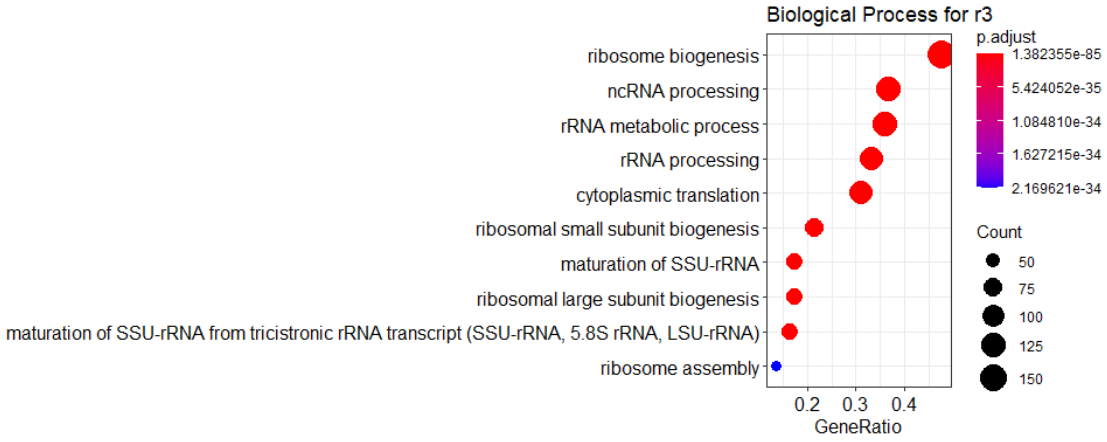
```
upregulated_r3 <- r3[!is.na(r3$padj),]
upregulated_r3 <- upregulated_r3[upregulated_r3$padj < 0.05,]
upregulated_r3 <- upregulated_r3[upregulated_r3$log2FoldChange > 1,]
```

Find the Gene Names

```
upregulated_r3$symbol <- mapIds(org.Sc.sgd.db,
                     keys=rownames(upregulated_r3),
                     column="GENENAME",
                     keytype="ENSEMBL",
                     multiVals="first")
```

Use the `enrichGO` function with `Biological Process (BP)` ontology and plot the results with `dotplot`

```
GO_BP_3 <- enrichGO(upregulated_r3$symbol, OrgDb = "org.Sc.sgd.db", keyType =
"GENENAME", ont = "BP")
dotplot(GO_BP_3, title = "Biological Process for r3")
```



Analysis on r4

Select **UP regulated** genes (`log2FoldChange > 1` and `padj < 0.05`)

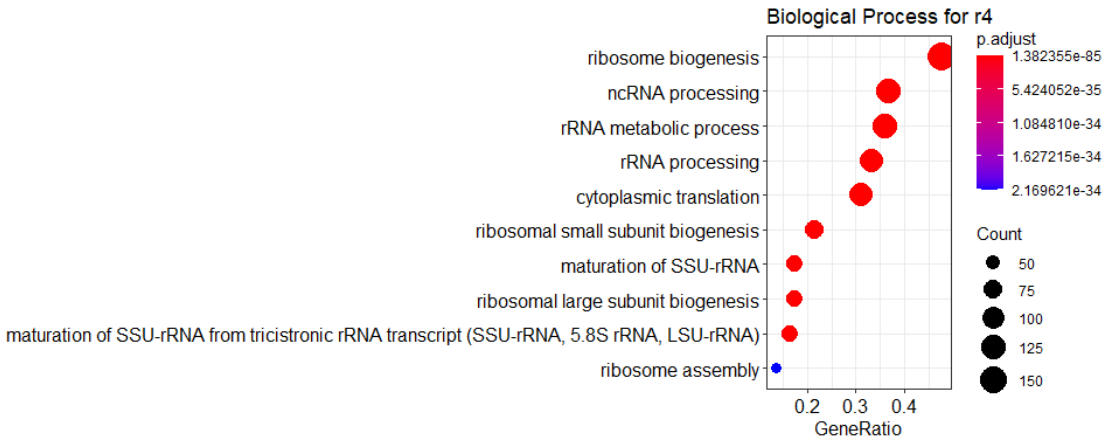
```
upregulated_r4 <- r4[!is.na(r4$padj),]
upregulated_r4 <- upregulated_r4[upregulated_r4$padj < 0.05,]
upregulated_r4 <- upregulated_r4[upregulated_r4$log2FoldChange > 1,]
```

Find the Gene Names

```
upregulated_r4$symbol <- mapIds(org.Sc.sgd.db,
                     keys=rownames(upregulated_r4),
                     column="GENENAME",
                     keytype="ENSEMBL",
                     multiVals="first")
```

Use the `enrichGO` function with `Biological Process (BP)` ontology and plot the results with `dotplot`

```
GO_BP_4 <- enrichGO(upregulated_r4$symbol, OrgDb = "org.Sc.sgd.db", keyType =
"GENENAME", ont = "BP")
dotplot(GO_BP_3, title = "Biological Process for r4")
```



7 - DESeq2 on Authors' Data

Our analysis on the raw data of the study is finished. Now it's time to repeat the same **Differential Expression Analysis with DESeq2** on data which had been previously processed by the *paper authors*. Remember this processed data have been already downloaded in the `GSE98352_DESeq2_raw_counts.tsv`file in `Step 1`.

Import the Data

In order to import the data from the `GSE98352_DESeq2_raw_counts.tsv` file, we use the `read.table` function

```
countData <- read.table('GSE98352_DESeq2_raw_counts.tsv')
head(countData)
```

```
##           X46_17739_CCGTCC X47_17740_CGTACG X48_17741_TGACCA X49_17742_GTCCGC
## YAL012W              16225            30561            22107             8642
## YAL069W                  1                0                0                1
## YAL068W-A                0                0                0                0
## YAL068C                  5                6               13                7
## YAL067W-A                0                0                0                0
## YAL067C                 18               38               44               28
##           X50_17743_TTAGGC X51_17744_ACAGTG X52_17745_CGTACG X53_17746_CGATGT
## YAL012W               7921             8333            24279            20016
## YAL069W                  0                0                1                0
## YAL068W-A                0                0                0                0
## YAL068C                  6                6               20               14
## YAL067W-A                0                0                0                0
## YAL067C                 23               18               18               14
##           X54_17747_GCCAAT X55_17899_GAGTGG X56_17900_GATCAG X57_17750_CAGATC
## YAL012W              17950             9891            10373            11143
## YAL069W                  0                1                1                0
## YAL068W-A                0                0                0                0
## YAL068C                 20               17                8                5
## YAL067W-A                0                0                0                0
## YAL067C                  7               12               10               22
```

It's now necessary to change the `colnames` of our new `countData` object, in order for them to match to the `rownames` of `sampledata

```
colnames(countData) <- rownames(sampledata)
head(countData)
```

```
##           SRR5486478 SRR5486479 SRR5486480 SRR5486481 SRR5486482 SRR5486483
## YAL012W        16225      30561      22107       8642       7921       8333
## YAL069W            1          0          0          1          0          0
## YAL068W-A          0          0          0          0          0          0
## YAL068C            5          6         13          7          6          6
## YAL067W-A          0          0          0          0          0          0
## YAL067C           18         38         44         28         23         18
##           SRR5486484 SRR5486485 SRR5486486 SRR5486487 SRR5486488 SRR5486489
## YAL012W        24279      20016      17950       9891      10373      11143
```

```
## YAL069W            1          0          0          1          1          0
## YAL068W-A          0          0          0          0          0          0
## YAL068C           20         14         20         17          8          5
## YAL067W-A          0          0          0          0          0          0
## YAL067C           18         14          7         12         10         22
```

DESeq2 Pipeline

Now we can run the same **DESeq2 Pipeline** we used earlier in `Step 5` and take a look at the results

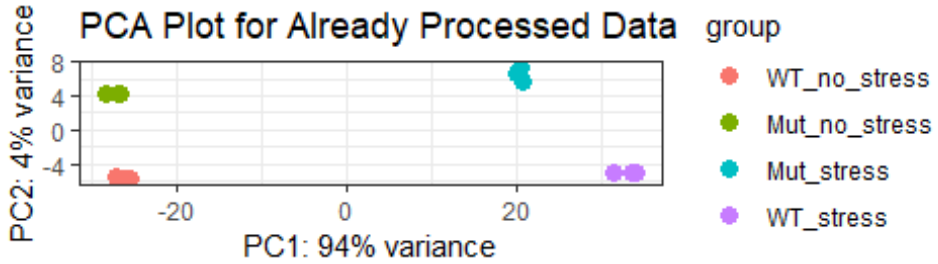
```
# DESeq2 pipeline ----
new_dds <- DESeqDataSetFromMatrix(countData = countData, colData = sampledata,
design = ~group)

# filtering not expressed genes ----
new_keep <- rowSums(counts(new_dds)) > 1
new_dds <- new_dds[new_keep, ]

new_dds$group <- relevel(x = new_dds$group, ref = "WT_no_stress")
new_dds <- DESeq(new_dds)

# save the results of DE analysis ----
new_res <- results(new_dds)

# look at the results ----
summary(new_res)

##
## out of 6874 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)       : 2324, 34%
## LFC < 0 (down)     : 2264, 33%
## outliers [1]       : 4, 0.058%
## low counts [2]     : 400, 5.8%
## (mean count < 1)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results

new_res

## log2 fold change (MLE): group WT stress vs WT no stress
## Wald test p-value: group WT stress vs WT no stress
## DataFrame with 6874 rows and 6 columns
##                  baseMean       log2FoldChange             lfcSE
##                 <numeric>            <numeric>         <numeric>
## YAL012W   15810.268419489    -1.43715481994941 0.127782300699693
## YAL069W 0.417075787253881 -0.00608911932500127  2.76910609938262
## YAL068C  11.0503487050225    -0.33766725854825 0.625757575688139
## YAL067C  20.2139114815706   -0.507059602813827 0.400014665521222
## YAL066W 0.296691638131935 -0.00609032351124565  3.88275955984994
## ...                   ...                  ...               ...
```

```
## YMR321C  382.797395670517    -2.00249946948773  0.16667095492522
## YMR322C  22.7059218869664     4.29712121332978  0.60915187777631
## YMR323W   31.305156660636     2.87479553579823  0.41534826092609
## YMR325W  6.17462793852108     1.40265525122828  0.91620164728667
## YMR326C 0.580243237726815  -0.0060925526384371  2.64260016840328
##                         stat               pvalue                 padj
##                    <numeric>            <numeric>            <numeric>
## YAL012W    -11.2469004868439 2.39873944673103e-29 1.29873173392049e-28
## YAL069W -0.00219894764103075    0.998245495041155                   NA
## YAL068C   -0.539613536722941    0.589463579226292    0.651379907360224
## YAL067C    -1.26760253190499    0.204939927311455     0.26045203883424
## YAL066W  -0.0015685554094627    0.998748474369228                   NA
## ...                       ...                  ...                  ...
## YMR321C     -12.014687684403 2.97491466344636e-33 1.78384595667266e-32
## YMR322C      7.0542690092597 1.73510420751321e-12 5.74226302946827e-12
## YMR323W     6.92140982940046 4.47170736390809e-12 1.44732099272063e-11
## YMR325W      1.5309460044983     0.12578273803018    0.168247739312645
## YMR326C  -0.0023055143609252     0.99816046731635                   NA
```

PCA Plot

We then generate a `PCA Plot`

```
new_rld <- rlog(new_dds)
DESeq2::plotPCA(new_rld, ntop = 500, intgroup = 'group') +
  theme_bw() + labs(title="PCA Plot for Already Processed Data")
```



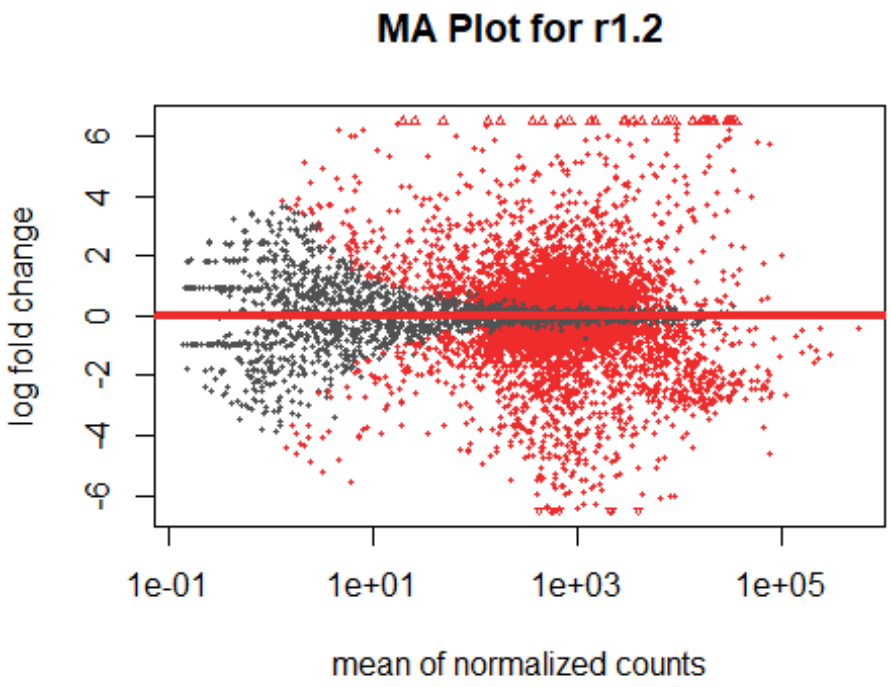
Group Comparisons

The last step of this analysis, as we did in step 5, is gonna be to analyse the comparisons between the 4 different groups of samples:

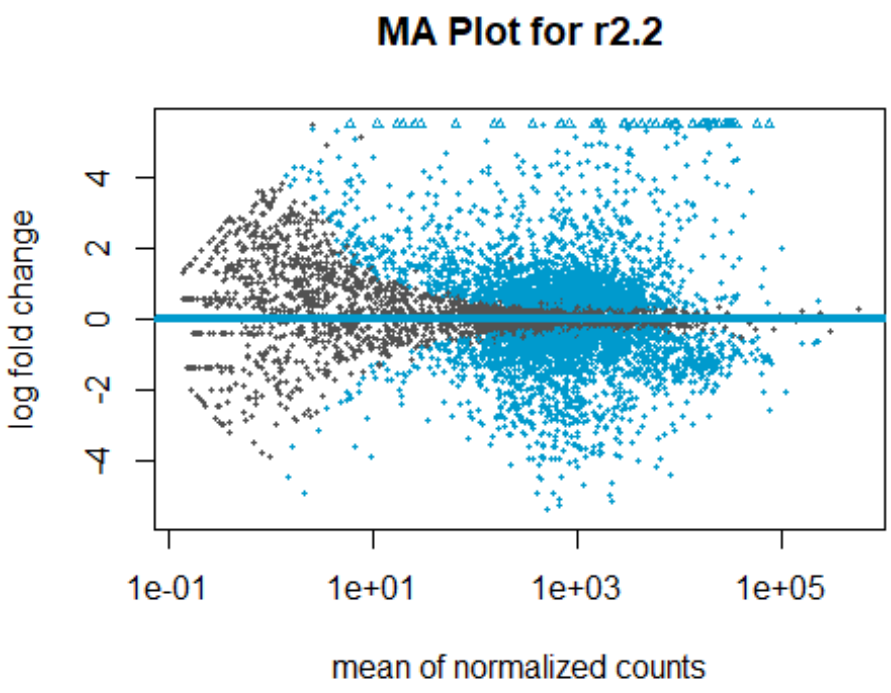
- `r1.2: WT_stress vs WT_no_stress`
- `r2.2: Mut_stress vs Mut_no_stress`
- `r3.2: Mut_stress vs WT_stress`
- `r4.2: Mut_no_stress vs WT_no_stress`

We are also going to produce an **MA Plot** for each of the four comparisons.

```
r1.2: WT_stress vs WT_no_stress
r1.2 <- results(new_dds, contrast = c("group", "WT_stress", "WT_no_stress"))
DESeq2::plotMA(r1.2, alpha = 0.05, main = "MA Plot for r1.2", colSig="firebrick2",
colLine="firebrick2")
```

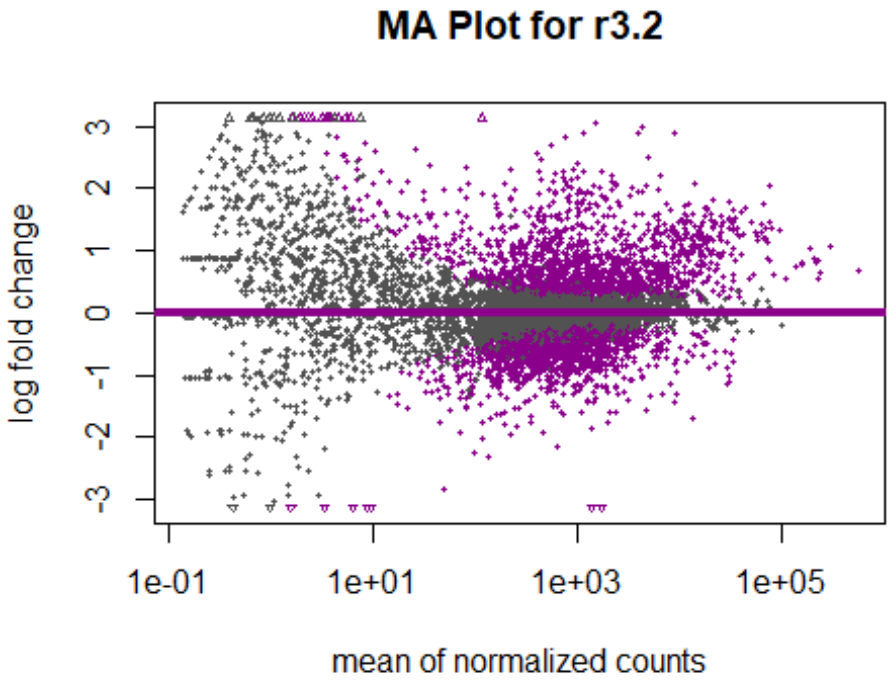



```
r2.2: Mut_stress vs Mut_no_stress
r2.2 <- results(new_dds, contrast = c("group", "Mut_stress","Mut_no_stress"))
DESeq2::plotMA(r2.2, alpha = 0.05, main = "MA Plot for r2.2",
colSig="deepskyblue3", colLine="deepskyblue3")
```



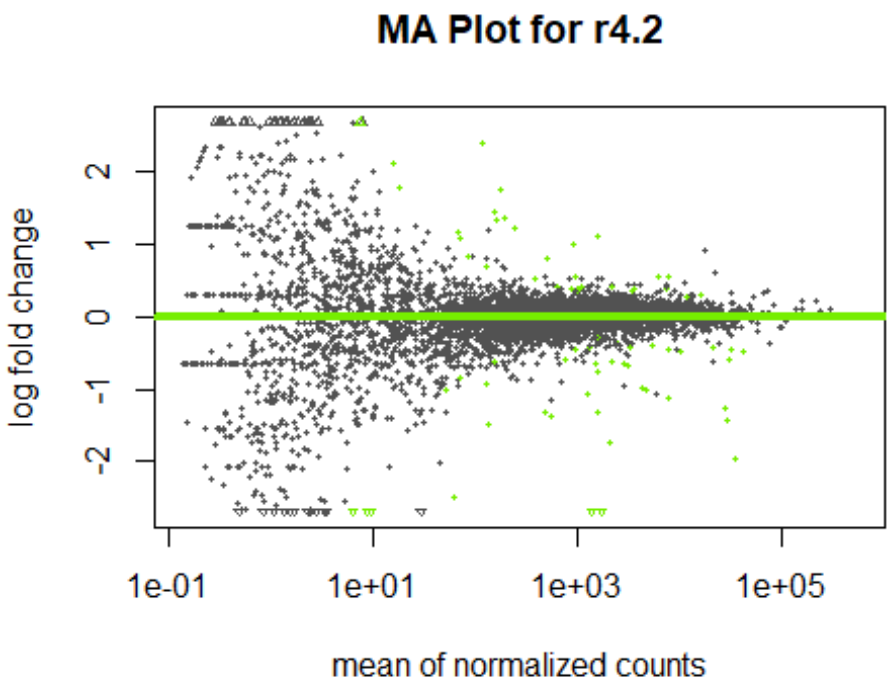
r3.2: Mut_stress vs WT_stress

```
r3.2 <- results(new_dds, contrast = c("group", "Mut_stress", "WT_stress"))
DESeq2::plotMA(r3.2, alpha = 0.05, main = "MA Plot for r3.2",
colSig="darkmagenta", colLine="darkmagenta")
```



r4.2: Mut_no_stress vs WT_no_stress

```
r4.2 <- results(new_dds, contrast = c("group", "Mut_no_stress","WT_no_stress"))
DESeq2::plotMA(r4.2, alpha = 0.05, main = "MA Plot for r4.2",
colSig="chartreuse2", colLine="chartreuse2")
```



8 - Final Step

We finally got to the conclusions of our analysis

Save the DE Genes

We can save **the differentially expressed genes** with `padj < 0.05` of the 4 comparisons created in `Step 5` in a `results*` file using the `write.table`.

Results for r1

First we need to select the **differentially expressed genes** from `r1`

```
DEr1 <- r1[!is.na(r1$padj),]
DEr1 <- DEr1[DEr1$padj < 0.05,]
DEr1 <- DEr1[abs(DEr1$log2FoldChange) > 1,]
head(DEr1)
```

```
## log2 fold change (MLE): group WT_stress vs WT_no_stress
## Wald test p-value: group WT stress vs WT no stress
## DataFrame with 6 rows and 6 columns
##                 baseMean    log2FoldChange             lfcSE                stat
##                <numeric>         <numeric>         <numeric>           <numeric>
```

```
## Q0140   10.4621919390357 -1.30072876143615 0.594264354600454 -2.18880495080456
## YAL002W 772.609213852169  1.27688421289951 0.128654507698777  9.92490846794981
## YAL003W 14743.9545723415 -2.26027151938458 0.171368894009491 -13.1895087054679
## YAL005C 54825.7977004424  3.14107157342205 0.117956713383331  26.6290190980001
## YAL008W 645.553891565133  1.65574337728817 0.146456390372288  11.3053679192783
## YAL012W 14229.4629717613 -1.43483358051018 0.125878434344606 -11.3985655126768
##                        pvalue                  padj
##                     <numeric>             <numeric>
## Q0140      0.0286110197479836     0.044307770367188
## YAL002W  3.24400230980876e-23  1.57132877203816e-22
## YAL003W  1.00840272956441e-39  7.38469840712646e-39
## YAL005C 3.13223634759583e-156 1.03045805932835e-154
## YAL008W  1.23435702585153e-29    7.136367425998e-29
## YAL012W  4.25065121059121e-30  2.48408210061692e-29
```

Now we save the `DEr1` object in a `results_r1` file

```
write.table(x = DEr1, file = "results_r1.txt", sep = "\t", col.names = NA)
```

We are now going to repeat the same process for the other 3 results.

Results for r2

Select the **differentially expressed genes** from `r2`

```
DEr2 <- r2[!is.na(r2$padj),]
DEr2 <- DEr2[DEr2$padj < 0.05,]
DEr2 <- DEr2[abs(DEr2$log2FoldChange) > 1,]
head(DEr2)
```

```
## log2 fold change (MLE): group Mut_stress vs Mut_no_stress
## Wald test p-value: group Mut_stress vs Mut_no_stress
## DataFrame with 6 rows and 6 columns
##                   baseMean    log2FoldChange             lfcSE
##                  <numeric>         <numeric>         <numeric>
## YAL002W   772.609213852169  1.14357449364594 0.130410410618146
## YAL003W   14743.9545723415 -1.34330007158872 0.171176865491924
## YAL005C   54825.7977004424  2.80260718243796 0.117900852449587
## YAL008W   645.553891565133    1.145510030421 0.147421813567139
## YAL012W   14229.4629717613 -1.37976901942383 0.125784056073298
## YAL016C-A 31.7905798927562  1.66905011907848 0.442890694744961
##                       stat                pvalue                  padj
##                  <numeric>             <numeric>             <numeric>
## YAL002W    8.76904296386605 1.80191280059142e-18   1.1186012318614e-17
## YAL003W    -7.8474393588665 4.24617613285067e-15  2.22829696493969e-14
## YAL005C    23.7708814161145 6.6840245454689e-125 2.98752848821958e-123
## YAL008W    7.77028855298482 7.83075309518132e-15  4.06008886478961e-14
## YAL012W   -10.9693474872507 5.36582199519818e-28  4.71857426741919e-27
## YAL016C-A  3.76853733637282 0.000164206926608478  0.000371976613544056
```

Save the `DEr2` object in a `results_r2` file

```
write.table(x = DEr2, file = "results_r2.txt", sep = "\t", col.names = NA)
```

Results for r3

Select the **differentially expressed genes** from `r3`

```
DEr3 <- r3[!is.na(r3$padj),]
DEr3 <- DEr3[DEr3$padj < 0.05,]
DEr3 <- DEr3[abs(DEr3$log2FoldChange) > 1,]
head(DEr3)
```

```
## log2 fold change (MLE): group Mut_stress vs WT_stress
## Wald test p-value: group Mut_stress vs WT_stress
## DataFrame with 6 rows and 6 columns
##                   baseMean    log2FoldChange             lfcSE
##                  <numeric>         <numeric>         <numeric>
## YAL016C-A 31.7905798927562  1.00917663403212  0.41848887603331
## YAL016C-B 36.4965112530306 -1.03091786200639 0.340802677604301
## YAL017W   1142.58550379625 -1.40318304198888  0.23430649659758
## YAL025C   417.789436073827  1.48210989735898 0.242628900414724
## YAL034C   812.399157652244 -1.19898442342628 0.106731099613816
## YAL036C    1755.7952901525  1.25880303148539 0.102208121822718
##                       stat               pvalue                padj
##                  <numeric>            <numeric>           <numeric>
## YAL016C-A  2.41147779983474   0.0158880190700635   0.0376326355535793
## YAL016C-B  -3.0249699598997  0.00248657823418747  0.00767469872669323
## YAL017W   -5.98866468648898 2.11570839871226e-09 2.25971880232209e-08
## YAL025C    6.10854640492381 1.00542627864815e-09 1.14094508385398e-08
## YAL034C   -11.2336931575197 2.78585482454708e-29 2.32948781710482e-27
## YAL036C    12.3160763453691 7.42183181220756e-35 1.15038393089217e-32
```

Save the `DEr3` object in a `results_r3` file

```
write.table(x = DEr3, file = "results_r3.txt", sep = "\t", col.names = NA)
```

Results for r4

Select the **differentially expressed genes** from `r4`

```
DEr4 <- r4[!is.na(r4$padj),]
DEr4 <- DEr4[DEr4$padj < 0.05,]
DEr4 <- DEr4[abs(DEr4$log2FoldChange) > 1,]
head(DEr4)
```

```
## log2 fold change (MLE): group Mut_no_stress vs WT_no_stress
## Wald test p-value: group Mut no stress vs WT no stress
## DataFrame with 6 rows and 6 columns
##                   baseMean    log2FoldChange              lfcSE
##                  <numeric>         <numeric>          <numeric>
## YAR071W   319.043913683148 -1.23301075348617  0.203975770715957
## YBR054W   1885.16965912913 -1.54825140307086  0.145918825276407
## YEL070W   47.7422222713786   1.1513602631911  0.306376523093121
## YER081W   1494.49716731954   1.0963705676535 0.0926076368272862
## YFL014W   31181.9344350388 -1.93106433574399  0.267498165956147
## YFR052C-A  3210.8650979517 -1.28308226288712  0.248167467011978
```

```
##                        stat               pvalue                 padj
##                   <numeric>            <numeric>            <numeric>
## YAR071W   -6.04488831765795 1.49513533271296e-09 6.05623255707042e-07
## YBR054W   -10.6103609327863 2.66716752502983e-26 4.32147818242959e-23
## YEL070W    3.75799115273969 0.000171282935892422   0.0199475661246443
## YER081W    11.8388785764854 2.45716435513872e-32 5.30829406188468e-29
## YFL014W   -7.21898159130019 5.23783445087167e-13 2.73930765105251e-10
## YFR052C-A -5.17022750135574 2.33809180453223e-07 7.21579665960639e-05
```

Save the `DEr4` object in a `results_r4` file

```
write.table(x = DEr4, file = "results_r4.txt", sep = "\t", col.names = NA)
```

Questions

1. How many genes are differentially expressed at the thresholds of `padj < 0.05`? And `padj < 0.1`?

The number of differentially expressed genes at the threshold of `padj < 0.05` is 2172;

```
count.05 <- res[!is.na(res$padj),]
count.05 <- count.05[count.05$padj < 0.05,]
count.05 <- count.05[abs(count.05$log2FoldChange) > 1,]
nrow(count.05)

## [1] 2172
```

The number of differentially expressed genes at the threshold of `padj < 0.1` is 2209.

```
count.1 <- res[!is.na(res$padj),]
count.1 <- count.1[count.1$padj < 0.1,]
count.1 <- count.1[abs(count.1$log2FoldChange) > 1,]
nrow(count.1)

## [1] 2209
```

2. How many genes at the threshold of `padj < 0.05` are upregulated (> 1)?

The **upregulated** genes at the threshold of `padj < 0.05` is 1037.

```
upregulated_count.05 <- res[!is.na(res$padj),]
upregulated_count.05 <- upregulated_count.05[upregulated_count.05$padj < 0.05,]
upregulated_count.05 <- upregulated_count.05[upregulated_count.05$log2FoldChange >
1,]
nrow(upregulated_count.05)

## [1] 1037
```

3. Choice one of the GO enrichment results and report how many categories are significant (`p.adjust < 0.05`). *Hint: first convert the object with GO results with `as.data.frame` function.*

If we choose the *GO enrichment results* of the 2nd comparison and we use the `as.data.frame` function, we select `p.adjust < 0.05` and we use the `nrow` function, we get that the number of *significant categories* for this analysis is 105.

```
GO_BP_2_df <- as.data.frame(GO_BP_2)
sig_cat <- GO_BP_2_df[GO_BP_2_df$p.adjust < 0.05,]
nrow(sig_cat)

## [1] 105
```

4. How many genes are present in the most enriched category of the GO enrichment result?

The number of genes in the most enriched category is 107.

```
GO_BP_2_df$Count[1]

## [1] 107
```

5. The numbers of significant differentially expressed genes of **Step 7** are the same as the results in **Step 5**?

The numbers of significant differentially expressed genes of **Step 5** were **2172** for `padj < 0.05` and **2209** for `padj < 0.1`. In **Step 7** we get **2268** for `padj < 0.05` and **2331** for `padj < 0.1`. *These results are pretty similar but there is a difference of about 100 genes.*

```
new_count.05 <- new_res[!is.na(new_res$padj),]
new_count.05 <- new_count.05[new_count.05$padj < 0.05,]
new_count.05 <- new_count.05[abs(new_count.05$log2FoldChange) > 1,]
nrow(new_count.05)

## [1] 2268

new_count.1 <- new_res[!is.na(new_res$padj),]
new_count.1 <- new_count.1[new_count.1$padj < 0.1,]
new_count.1 <- new_count.1[abs(new_count.1$log2FoldChange) > 1,]
nrow(new_count.1)

## [1] 2331
```

About R Markdown document

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.