BINF6420: DESeq2 and GSEA with clusterProfiler

Rima Zinjuwadia

2024-02-05

Installation

There are a lot of packages that we will be using here. Don’t be worried about it. Most of the packages aren’t actually used directly. Rather they are used as dependencies for other packages.

Non-Bioconductor packages

Say yes to any prompts.

```r
install.packages(c(
    "dplyr",
    "stringr",
    "magrittr",
    "tidyverse",
    "RColorBrewer",
    "pheatmap",
    "ggrepel",
    "cowplot",
    "ggplot2"
))
```

Bioconductor Packages

Say yes to any prompts. If asked for what updates/regions that you want, select the number 1.

```r
if (!require("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
```

```
BiocManager::install(c(
    "devtools",
    "GenomicFeatures",
    "DESeq2",
    "tximport",
    "RMariaDB",
    "biomaRt",
    "clusterProfiler",
    "pathview",
    "enrichplot",
    "msigdbr",
    "org.Hs.eg.db",
    "DOSE",
    "DEGreport",
    "apeglm"),
    update = FALSE
)
```

These installations can/will take a long time to complete. Do not worry if it feels like it is taking too long.

Importing them for use

Now we will load them into the namespace.

Loading annotations

Using `biomaRt` to get the annotations. `biomaRt` (and any other packages associated with `AnnotationHub`) are amazing resources that allow you to temporarily download annotations on an ad-hoc basis and store them as objects in your working environment. While it can sometimes cause "timeout" issues, these are some of the best ways to have access to up-to-date information regarding your model organism.

```
human_mart <-biomaRt::useMart("ENSEMBL_MART_ENSEMBL", dataset = "hsapiens_gene_ensembl",
host="https://useast.ensembl.org")
```

Getting specific annotation entries and creating an annotation file

```
transcript_to_gene <- getBM(
    attributes = c(
        "ensembl_transcript_id",
        "transcript_version",
        "ensembl_gene_id",
        "description",
        "external_gene_name"),
    mart = human_mart)
transcript_to_gene$ensembl_transcript_id <- paste(transcript_to_gene$ensembl_transcript_
id, transcript_to_gene$transcript_version, sep = ".")
```

This code acts as an intermediate annotation file that links transcripts to specific genes.

```
transcript_to_gene$ensembl_transcript_id <- paste(transcript_to_gene$ensembl_transcript_
id, transcript_to_gene$transcript_version, sep = ".")
```

Getting the final form of the annotation file that will be used downstream.

```
t2g <- dplyr::rename(
    transcript_to_gene,
    target_id = ensembl_transcript_id,
    ens_gene = ensembl_gene_id,
    ext_gene = external_gene_name
    )[,c(
        "target_id",
        "ens_gene",
        "description",
        "ext_gene")]

knitr::kable(head(t2g))
```

target_id	ens_gene	description	ext_gene
ENST00000387314.1.1	ENSG00000210049	mitochondrially encoded tRNA-Phe (UUU/C) [Source:HGNC Symbol;Acc:HGNC:7481]	MT-TF
ENST00000389680.2.2	ENSG00000211459	mitochondrially encoded 12S rRNA [Source:HGNC Symbol;Acc:HGNC:7470]	MT-RNR1
ENST00000387342.1.1	ENSG00000210077	mitochondrially encoded tRNA-Val (GUN) [Source:HGNC Symbol;Acc:HGNC:7500]	MT-TV
ENST00000387347.2.2	ENSG00000210082	mitochondrially encoded 16S rRNA [Source:HGNC Symbol;Acc:HGNC:7471]	MT-RNR2
ENST00000386347.1.1	ENSG00000209082	mitochondrially encoded tRNA-Leu (UUA/G) 1 [Source:HGNC Symbol;Acc:HGNC:7490]	MT-TL1
ENST00000361390.2.2	ENSG00000198888	mitochondrially encoded NADH:ubiquinone oxidoreductase core subunit 1 [Source:HGNC Symbol;Acc:HGNC:7455]	MT-ND1

Using `DESeq2`

Prepare and fit the model

BMAT

Load the existing data. Because I don’t think you should have to run `STAR` or `salmon` on some of this real-life dirty data (and because I want to save everyone some time), I have made the data available to you through objects.

```
data_bmat <- readRDS("data_bmat.Rds")
s2c_bmat <- readRDS("s2c_bmat.Rds")
```

Ingest the data into DESeq2

```
dds_bmat <- DESeqDataSetFromMatrix(
    countData = data_bmat,
    colData = s2c_bmat,
    design = ~replicate + co_cultured)
```

Perform differential expression

```
dds_bmat <- DESeq(dds_bmat)
```

Estimates

The first step in the differential expression analysis is to estimate the size factors, which is exactly what we already did to normalize the raw counts.

The next step in the differential expression analysis is the estimation of gene-wise dispersions.

> In RNA-seq count data, we know: 1. To determine differentially expressed genes, we need to identify genes that have significantly different mean expression between groups given the variation within the groups (between replicates). 1. The variation within group (between replicates) needs to account for the fact that variance increases with the mean expression, as shown in the plot below (each black dot is a gene).

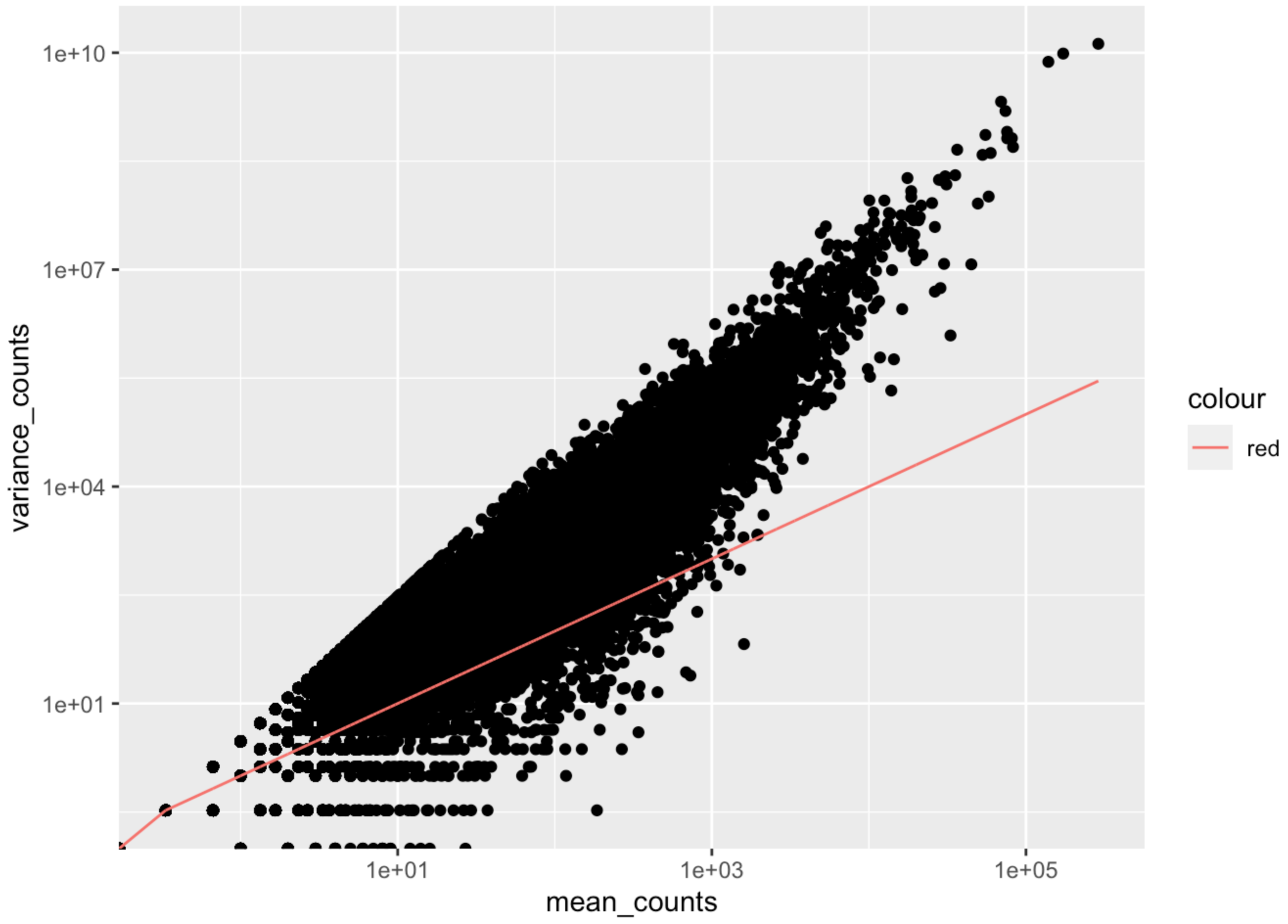
To accurately identify DE genes, DESeq2 needs to account for the relationship between the variance and mean. We don't want all of our DE genes to be genes with low counts because the variance is lower for lowly expressed genes.

Instead of using variance as the measure of variation in the data (since variance correlates with gene expression level), it uses a measure of variation called dispersion, which accounts for a gene's variance and mean expression level.

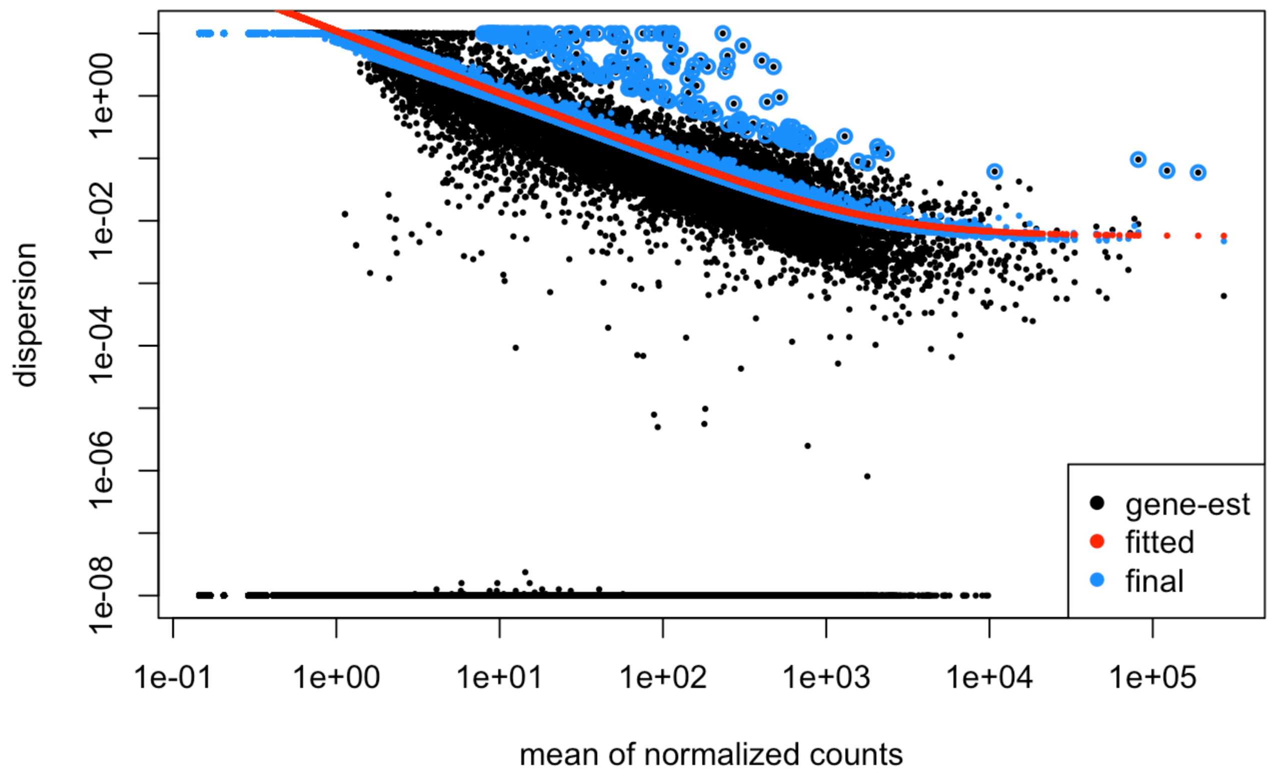
For genes with moderate to high count values, the square root of dispersion will be equal to the coefficient of variation `(Var / μ)`. So 0.01 dispersion means 10% variation around the mean expected across biological replicates. The dispersion estimates for genes with the same mean will differ only based on their variance. Therefore, the dispersion estimates reflect the variance in gene expression for a given mean value (given as black dots in image below).

BMAT

```
mean_counts <- apply(data_bmat[,c(2,4,6)], 1, mean)
variance_counts <- apply(data_bmat[,c(2,4,6)], 1, var)
df <- data.frame(mean_counts, variance_counts)
ggplot(df)+
    geom_point(aes(x=mean_counts, y=variance_counts)) +
    geom_line(aes(x=mean_counts, y=mean_counts, color="red")) +
    scale_y_log10() +
    scale_x_log10()
```



```
plotDispEsts(dds_bmat)
```



There are simple functions in the `DESeq2` package that makes these calculations straight forward and automatically annotates the model for us. When you run the `DESeq` function, it is performing the following commands wrapped up into one.

```
# BMAT
dds_bmat <- estimateSizeFactors(dds_bmat)
dds_bmat <- estimateDispersions(dds_bmat)
dds_bmat <- nbinomWaldTest(dds_bmat)
```

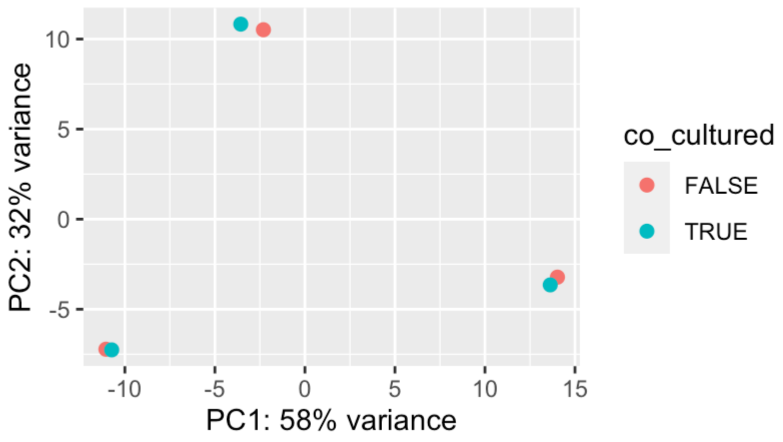
PCA

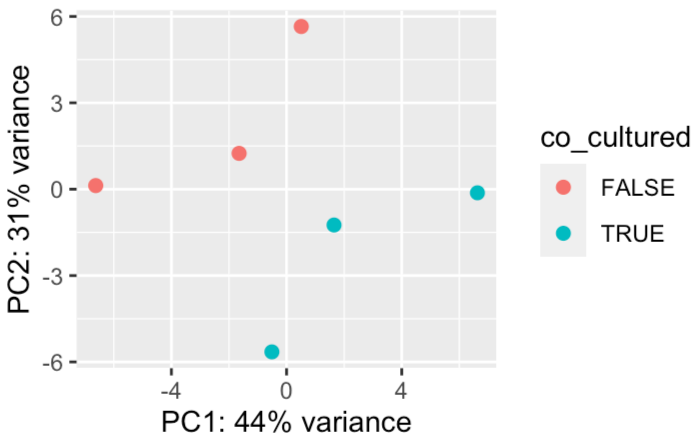
There is a direct way for us to manage the data when using `DESeq2` and `limma` that allows us to actually observe the effect our normalization and regressions have on the data. Let's see the change in the PCA plots when these batch corrections are done.

BMAT

```r
vsd_bmat <- vst(dds_bmat, blind=FALSE)
pca_bmat <- plotPCA(vsd_bmat, intgroup=c("co_cultured", "replicate"), returnData=TRUE)
percentVar <- round(100 * attr(pca_bmat, "percentVar"))
pca_bmat_before <- ggplot(pca_bmat, aes(PC1, PC2, color=co_cultured)) +
    geom_point(size=2) +
    xlab(paste0("PC1: ",percentVar[1],"% variance")) +
    ylab(paste0("PC2: ",percentVar[2],"% variance")) +
    coord_fixed()

assay(vsd_bmat) <- limma::removeBatchEffect(assay(vsd_bmat), vsd_bmat$replicate)

pca_bmat <- plotPCA(vsd_bmat, intgroup=c("co_cultured", "replicate"), returnData=TRUE)
percentVar <- round(100 * attr(pca_bmat, "percentVar"))
pca_bmat_after <- ggplot(pca_bmat, aes(PC1, PC2, color=co_cultured)) +
    geom_point(size=2) +
    xlab(paste0("PC1: ",percentVar[1],"% variance")) +
    ylab(paste0("PC2: ",percentVar[2],"% variance")) +
    coord_fixed()

title <-ggdraw() +
    draw_label(
        "BMAT alone vs co_cultured",
        fontface = 'bold',
        x = 0,
        hjust = 0
    )
plot_grid(title, pca_bmat_before, pca_bmat_after, ncol=1, rel_heights = c(0.1, 1, 1))
```

BMAT alone vs co_cultured





Shrinkage

From HBC Training: > To generate more accurate log2 foldchange estimates, `DESeq2` allows for the **shrinkage of the LFC estimates toward zero** when the information for a gene is low […]. As with the shrinkage of dispersion estimates, LFC shrinkage uses **information from all** genes to generate more accurate estimates. Specifically, the distribution of LFC estimates for all genes is used (as a prior) to shrink the LFC estimates of genes with little information or high dispersion toward more likely (lower) LFC estimates.

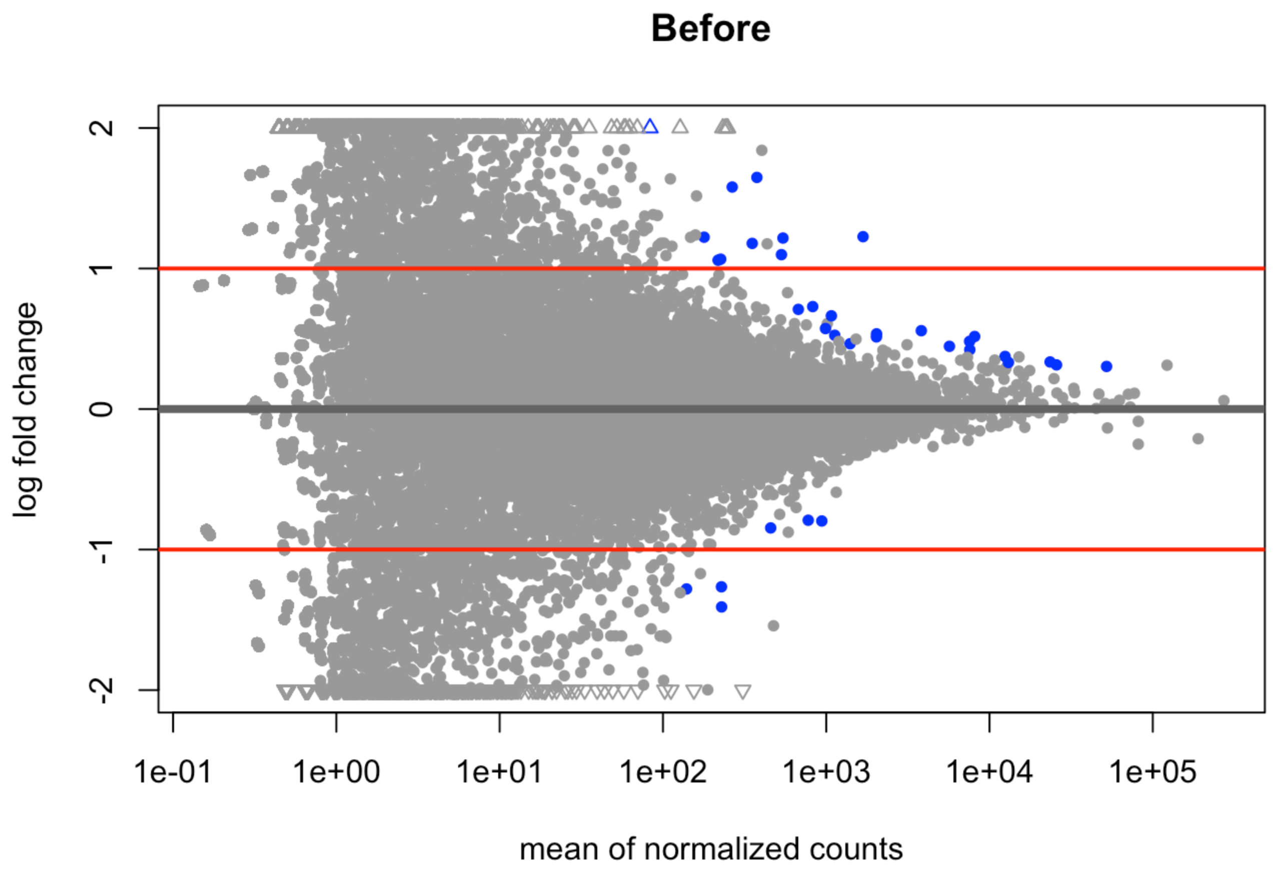
Let's set our condition contrast

```
contrast <- c("co_cultured", "TRUE", "FALSE")
```

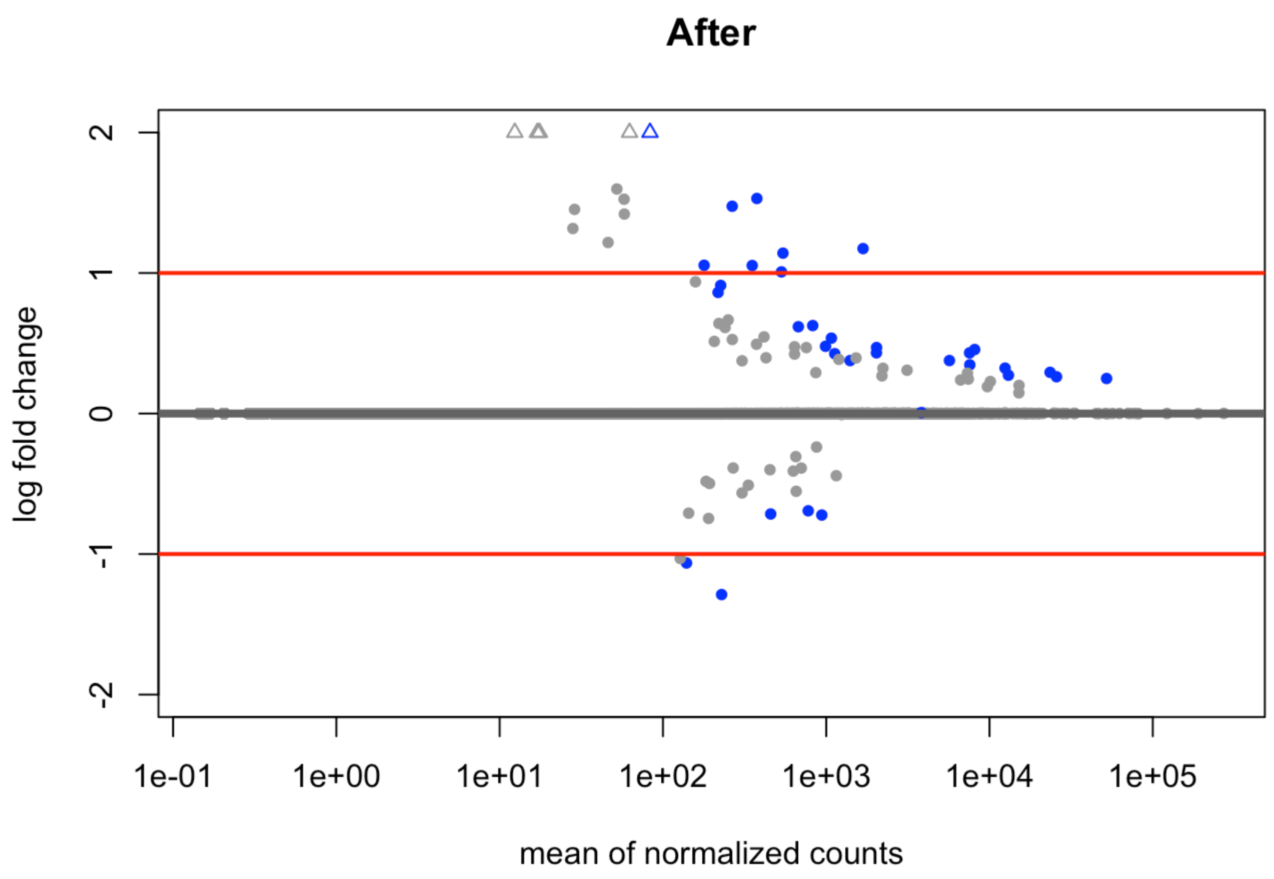
We are going to save our results before and after shrinkage and compare the two to see what effect `lfcShrink` had on the data.

Since this is an under powered pilot study, I am setting the p-value cutoff to 0.1. Anything less than this can be considered of a "low assumption of the null hypothesis." That is, it is *leaning* towards statistical significance.

```
# BMAT
res_bmat <- results(dds_bmat, contrast = contrast, alpha = 0.1)
res_bmat_before <- res_bmat
res_bmat <- lfcShrink(
    dds_bmat,
    res = res_bmat,
    coef = "co_cultured_TRUE_vs_FALSE",
    type = "apeglm")

# Plot the before and after
ma_bmat_before <- plotMA(res_bmat_before, ylim=c(-2,2), cex=.8, main = 'Before') +
    abline(h=c(-1,1), col='red', lwd=2)
```



```
ma_bmat_after <- plotMA(res_bmat, ylim=c(-2,2), cex=.8, main = 'After') +
    abline(h=c(-1,1), col='red', lwd=2)
```



Anything less than 0.1 can be considered of a “low assumption of the null hypothesis.” That is, it is *leaning* towards statistical significance.

Summarizing results

The results after shrinkage are what we will be working with for our GSEA.

`summary`

`summary` gives us a high-level view of what our results are without having to open up a `data.frame` and visually inspect it. It tells us how many genes are over and under DE. The results we see are just the genes that are significantly DE, but no threshold on fold-change has been applied yet.

```
# BMAT
summary(res_bmat, alpha = 0.1)
```

```
##
## out of 25240 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)       : 28, 0.11%
## LFC < 0 (down)     : 6, 0.024%
## outliers [1]       : 0, 0%
## low counts [2]     : 15773, 62%
## (mean count < 74)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

```
# Do some wrangling for downstream analysis
res_bmat_tb <- res_bmat %>%
    data.frame() %>%
    rownames_to_column(var="gene") %>%
    as_tibble()
```

Full DE genes

To get the full list of DE genes that adhere to a fold change threshold, we perform some additional data wrangling.

To explain about `log2foldchange`, you may see a lot of analysis that use `2` as the threshold. However, in the realm of `log2`, a `abs(value) > 1` indicates a doubling/halving over the other. This can still be significant. As such, it is a useful relaxed constraint on results. Since this is still a pilot research project, you can see below how a more stringent cutoff would be a considerable loss of data.

```
# BMAT
DE_bmat <- res_bmat_tb %>%
    filter(padj < 0.1 & abs(log2FoldChange) > 1)

DE_bmat$symbol <- mapIds(
    org.Hs.eg.db,
    keys=DE_bmat$gene,
    column="SYMBOL",
    keytype="ENSEMBL",
    multiVals="first")

DE_bmat$entrez <- mapIds(
    org.Hs.eg.db,
    keys=DE_bmat$gene,
    column="ENTREZID",
    keytype="ENSEMBL",
    multiVals="first")

write.csv(DE_bmat, "DE_BMAT.csv", row.names = T)
knitr::kable(head(DE_bmat))
```

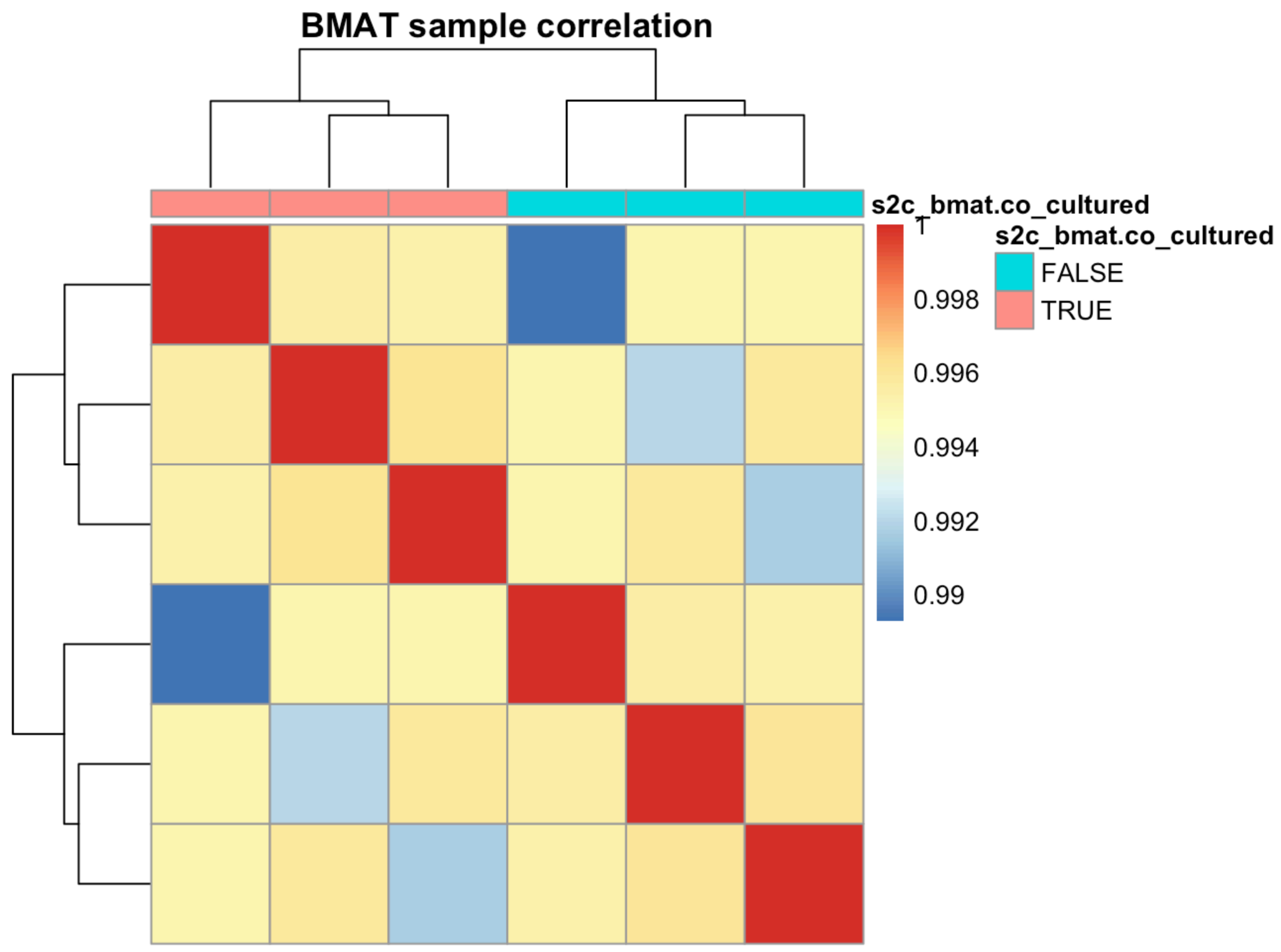
gene	baseMean	log2FoldChange	lfcSE	pvalue	padj	symbol	entrez
ENSG00000099998	352.1375	1.053354	0.2742588	3.6e-06	0.0031566	GGT5	2687

gene	baseMean	log2FoldChange	lfcSE	pvalue	padj	symbol	entrez
ENSG00000101868	139.4418	-1.063507	0.3803569	1.5e-04	0.0525856	POLA1	5422
ENSG00000124333	83.3271	7.171929	2.0807018	1.5e-06	0.0015911	VAMP7	6845
ENSG00000134900	543.3568	1.141291	0.2158531	0.0e+00	0.0000204	TPP2	7174
ENSG00000142949	531.7403	1.008092	0.2243275	2.0e-07	0.0003645	PTPRF	5792
ENSG00000164742	178.7295	1.055091	0.3252670	3.9e-05	0.0217231	ADCY1	107

Sanity checks

Below, we are plotting sample correlation heatmaps. These *should* show a 1:1 diagonal. That is each sample is exactly correlated with itself. Then the heatmaps *should* cluster similar samples together. That is, the “alone” cells should be cluster together and look similar compared to the “co-cultured” cells.

```
vst_mat_bmat <- assay(vsd_bmat)
vst_cor_bmat <- cor(vst_mat_bmat)
bmat_annotation_heatmap <- data.frame(row.names=colnames(vst_mat_bmat), s2c_bmat$co_cult
ured)

pb <- pheatmap(vst_cor_bmat,
         annotation_col = bmat_annotation_heatmap,
         main = "BMAT sample correlation",
         show_rownames = F,
         show_colnames = F)
```



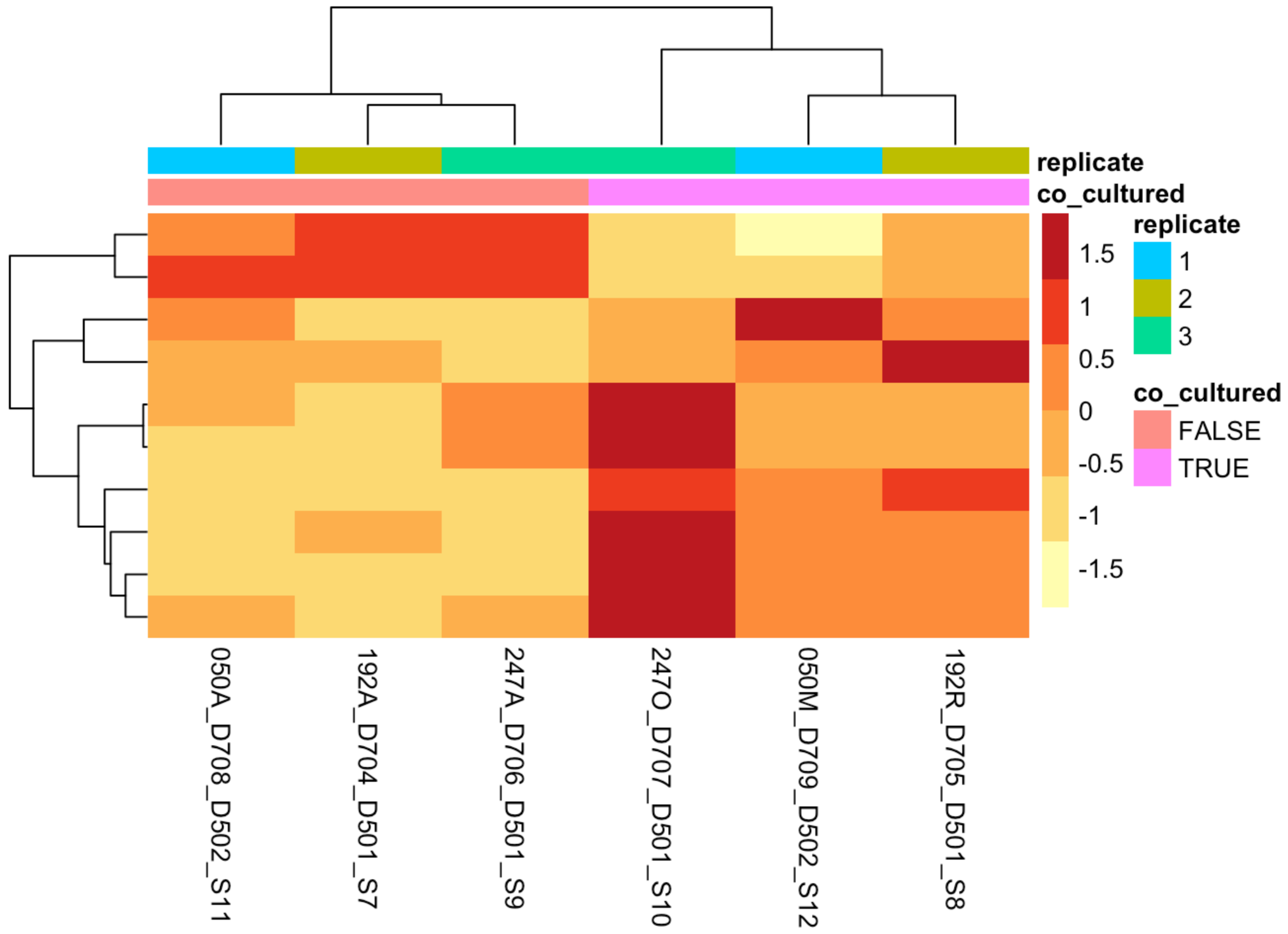
Yet another annotation file

```
grch38annot <- transcript_to_gene %>%
    dplyr::select(ensembl_gene_id, external_gene_name) %>%
    dplyr::distinct()
```

`pheatmap`

Below we are comparing normalized read counts across samples and clustering similar samples together

```r
# BMAT
normalized_counts_bmat <- counts(dds_bmat, normalized=T) %>%
    data.frame() %>%
    rownames_to_column(var="gene")

names(normalized_counts_bmat) <- sub("X", "", names(normalized_counts_bmat))

normalized_counts_bmat <- merge(normalized_counts_bmat, grch38annot, by.x="gene", by.y
="ensembl_gene_id") %>%
    as_tibble()

norm_sig_bmat <- normalized_counts_bmat %>% filter(gene %in% DE_bmat$gene)


heat_colors <- brewer.pal(6, "YlOrRd")
pheatmap(norm_sig_bmat[,2:7],
         color = heat_colors,
         cluster_rows = T,
         show_rownames = F,
         annotation = s2c_bmat,
         border_color = NA,
         fontsize = 10,
         scale = "row",
         fontsize_row = 10,
         height = 20,
         drop_levels = T)
```

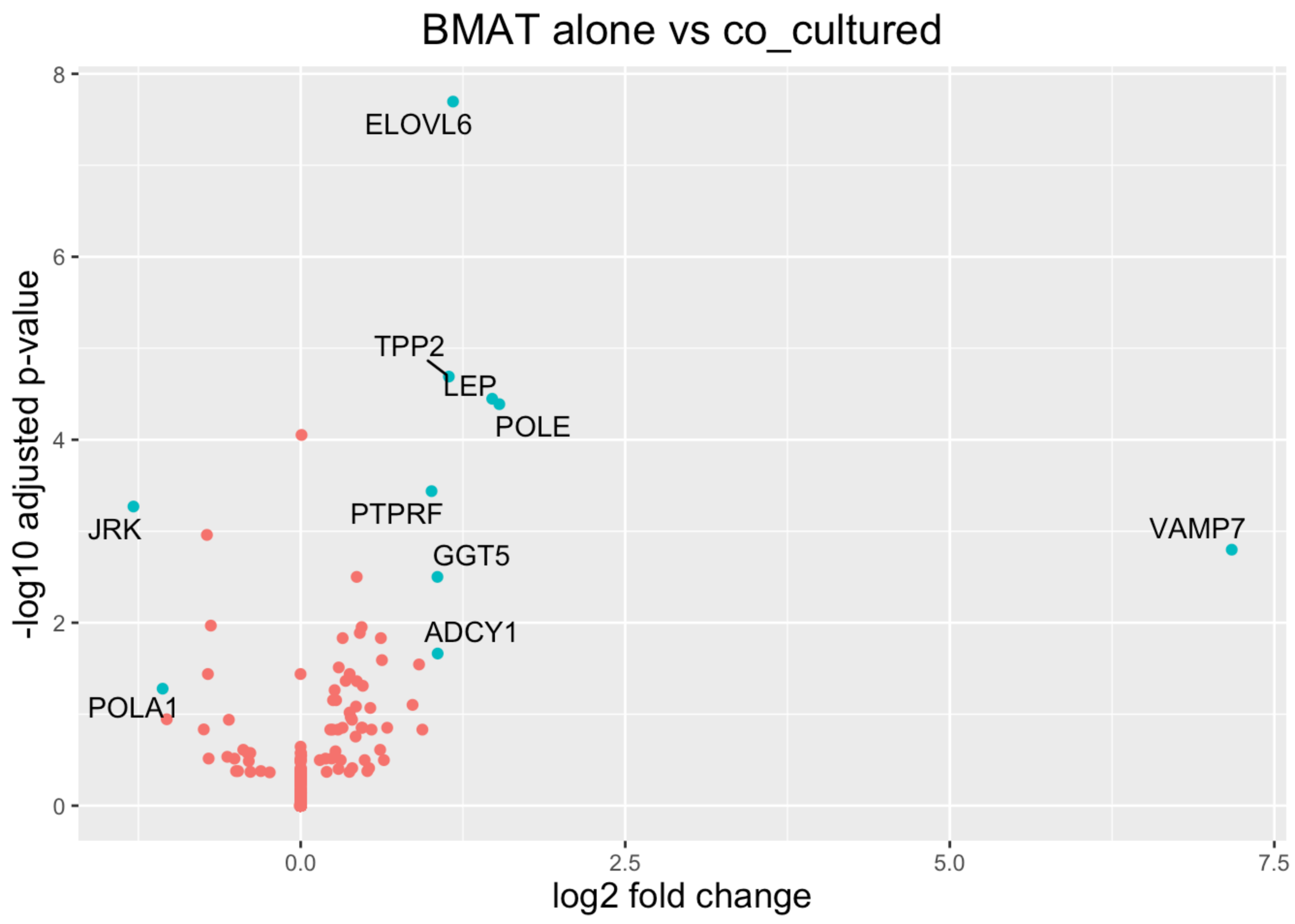


Given this heatmap, what are you takeaways when considering how the samples and genes were clustered?

Volcano plot

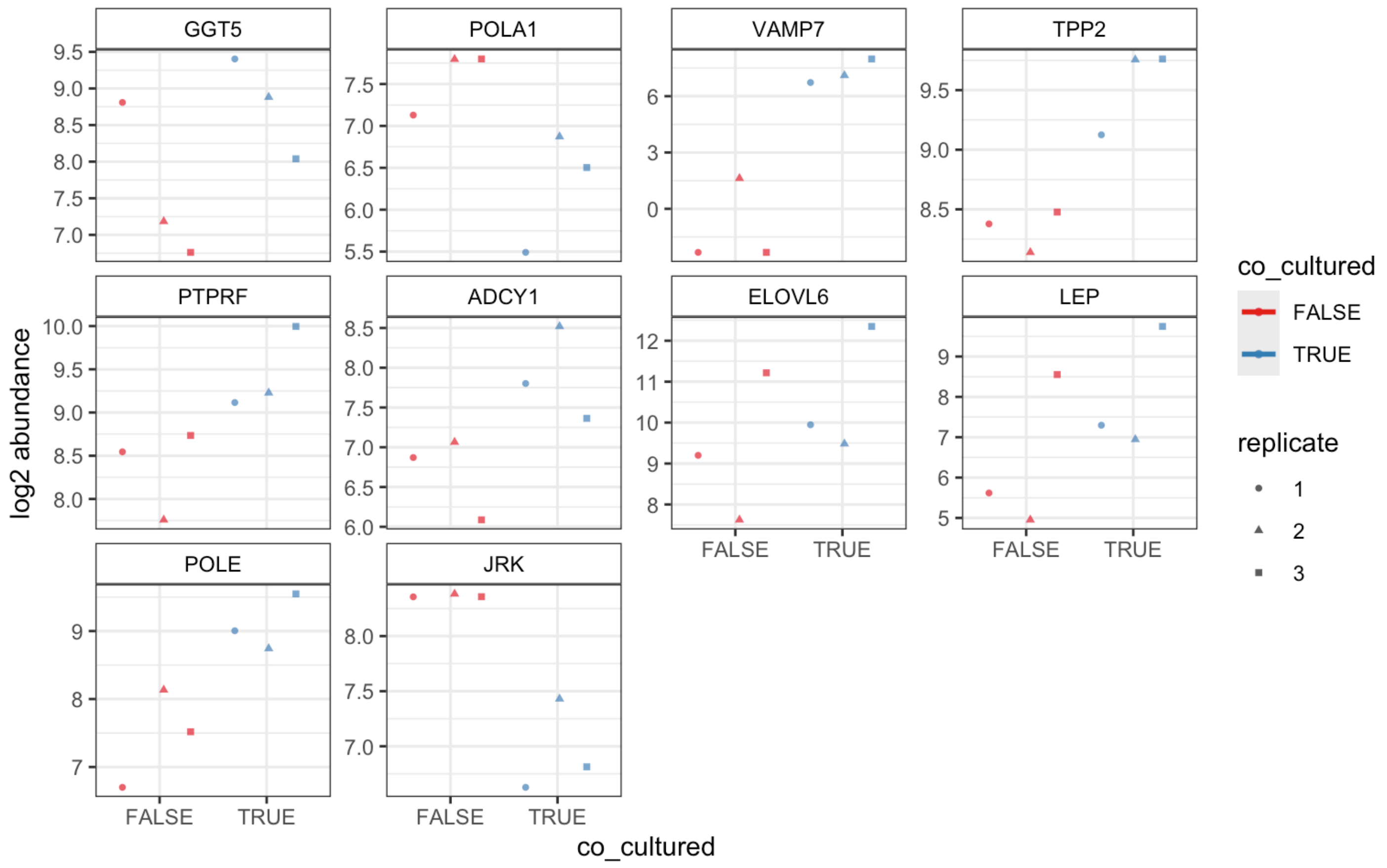
We can use volcano plots to visually see the spread of DE genes, either up or down regulated. The color of the points denotes significance and the label is applied to the genes that met p-value and `log2foldchange` thresholds.

```
# BMAT
res_bmat_tb <- na.omit(res_bmat_tb %>%
    mutate(threshold_OE = padj < 0.1 & abs(log2FoldChange) >= 1))

## Add all the gene symbols as a column from the grch38 table using bind_cols()
res_bmat_tb <- bind_cols(res_bmat_tb, symbol=grch38annot$external_gene_name[match(res_bm
at_tb$gene, grch38annot$ensembl_gene_id)])
res_bmat_tb <- res_bmat_tb %>% mutate(genelabels = "")
res_bmat_tb <- res_bmat_tb %>% arrange(padj)

res_bmat_tb$genelabels[res_bmat_tb$threshold_OE == T] <- as.character(res_bmat_tb$symbol
[res_bmat_tb$threshold_OE == T])

ggplot(res_bmat_tb, aes(x = log2FoldChange, y = -log10(padj))) +
    geom_point(aes(colour = threshold_OE)) +
    geom_text_repel(aes(label = genelabels)) +
    ggtitle("BMAT alone vs co_cultured") +
    xlab("log2 fold change") +
    ylab("-log10 adjusted p-value") +
    theme(legend.position = "none",
          plot.title = element_text(size = rel(1.5), hjust = 0.5),
          axis.title = element_text(size = rel(1.25)))
```

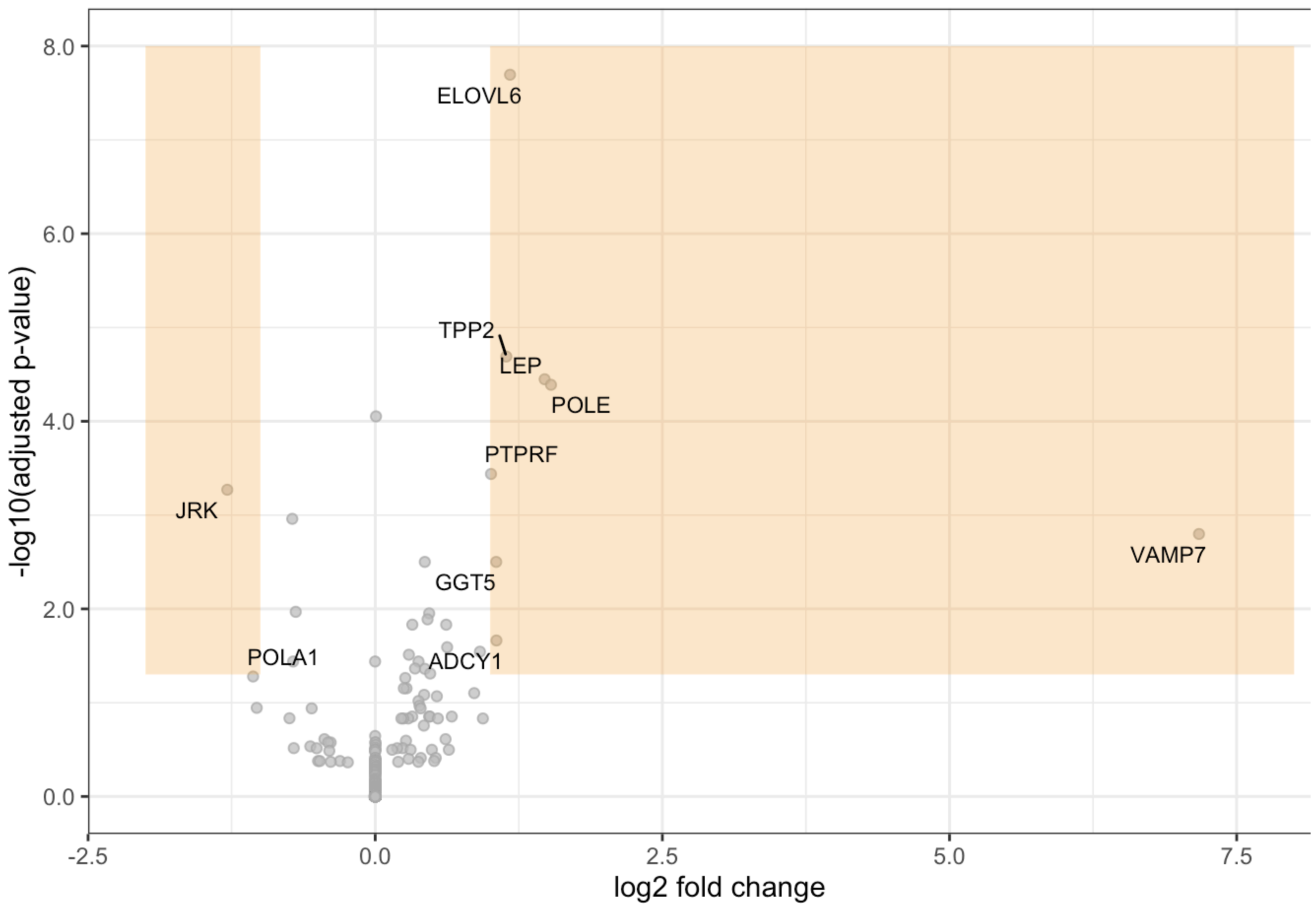


As you can see, generating these plots can be a burden. Thankfully there are some good packages out there (that you have already installed) that make this easier. Introducing you to `DEGreport`.

```
dds_bmat_deg <- dds_bmat

rowData(dds_bmat_deg)$symbol <- mapIds(
    org.Hs.eg.db,
    keys=rownames(dds_bmat_deg),
    column="SYMBOL",
    keytype="ENSEMBL",
    multiVals="first")

rowData(dds_bmat_deg)$gene <- rownames(dds_bmat_deg)

degPlot(
    dds=dds_bmat_deg,
    res=DE_bmat,
    n = nrows(DE_bmat),
    genes = DE_bmat$gene,
    xs = "co_cultured",
    group = "co_cultured",
    ann = c("gene", "symbol"),
    batch = "replicate")
```



```
# res_bmat_tb$genelabels[1:20] <- as.character(res_bmat_tb$symbol[1:20])
degVolcano(data.frame(res_bmat_tb[,c("log2FoldChange","padj")]),
           plot_text=data.frame(res_bmat_tb[res_bmat_tb$threshold_OE == T,c("log2FoldCha
nge","padj","genelabels")]))
```



Gene Set Enrichment Analysis

Because of the small number of DE genes that we have identified, programmatic analysis of the DE genes is somewhat constrained and results will be minimum.

That said, we can attempt to see some of these packages in action for the purposes of introducing you to them.

```r
# BMAT
original_gene_list_bmat <- DE_bmat$log2FoldChange
names(original_gene_list_bmat) <- DE_bmat$gene
gene_list_bmat <- na.omit(original_gene_list_bmat)
gene_list_bmat <- sort(original_gene_list_bmat, decreasing = T)

gse_bmat_enrich <- enrichGO(names(gene_list_bmat),
             ont ="BP",
             keyType = "ENSEMBL",
             minGSSize = 3,
             maxGSSize = 800,
             pvalueCutoff = 0.1,
             OrgDb = org.Hs.eg.db,
             pAdjustMethod = "none")
gse_bmat <- gseGO(geneList=gene_list_bmat,
             ont ="BP",
             keyType = "ENSEMBL",
             minGSSize = 3,
             maxGSSize = 800,
             pvalueCutoff = 1,
             verbose = TRUE,
             OrgDb = org.Hs.eg.db,
             pAdjustMethod = "none")
```

```
## preparing geneSet collections...
```

```
## GSEA analysis...
```

```
## leading edge analysis...
```

```
## done...
```

```r
gse_bmat_pairwise <- pairwise_termsim(gse_bmat)

# Converting some names from Ensembl to Entrez, which is used by KEGG
original_gene_list_bmat_entrez <- original_gene_list_bmat
names(original_gene_list_bmat_entrez) <- DE_bmat$entrez
gene_list_bmat_entrez <- na.omit(original_gene_list_bmat_entrez)
gene_list_bmat_entrez <- sort(original_gene_list_bmat_entrez, decreasing = T)
gse_bmat_do <- gseDO(gene_list_bmat_entrez, pvalueCutoff = 1)
```

```
## preparing geneSet collections...
```

```
## GSEA analysis...
```

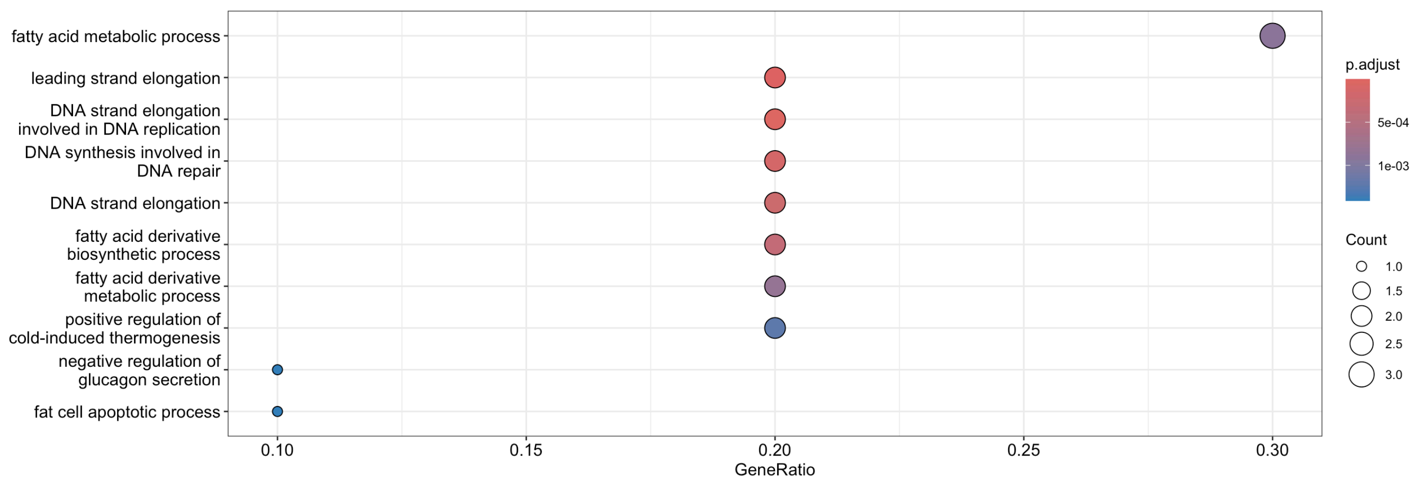
```
## no term enriched under specific pvalueCutoff...
```

Considering Gene Ontology

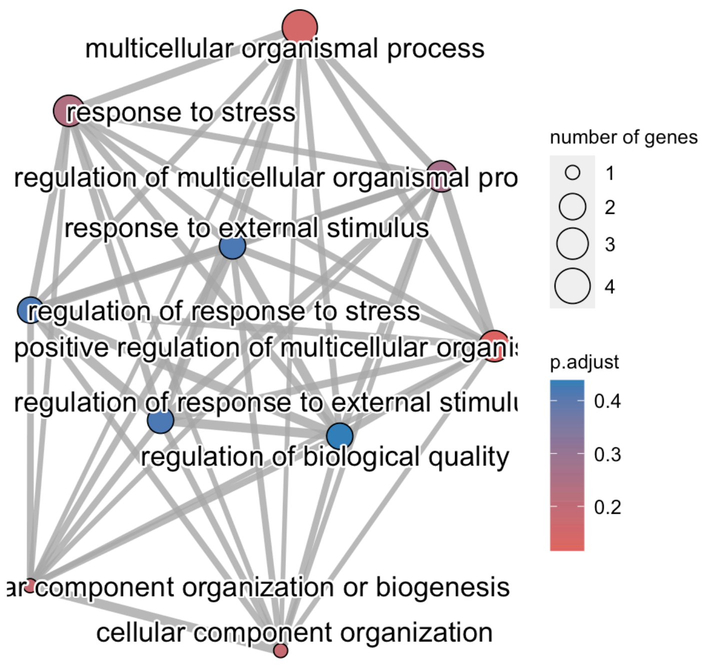
Once the initial GSEA is complete, we can explore some of the different plots that it can generate.

> A word of caution: These results may appear as informative, but take it all with a grain of salt. Remember that we strongly relaxed our cutoffs for p values. Furthermore, in the `gseGO` calls above, we have it returning anything with a p value after enrichment (`pvalueCutoff = 1`). This is because without these details, we would not have much to explore.

```
dotplot(gse_bmat_enrich, showCategory=10)
```

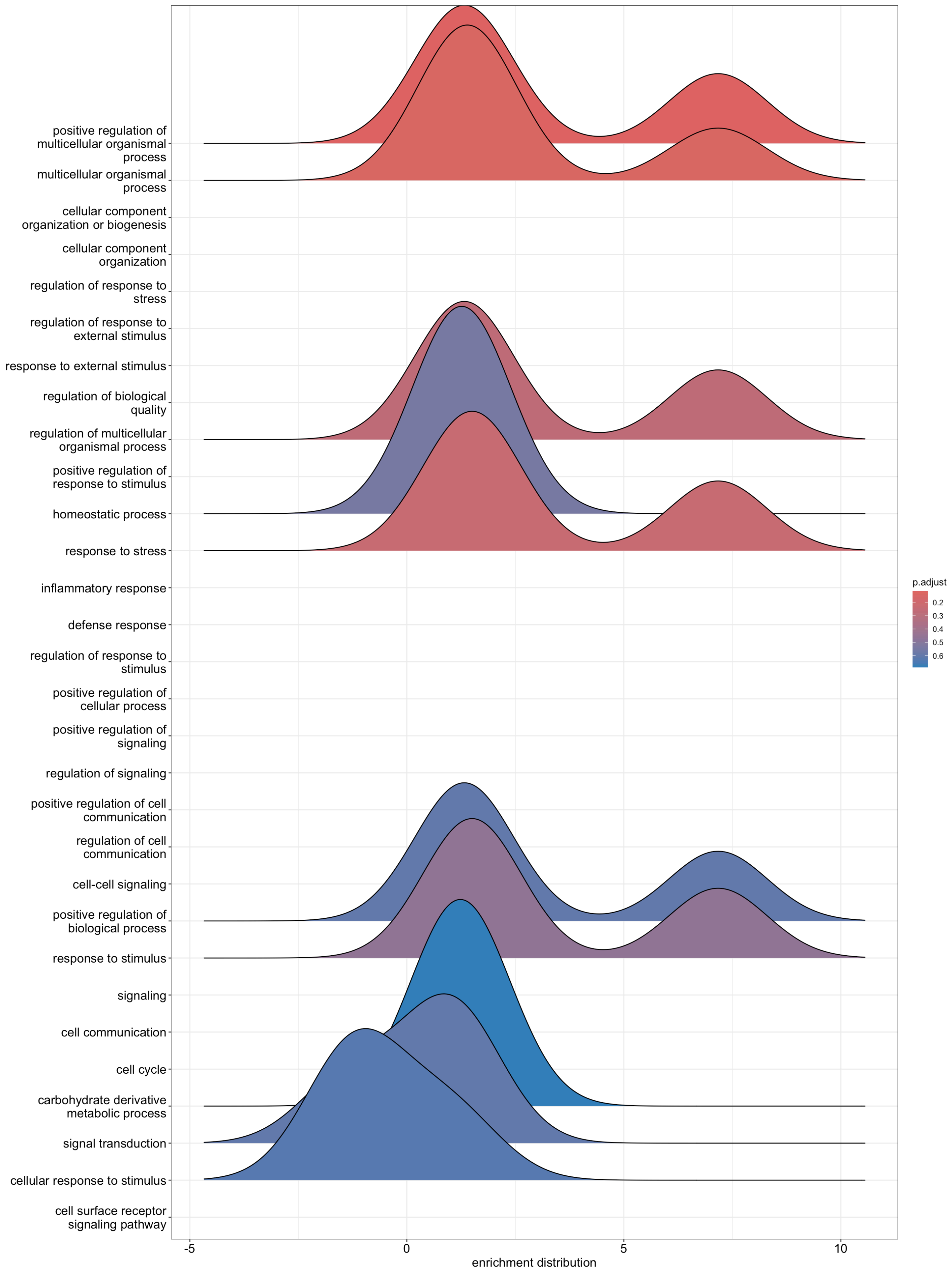


```
emapplot(gse_bmat_pairwise, font.size = 6, showCategory=10) + scale_color_gradient()
```



```
ridgeplot(gse_bmat) + labs(x = "enrichment distribution")
```

```
## Picking joint bandwidth of 1.13
```



positive regulation of multicellular organismal process
multicellular organismal process
cellular component organization or biogenesis
cellular component organization
regulation of response to stress
regulation of response to external stimulus
response to external stimulus
regulation of biological quality
regulation of multicellular organismal process
positive regulation of response to stimulus
homeostatic process
response to stress
inflammatory response
defense response
regulation of response to stimulus
positive regulation of cellular process
positive regulation of signaling
regulation of signaling
positive regulation of cell communication
regulation of cell communication
cell-cell signaling
positive regulation of biological process
response to stimulus
signaling
cell communication
cell cycle
carbohydrate derivative metabolic process
signal transduction
cellular response to stimulus
cell surface receptor signaling pathway
-5
0
5
10
enrichment distribution
p.adjust
0.2
0.3
0.4
0.5
0.6

Consider KEGG Pathways

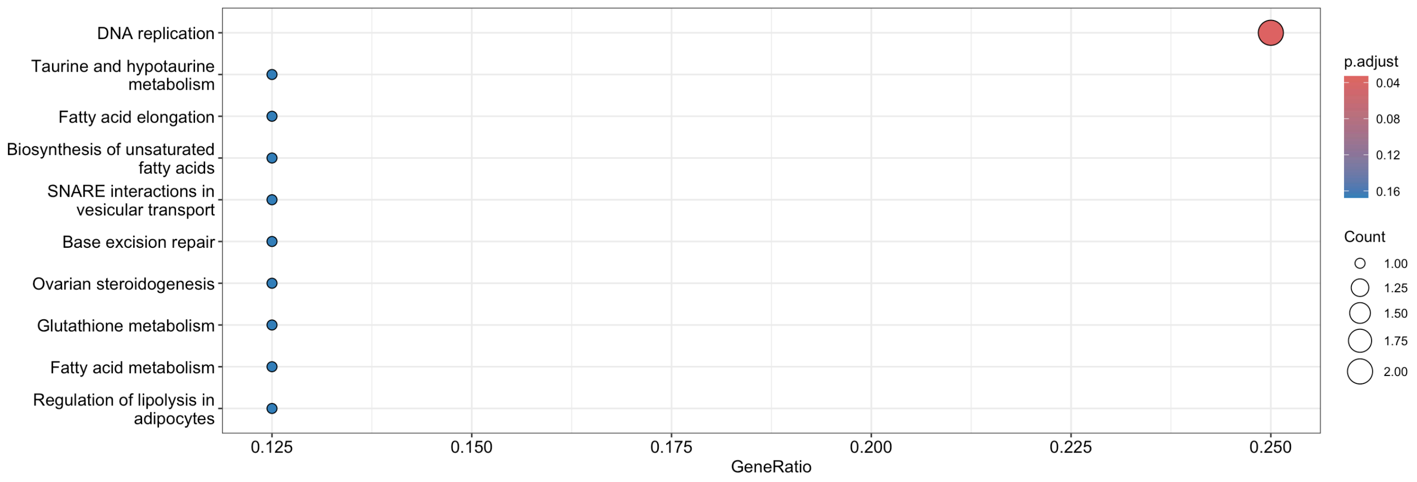
```
kk_bmat<- enrichKEGG(names(gene_list_bmat_entrez),
               organism     = "hsa",
               minGSSize    = 3,
               maxGSSize    = 800,
               pvalueCutoff = 1,
               pAdjustMethod = "BH",
               keyType       = "kegg")
```

```
## Reading KEGG annotation online: "https://rest.kegg.jp/link/hsa/pathway"...
```

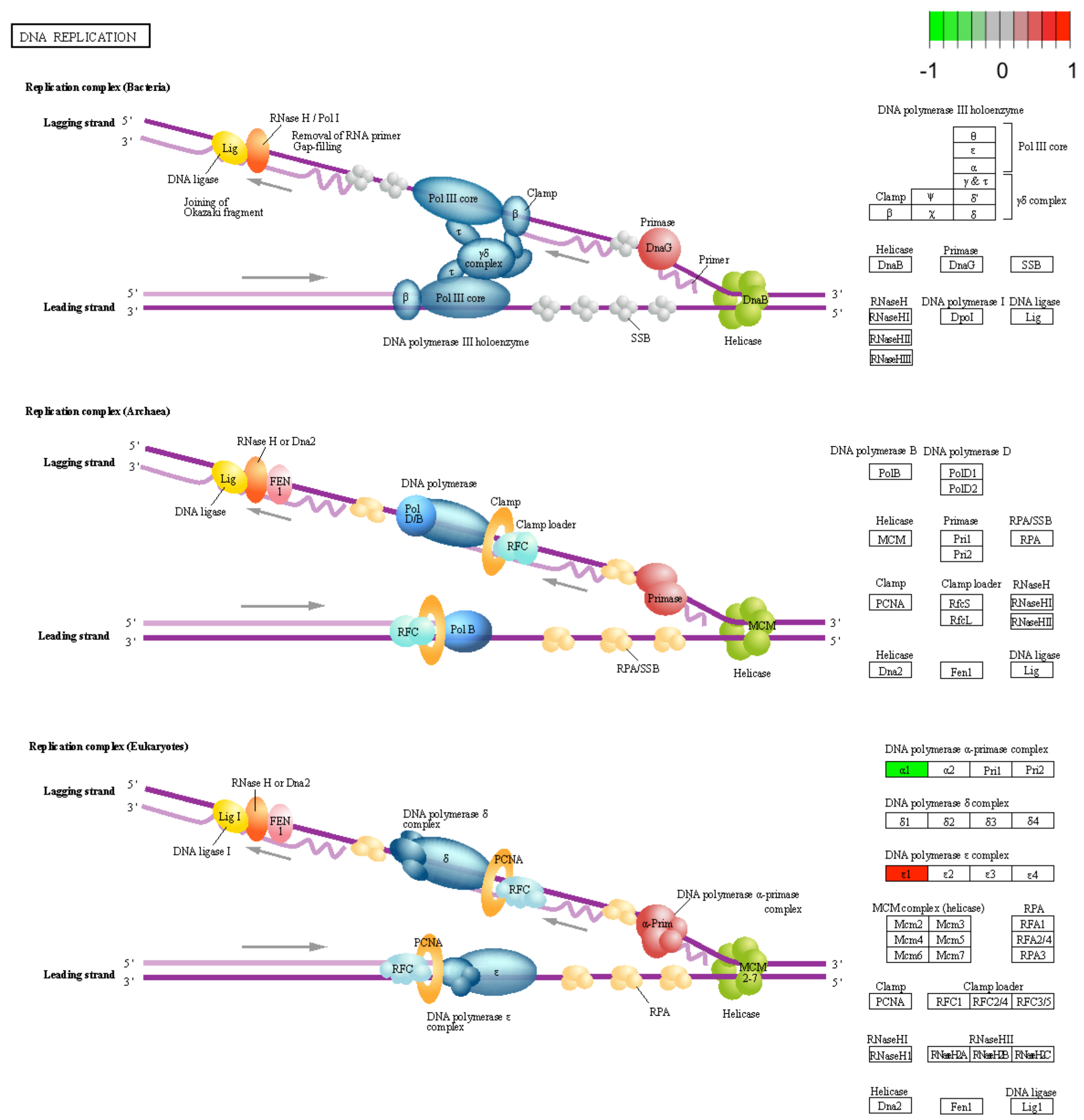
```
## Reading KEGG annotation online: "https://rest.kegg.jp/list/pathway/hsa"...
```

> Notice the p values. How do you interpret these?

```
dotplot(kk_bmat, showCategory=10)
```



```
pathview(gene_list_bmat_entrez, pathway.id = "hsa03030", species = "hsa")
knitr::include_graphics("hsa03030.pathview.png")
```



Data on KEGG graph
Rendered by Pathview