RNA-seq Data Analysis using DESeq2

Pang-Kuo Lo

11/29/2019

RNA-seq Differential Expression Analysis

Load the following packages **DESeq2, RColorBrewer, pheatmap, and tidyverse** for RNA-seq differential expression analysis.

```
# Load library for DESeq2
library(DESeq2)

# Load library for RColorBrewer
library(RColorBrewer)

# Load library for pheatmap
library(pheatmap)

# Load library for tidyverse
library(tidyverse)
```

Create an object for RNA-seq rawcount data

The raw RNA-seq count data can be downloade from the [link] (https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE85209 (https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE85209))

```
# Create an empty data frame for storing raw data count data
wt_rawcounts <- read.table("GSM2260473_WT_normal_1.rawcounts.txt", stringsAsFactors = FALSE)
wt_rawcounts <- wt_rawcounts[1]
class(wt_rawcounts)
```

```
## [1] "data.frame"
```

```
head(wt_rawcounts)
```

```
##                   V1
## 1 ENSMUSG00000102693
## 2 ENSMUSG00000064842
## 3 ENSMUSG00000051951
## 4 ENSMUSG00000102851
## 5 ENSMUSG00000103377
## 6 ENSMUSG00000104017
```

```
nrow(wt_rawcounts)
```

```
## [1] 47729
```

```
# Create an object for file names of raw count data
file_names <- read.table("file_names.txt", stringsAsFactors = FALSE, header = TRUE)

# Store raw count data to the wt_rawcounts
for (i in 1:7) {
  df <- read.table(file_names[i,1])
  wt_rawcounts <- cbind(wt_rawcounts, df[2])
}

# Create a gene_id object
gene_id <- wt_rawcounts[,1]
head(gene_id)
```

```
## [1] "ENSMUSG00000102693" "ENSMUSG00000064842" "ENSMUSG00000051951"
## [4] "ENSMUSG00000102851" "ENSMUSG00000103377" "ENSMUSG00000104017"
```

```
# Assign gene_id as row names to wt_rawcounts
wt_rawcounts <- wt_rawcounts[2:8]
rownames(wt_rawcounts) <- gene_id

# Assign sample names as column names to wt_rawcounts
colnames(wt_rawcounts) <- paste0(rep(c("normal", "fibrosis"), c(3,4)), "_", c(1:3, 1:4))
head(wt_rawcounts)
```

```
##                    normal_1 normal_2 normal_3 fibrosis_1 fibrosis_2 fibrosis_3
## ENSMUSG00000102693        0        0        0          0          0          0
## ENSMUSG00000064842        0        0        0          0          0          0
## ENSMUSG00000051951        3        1        1         42         52         16
## ENSMUSG00000102851        0        0        0          0          0          0
## ENSMUSG00000103377        0        0        0          0          0          0
## ENSMUSG00000104017        0        0        0          0          0          0
##                    fibrosis_4
## ENSMUSG00000102693          0
## ENSMUSG00000064842          0
## ENSMUSG00000051951         35
## ENSMUSG00000102851          0
## ENSMUSG00000103377          0
## ENSMUSG00000104017          0
```

Differential Gene Expression (DGE) Theory: Metadata

Use the information below to create a metadata data frame for the fibrosis count data called metadata with columns genotype and condition. The sample names (e.g. smoc2_fibrosis1, smoc2_fibrosis2, etc.) should be the row names of the data frame:

- Create a character vector called genotype for the above data using c().
- Create a character vector called condition for the above data using c().
- Create a data frame called smoc2_metadata using data.frame() and the genotype and condition character vectors.
- Create a vector of sample names using c() and assign it to the row names of the data frame using rownames().

```
# Create genotype vector
genotype <- rep("wt", 7)

# Create condition vector
condition <- rep(c("normal", "fibrosis"), c(3,4))

# Create data frame
wt_metadata <- data.frame(genotype, condition)

# Assign the row names of the data frame
rownames(wt_metadata) <- paste0(rep(c("normal", "fibrosis"), c(3,4)), "_", c(1:3, 1:4))
wt_metadata
```

```
##            genotype condition
## normal_1         wt    normal
## normal_2         wt    normal
## normal_3         wt    normal
## fibrosis_1       wt  fibrosis
## fibrosis_2       wt  fibrosis
## fibrosis_3       wt  fibrosis
## fibrosis_4       wt  fibrosis
```

DESeq2

DESeq2 estimates variance-mean dependence in count data from high-throughput sequence assays and tests for differential expression based on a model using the negative binomial distribution.

Documentation of DESeq2 can be viewed using the following syntax:

```
**vignette("DESeq2")**
```

The analysis pipeline for DESeq2

- Read counts associated with genes
- Normalization (Quality control)
- Unsupervised clustering analysis
- Modelling raw counts for each gene (DE analysis)
- Shrinking log2 fold changes (DE analysis)
- Testing for differential expression

In these exercises, gene expression differences between the normal and fibrosis samples of wild-type mice will be explored. Gene expression between the normal and fibrosis samples from mice over-expressing the smoc2 gene will be explored.

Create a DESeq2 object

To perform any analysis with **DESeq2**, we need to create a **DESeq2 object** by providing the raw counts, metadata, and design formula. To do this, we need to read in the raw counts data and associated metadata we created previously, make sure the sample names are in the same order in both datasets, then create a DESeq2 object to use for differential expression analysis. We will use the design formula `~ condition` to test for differential expression between conditions (normal and fibrosis).

```
wt_dge <- DESeqDataSetFromMatrix(countData = wt_rawcounts,
                                 colData =  wt_metadata,
                                 design = ~ condition)
```

Normalizing counts with DESeq2

After creating the DESeq2 object, quality control needs to be performed on samples. Therefore, we need to generate the normalized counts (normalized for library size, which is the total number of gene counts per sample, while accounting for library composition). To obtain the normalized counts, use the DESeq2 object and generate the normalized counts matrix.

The `estimateSizeFactors()` function is used to estimate the size factors of raw count data.

```
wt_dge <- estimateSizeFactors(wt_dge)

wt_dge_normalized <- counts(wt_dge, normalized = TRUE)
head(wt_dge_normalized)
```

```
##                    normal_1 normal_2  normal_3 fibrosis_1 fibrosis_2 fibrosis_3
## ENSMUSG00000102693 0.000000 0.000000 0.0000000    0.00000    0.00000    0.00000
## ENSMUSG00000064842 0.000000 0.000000 0.0000000    0.00000    0.00000    0.00000
## ENSMUSG00000051951 3.285193 1.379266 0.9577519   37.01722   43.12125   13.63828
## ENSMUSG00000102851 0.000000 0.000000 0.0000000    0.00000    0.00000    0.00000
## ENSMUSG00000103377 0.000000 0.000000 0.0000000    0.00000    0.00000    0.00000
## ENSMUSG00000104017 0.000000 0.000000 0.0000000    0.00000    0.00000    0.00000
##                    fibrosis_4
## ENSMUSG00000102693     0.0000
## ENSMUSG00000064842     0.0000
## ENSMUSG00000051951    37.1603
## ENSMUSG00000102851     0.0000
## ENSMUSG00000103377     0.0000
## ENSMUSG00000104017     0.0000
```

Hierarchical heatmap by condition

When performing quality assessment of RNA-seq count data, we need to transform the normalized counts for better visualization of the variance for **unsupervised clustering analyses**. To assess the similarity of the RNA-seq samples using **hierarchical heatmaps**, transform the normalized counts and perform hierarchical clustering analysis. The pheatmap() function is used to create hierarchical heatmaps after all libraries have been loaded, the DESeq2 object created, and the size factors have been stored in the DESeq2 object.

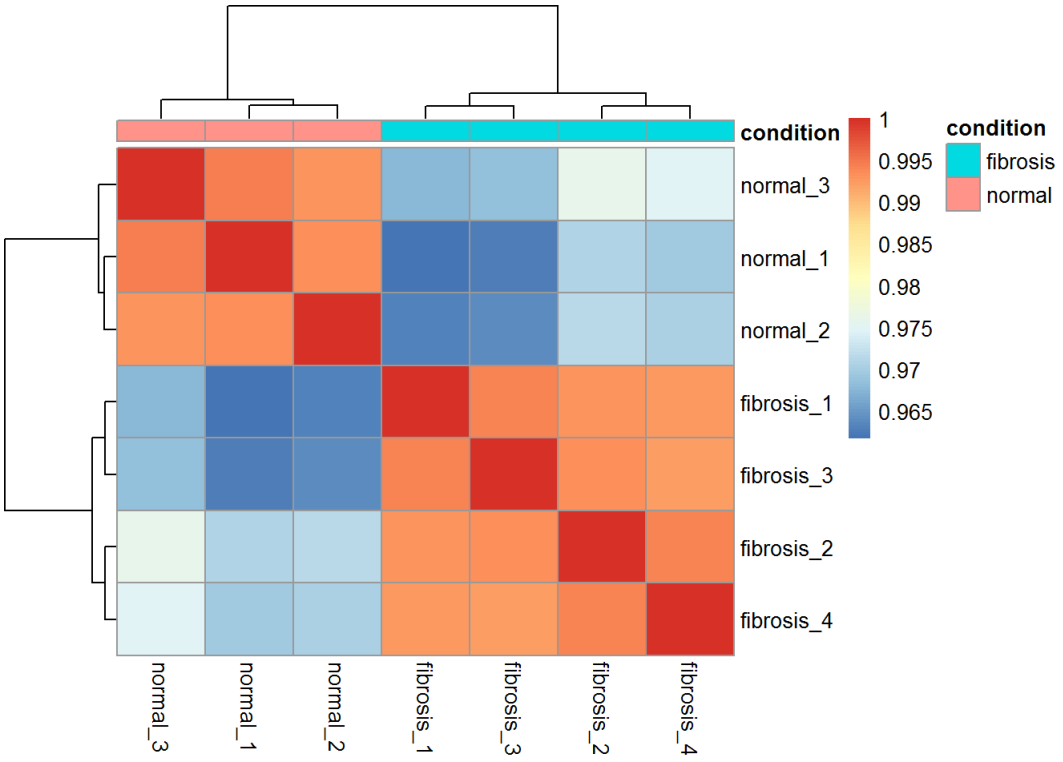
The workflow for creating a hierarchical heatmap * Transform the normalized counts from the DESeq2 object using the `vst()` **(variance stabilizing transformation)** function with the `blind argument` and save a `vsd` . * Extract the matrix of transformed normalized counts from the `vsd` object using the `assay()` function and save as a `vsd matrix` object. * Calculate **the correlation values between samples** and save as a `vsd_cor` object. * Create **a heatmap of the correlation values** using `pheatmap()` with an annotation bar designating condition from the `metadata` data frame.

```
# Transform the normalized counts
vsd_wt <- vst(wt_dge, blind = TRUE)

# Extract the matrix of transformed counts
vsd_mat_wt <- assay(vsd_wt)

# Compute the correlation values between samples
vsd_cor_wt <- cor(vsd_mat_wt)

# Plot the heatmap
pheatmap(vsd_cor_wt, annotation = select(wt_metadata, condition))
```



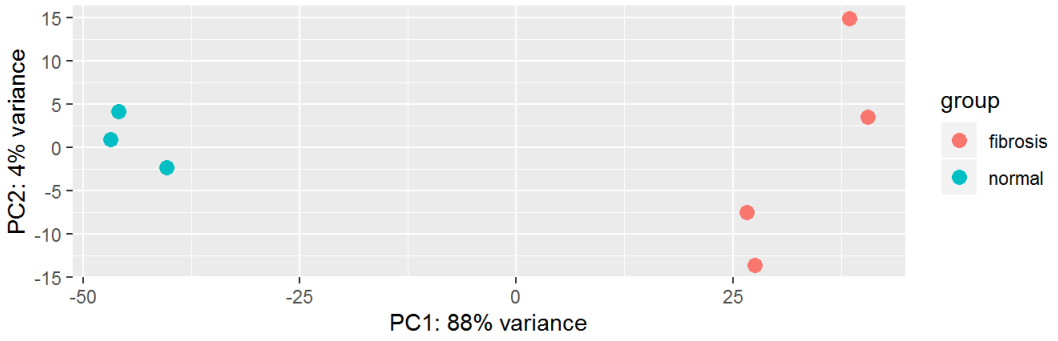
Principal Component Analysis (PCA)

Principal component analysis (PCA) is an unsupervised machine learning algorithm to perform reduction dimensionality that can be used to examine the similarity between samples.

To assess the similarity of the RNA-seq samples using PCA, we need to transform the normalized counts using vst() then perform the PCA analysis.

Perform PCA by plotting PC1 vs PC2 using the `DESeq2 plotPCA()` function on the DESeq2 transformed counts object and specify the `intgroup` argument as the factor to color the plot.

```
# Plot the PCA of PC1 and PC2
plotPCA(vsd_wt, intgroup="condition")
```



Creating the DE object

Use RNA-seq raw count data and metadata to create the DESeq2 object such that the design formula specifies the comparison of the expression differences between the fibrosis and normal samples.

- Create a DESeq2 object using the `DESeqDataSetFromMatrix()` function by specifying the arguments: `countData` , `colData` , and `design` .
- Run the `DESeq()` function to estimate the size factors, calculate the dispersions, and perform the model fitting and testing.

```
# Create DESeq2 object
wt_dge <- DESeqDataSetFromMatrix(countData = wt_rawcounts,
                                 colData =  wt_metadata,
                                 design = ~ condition)

# Run the DESeq2 analysis
de_wt <- DESeq(wt_dge)
```

```
## estimating size factors
```

```
## estimating dispersions
```

```
## gene-wise dispersion estimates
```

```
## mean-dispersion relationship
```

```
## final dispersion estimates
```

```
## fitting model and testing
```

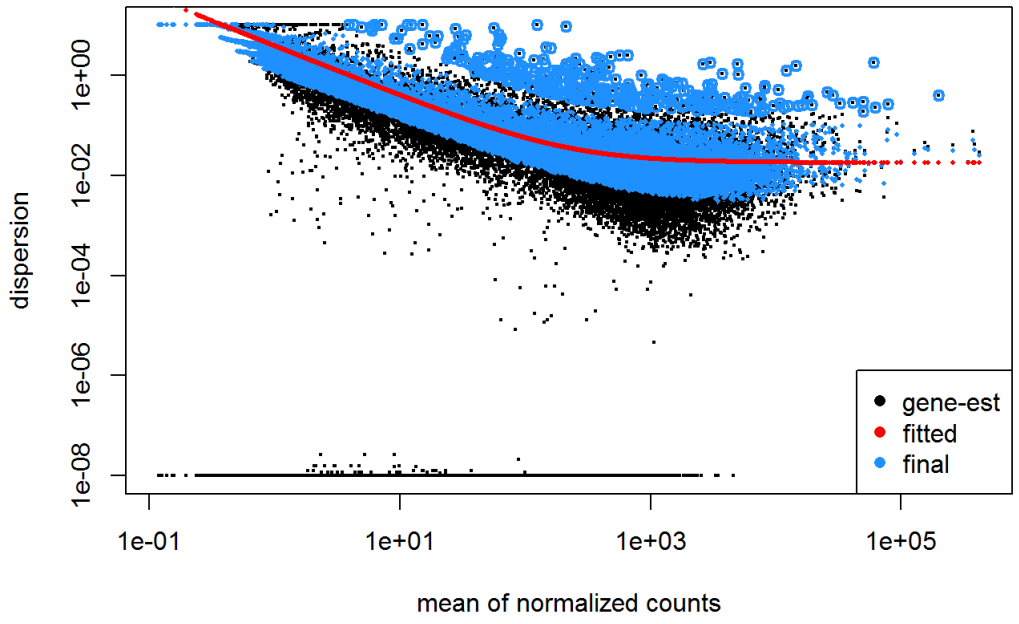
DESeq2 model - exploring dispersions

After fitting the model using DESeq(), explore the fit of our data to the **negative binomial model** by plotting the **dispersion estimates** using the `plotDispEsts()` function. Remember that the dispersion estimates are used to model the raw counts; if the dispersions don't follow the assumptions made by DESeq2, then the variation in the data could be poorly estimated and the DE results could be less accurate.

The assumptions DESeq2 makes are that the dispersions should generally decrease with increasing mean and that they should more or less follow the fitted line.

- Plot the dispersion estimates for the DESeq2 data using the `plotDispEsts()` function after all prior steps have been executed, including the creation of the DESeq2 object, and running the `DESeq()` function.

```
# Plot dispersions
plotDispEsts(de_wt)
```



DESeq2 model - extracting results

After exploring the dispersions and deciding the data fits the DESeq2 model well, the results can be extracted.

After all prior steps have been executed, including the creation of the DESeq2 object and running the `DESeq()` function, the extraction of results can be performed.

Use the `results()` function to specify the `contrast` for the comparison using an `alpha of 0.05`. For the condition of interest, condition, output the results for the fibrosis sample group relative to the normal sample group, so that the normal sample group is the base level.

```
# Extract the results of the differential expression analysis
wt_res <- results(de_wt,
                  contrast = c("condition", "fibrosis", "normal"),
                  alpha = 0.05)
```

DESeq2 results - LFC shrinkage

To improve the fold change estimates for DESeq2 data, the log2 fold changes in the results can be shrunk using the `lfcShrink()` function.

After all prior steps have been executed, including the creation of the DESeq2 object, running the `DESeq()` function, and extracting the results, LFC shrinkage can be performed using the `lfcShrink()` function.

```
# Shrink the log2 fold change estimates to be more accurate
wt_res <- lfcShrink(de_wt,
                    contrast =  c("condition", "fibrosis", "normal"),
                    res = wt_res)
```

```
## using 'normal' for LFC shrinkage, the Normal prior from Love et al (2014).
##
## Note that type='apeglm' and type='ashr' have shown to have less bias than type='normal'.
## See ?lfcShrink for more details on shrinkage type, and the DESeq2 vignette.
## Reference: https://doi.org/10.1093/bioinformatics/bty895
```

DESeq2 Results - Exploration Data Analysis

A log2 fold change threshold can be set to reduce the number of DE genes and to reduce the likelihood of the DE genes being biologically meaningful.

Extract the significant DE results using the `results()` function, similar to before, with an alpha of 0.05 and with normal as the base level of condition. However, this time use a `log2 fold change threshold` of `0.32` (equivalent to 1.25 fold) . After all prior steps have been executed, including the creation of the DESeq2 object, and running the `DESeq()` function, the `results()` can be performed.

Perform shrinkage of the log2 foldchanges using the `lfcShrink()` .

```
# Extract results
wt_res <- results(de_wt,
                  contrast = c("condition", "fibrosis", "normal"),
                  alpha = 0.05,
                  lfcThreshold = 0.32)

# Shrink the log2 fold changes
wt_res <- lfcShrink(de_wt,
                    contrast = c("condition", "fibrosis", "normal"),
                    res = wt_res)
```

Summarizing DESeq2 results

The `summary()` function can be used to get a nice overview of the number of differentially expressed genes there are for the designated alpha level and the threshold of a log2 fold change. It will output the numbers/percentages of up- and down-regulated genes, as well as, give information about independent filtering and outliers removed.

Check how many genes are differentially expressed in our results using DESeq2's `summary()` function.

```
summary(wt_res)
```

```
##
## out of 29866 with nonzero total read count
## adjusted p-value < 0.05
## LFC > 0 (up)       : 3376, 11%
## LFC < 0 (down)     : 3071, 10%
## outliers [1]       : 46, 0.15%
## low counts [2]     : 7851, 26%
## (mean count < 1)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

Annotation to gene IDs in DESeq2 results

The `annotables()` library package contain gene information that can be used to annotate DESeq2 DE results.

```
# Load annotables
library(annotables)

# View the information of mouse genes in the grcm38 dataset
grcm38
```

```
## # A tibble: 53,728 x 9
##    ensgene   entrez symbol chr     start    end strand biotype description
##    <chr>      <int> <chr>  <chr>   <int>  <int>  <int> <chr>   <chr>
##  1 ENSMUSG0~  14679 Gnai3  3      1.08e8 1.08e8     -1 protei~ guanine nucleoti~
##  2 ENSMUSG0~  54192 Pbsn   X      7.78e7 7.79e7     -1 protei~ probasin [Source~
##  3 ENSMUSG0~  12544 Cdc45  16     1.88e7 1.88e7     -1 protei~ cell division cy~
##  4 ENSMUSG0~     NA H19    7      1.43e8 1.43e8     -1 lincRNA H19, imprinted m~
##  5 ENSMUSG0~ 107815 Scml2  X      1.61e8 1.61e8      1 protei~ sex comb on midl~
##  6 ENSMUSG0~  11818 Apoh   11     1.08e8 1.08e8      1 protei~ apolipoprotein H~
##  7 ENSMUSG0~  67608 Narf   11     1.21e8 1.21e8      1 protei~ nuclear prelamin~
##  8 ENSMUSG0~  12390 Cav2   6      1.73e7 1.73e7      1 protei~ caveolin 2 [Sour~
##  9 ENSMUSG0~  23849 Klf6   13     5.86e6 5.87e6      1 protei~ Kruppel-like fac~
## 10 ENSMUSG0~  29871 Scmh1  4      1.20e8 1.21e8      1 protei~ sex comb on midl~
## # ... with 53,718 more rows
```

Convert the DESeq result object to a data frame using the `data.frame()` function and move rownames (Esembl gene IDs) to a column with a variable name “ensgene”. Use the `left_join()` function from the `dplyr` package to join the gene information from `grcm38`.

```
wt_res_all <- data.frame(wt_res) %>% rownames_to_column(var = "ensgene")
wt_res_all <- left_join(x = wt_res_all,
                        y = grcm38[, c("ensgene", "entrez", "symbol", "description")],
                        by = "ensgene")
```

Extract Significant DE Genes using subset() and arrange() functions

The `dplyr` library package has the `filter()` and `arrange()` functions that can be used to filter the significant DE genes.

```
wt_res_sig <- wt_res_all %>% filter(padj < 0.05) %>% arrange(padj)
write.csv(wt_res_sig, file = "Fibrosis_vs_normal_wt_DESeq2.csv", row.names = FALSE)
head(wt_res_sig)
```

```
##              ensgene baseMean log2FoldChange     lfcSE     stat        pvalue
## 1 ENSMUSG00000053113 1318.172       4.875042 0.1602150 28.35017 8.330337e-177
## 2 ENSMUSG00000005087 2943.740       6.121134 0.2072198 27.89891 2.750335e-171
## 3 ENSMUSG00000036887 3899.514       3.866162 0.1274025 27.83466 1.652178e-170
## 4 ENSMUSG00000026822 8870.171       6.466148 0.2378236 25.82294 4.901141e-147
## 5 ENSMUSG00000036905 3237.605       3.835279 0.1377392 25.52164 1.133940e-143
## 6 ENSMUSG00000027962 9298.598       5.781446 0.2194960 24.88019 1.219170e-136
##            padj entrez symbol
## 1 1.830092e-172  12702  Socs3
## 2 3.021106e-167  12505   Cd44
## 3 1.209890e-166  12259   C1qa
## 4 2.691829e-143  16819   Lcn2
## 5 4.982305e-140  12260   C1qb
## 6 4.463989e-133  22329  Vcam1
##                                                                                 description
## 1                     suppressor of cytokine signaling 3 [Source:MGI Symbol;Acc:MGI:1201791]
## 2                                          CD44 antigen [Source:MGI Symbol;Acc:MGI:88338]
## 3 complement component 1, q subcomponent, alpha polypeptide [Source:MGI Symbol;Acc:MGI:88223]
## 4                                           lipocalin 2 [Source:MGI Symbol;Acc:MGI:96757]
## 5  complement component 1, q subcomponent, beta polypeptide [Source:MGI Symbol;Acc:MGI:88224]
## 6                         vascular cell adhesion molecule 1 [Source:MGI Symbol;Acc:MGI:98926]
```

DESeq2 visualizations - MA and volcano plots

To explore the results, visualizations can be helpful to see a global view of the data, as well as, characteristics of the significant genes. Usually, we expect to see significant genes identified across the range of mean values against log fold changes, which we can plot using the **MA plot**. If we only see significant genes with high mean values, then it could indicate an issue with our data. The **volcano plot** helps us to get an idea of the range of fold changes needed to identify significance in our data.

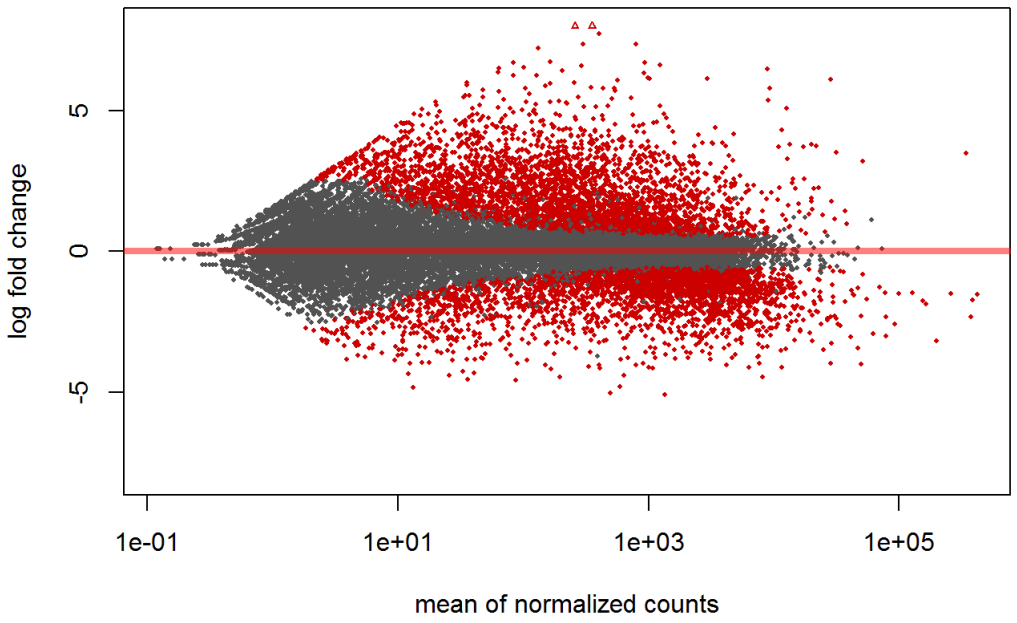
Let’s explore our results using `MA plots` and `volcano plots`.

Create an MA plot using the `plotMA()` function and using the results object as input.

Create a new column as a logical vector regarding whether `padj` values are less than `0.05` for the results using the `mutate()` function.

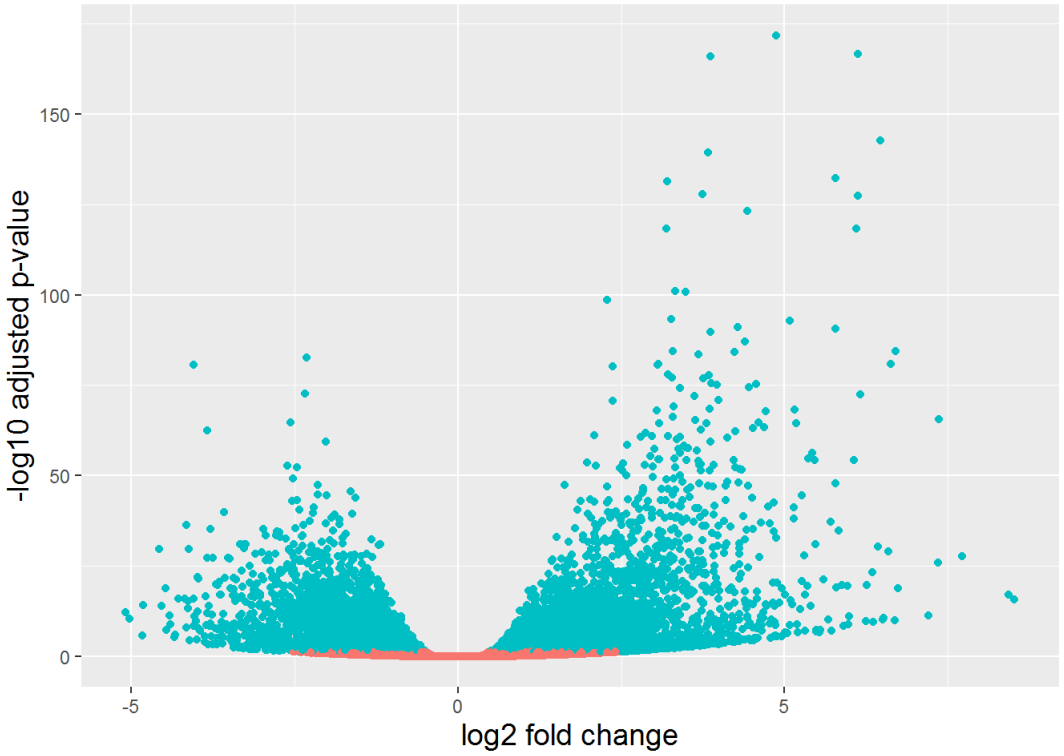
Create a `volcano plot` of the `log2 foldchange values` versus the `-log10 adjusted p-value` using `ggplot()` and coloring the points for the genes by whether or not they are significant.

```
# Create MA plot
plotMA(wt_res, ylim=c(-8,8))
```




```
# Generate logical column
wt_res_all <- data.frame(wt_res_all) %>% mutate(threshold = padj < 0.05)

# Create the volcano plot
ggplot(wt_res_all) +
        geom_point(aes(x = log2FoldChange, y = -log10(padj), color = threshold)) +
        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)))
```



DESeq2 visualizations - heatmap

Heatmap visualizations can also be helpful in exploring the significant genes by overall looking at how different the expression of all significant genes are between sample groups.

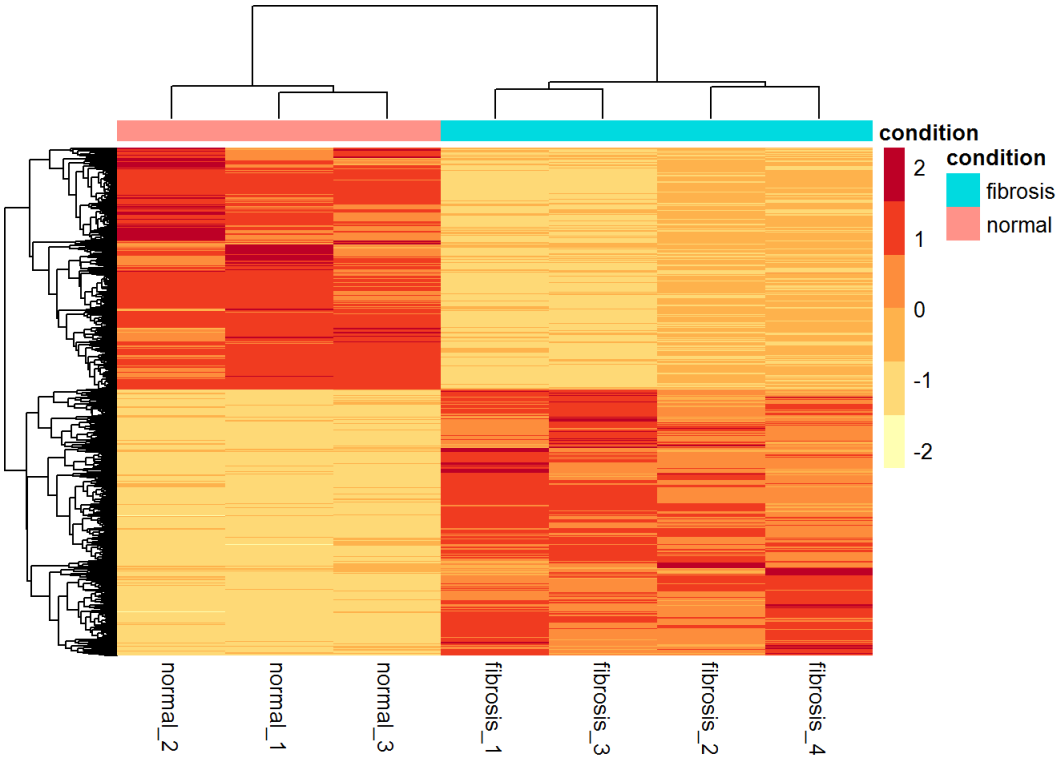
Subset `the normalized counts` to only include the significant genes. Use the row names of the significant normalized count results to subset the normalized counts.

Create the heatmap using significant normalized counts. Color the heatmap using the `palette` , `heat_colors` , cluster the rows without showing row names, and scale the values by “row”. For the annotation, use `select()` to select only the `condition` column from the `metadata` .

```
# Subset normalized counts to significant genes
sig_norm_counts_wt <- wt_dge_normalized[wt_res_sig$ensgene, ]

# Choose heatmap color palette
heat_colors <- brewer.pal(n = 6, name = "YlOrRd")

# Plot heatmap
pheatmap(sig_norm_counts_wt,
         color = heat_colors,
         cluster_rows = T,
         show_rownames = F,
         annotation = select(wt_metadata, condition),
         scale = "row")
```


condition
condition
fibrosis
normal
2
1
0
-1
-2
normal_2
normal_1
normal_3
fibrosis_1
fibrosis_3
fibrosis_2
fibrosis_4