Differential Expression analysis Example

Adam France

2022-03-04

What is Differential Expression analysis?

Lets say you have two phenotypes: a healthy, or wildtype, and non-healthy, or disease type. You want to know the difference in these two groups based on what genes are being expressed. Since we can't directly measure this in vivo, we take samples and measure the amount of messenger RNA that corresponds to that organisms known genes. By knowing which genes are being over, or under expressed in a disease state, we can better understand a disease and design therapies for specific target(S). Specifically for this example, I am using a kidney-fibrosis dataset, where samples where taken from the kidneys in mice from a healthy and disease(fibrosis) group.

The code was adapted from a course on Data Camp called RNA-Seq with Bioconductor in r.

The raw data file can be found here

Documentation for DESEq2, the main package used in this exercise.

r version used in this exercise:

```
R.version.string
```

```
## [1] "R version 4.1.2 (2021-11-01)"
```

The libraries needed for this exercise:

```
library(dplyr)
library(DESeq2)
library(RColorBrewer)
library(pheatmap)
library(tidyverse)
library(forcats)
```

Bioconductor is also needed to install some of these packages. once installed, you can call `Biocmanager::install()` with the package name in parentheses.

Load in the raw counts matrix

It's important to note that all of the pre-analysis steps have already been performed. That is, the filtering of bad reads and removal of adapter sequences from the raw data, typically, a FASTQ file. From the QC'd data, the reads are mapped to a reference genome, and from the location on that genome a read can be assigned to a particular gene.

```
smoc2_rawcounts <- read.csv("fibrosis-raw.txt")
```

Lets take a look at the data:

```
head(smoc2_rawcounts, 5)
```

```
##                   X smoc2_fibrosis1 smoc2_fibrosis4 smoc2_normal1
## 1 ENSMUSG00000102693               0               0             0
## 2 ENSMUSG00000064842               0               0             0
## 3 ENSMUSG00000051951              72              30             0
## 4 ENSMUSG00000102851               0               0             0
## 5 ENSMUSG00000103377               0               0             1
##   smoc2_normal3 smoc2_fibrosis3 smoc2_normal4 smoc2_fibrosis2
## 1             0               0             0               0
## 2             0               0             0               0
## 3             3              36             1              51
## 4             0               0             0               0
## 5             0               0             0               0
```

Okay, everything looks right. We have column (X) which list the names of genes being measured. the other columns are the names of the sample groups, with the raw counts for the corresponding gene in column x. The count is how many times a particular read was mapped to a location in the genome that was in the 'territory' of that gene.

Since no metadata file was included, we need to create the meta data table (a 7x3 Dataframe)

DESeq needs to know the exact structure of the sample groups to match with the raw counts matrix.

```
genotype <- c(replicate(7,"smoc2_oe"))
condition <- c(replicate(4, "fibrosis"))
condition <- append(condition, (replicate(3, "normal")))

samples <- c("smoc2_fibrosis1","smoc2_fibrosis2","smoc2_fibrosis3",
             "smoc2_fibrosis4","smoc2_normal1","smoc2_normal3",
             "smoc2_normal4")

smoc2_metadata <- data.frame( genotype, condition )
rownames(smoc2_metadata) <- samples
```

Lets check to make sure eveything is lined up:

```
smoc2_metadata
```

```
##                 genotype condition
## smoc2_fibrosis1 smoc2_oe  fibrosis
## smoc2_fibrosis2 smoc2_oe  fibrosis
## smoc2_fibrosis3 smoc2_oe  fibrosis
## smoc2_fibrosis4 smoc2_oe  fibrosis
## smoc2_normal1   smoc2_oe    normal
## smoc2_normal3   smoc2_oe    normal
## smoc2_normal4   smoc2_oe    normal
```

Use the match() function to reorder the columns of the raw counts

```
reorder_idx <- match(rownames(smoc2_metadata), colnames(smoc2_rawcounts))
```

Reorder the columns of the count data

```
reordered_smoc2_rawcounts <- smoc2_rawcounts[,reorder_idx] #all rows from the reorder_idx column

head(reordered_smoc2_rawcounts, 5)
```

```
##   smoc2_fibrosis1 smoc2_fibrosis2 smoc2_fibrosis3 smoc2_fibrosis4 smoc2_normal1
## 1               0               0               0               0             0
## 2               0               0               0               0             0
## 3              72              51              36              30             0
## 4               0               0               0               0             0
## 5               0               0               0               0             1
##   smoc2_normal3 smoc2_normal4
## 1             0             0
## 2             0             0
## 3             3             1
## 4             0             0
## 5             0             0
```

We can now see the order of columns has changed to match the meta data file we created.

Create the DESEq Object:

```
dds_smoc2 <- DESeqDataSetFromMatrix(countData = reordered_smoc2_rawcounts,
                                    colData = smoc2_metadata,
                                    design = ~ condition)
```

Normalization:

Before we move forward, our reads need to be normalized. sample 'A' might have five times the counts for many genes as sample 'B', but many reasons other than biology can cause this, and will make our visualizations impossible to interpret. normalization puts the data on a 'scale' instead of absolute values that makes useful comparisons between the sample groups possible.

Determine the size factors to use for normalization:

```
dds_smoc2 <- estimateSizeFactors(dds_smoc2)
```

Extract the normalized counts

```
smoc2_normalized_counts <- counts(dds_smoc2, normalized = TRUE) %>% data.frame()
#there are problems down the line if you dont format this as a data frame
```

Heirarchical HeatMap

Transform the normalized counts

```
vsd_smoc2 <- vst(dds_smoc2, blind = TRUE)
```

Extract the matrix of transformed counts

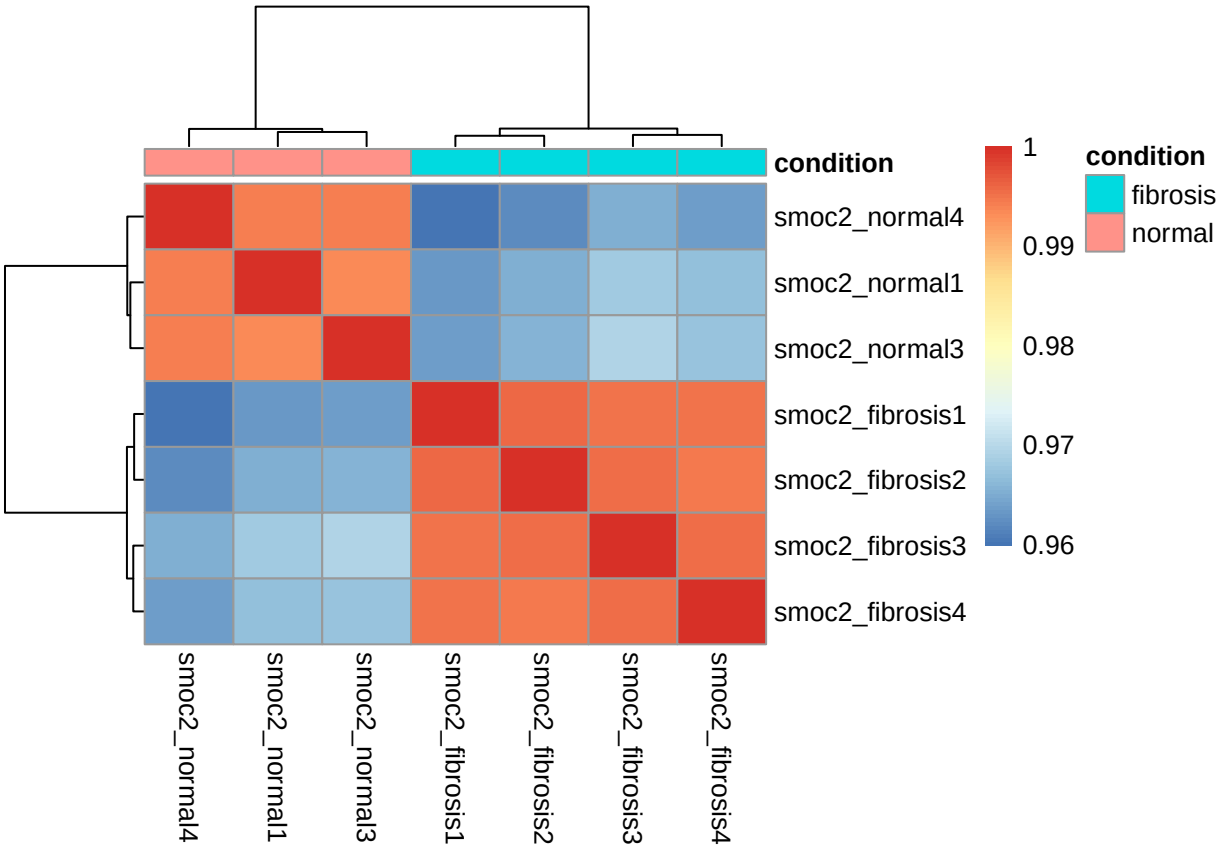
```
vsd_mat_smoc2 <- assay(vsd_smoc2)
```

Compute the correlation values between samples

```
vsd_cor_smoc2 <- cor(vsd_mat_smoc2)
```

Plot the heatmap

```
pheatmap(vsd_cor_smoc2, annotation = select(smoc2_metadata, condition))
```

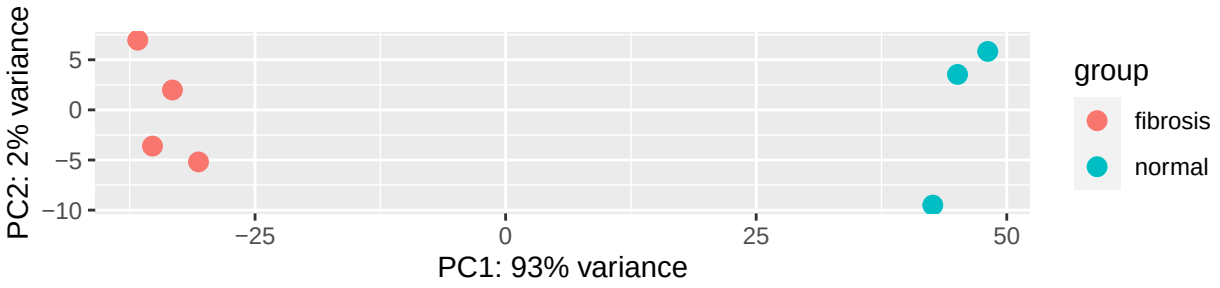


This visualization is important for understanding the grouping of our samples groups. the heirarchical heatmap clusters sample groups together based on overall similarity. so it would make sense for our wildtypes to be one side, and our disease types on the other, which is what we see here. We would know something was wrong if there was either no clear grouping, or if one member of either group was with the others. This helps us validate, biologically, that the differences we are seeing are due to the disease.

PCA PLot

Plot the PCA of PC1 and PC2

```
plotPCA(vsd_smoc2, intgroup='condition')
```



Here we can see that the two groups are clustered on each side of the x (PC1) axis. PC1 is the axis of greatest variance in the data, so for our two groups to be clustered on opposite sides is a good sign that the difference in the expression data is due to fibrosis and not chance.

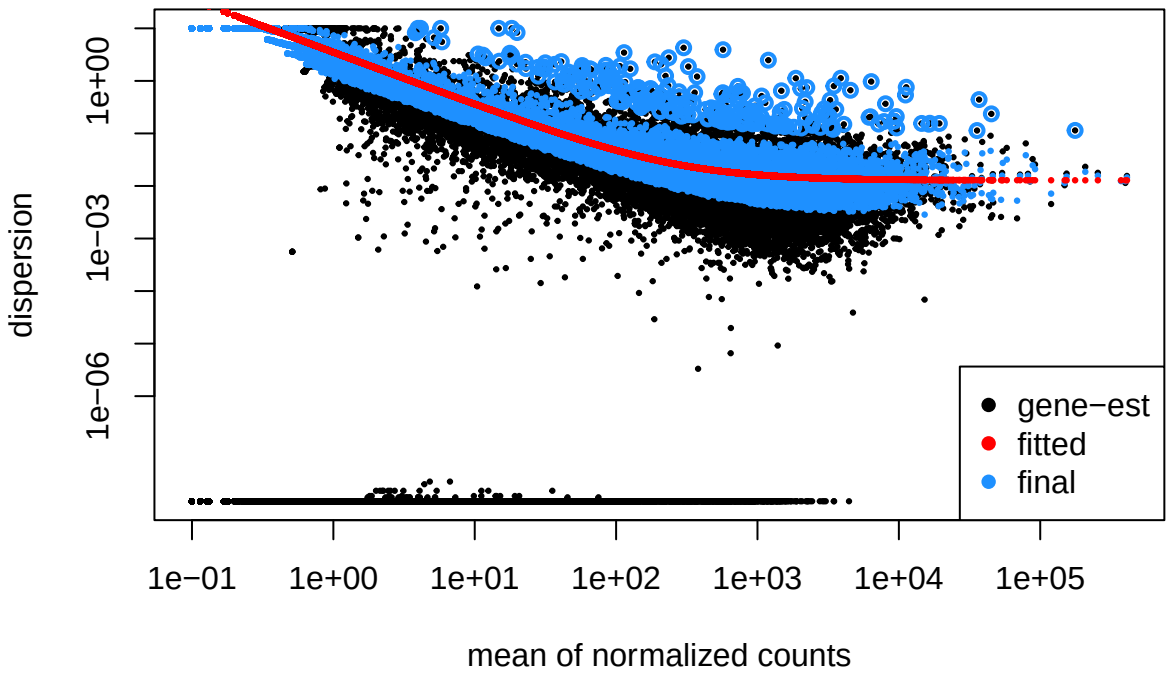
Run DESeq2

Run the DESeq2 analysis

```
analysis_object <- DESeq(dds_smoc2)
```

Plot dispersions

```
plotDispEsts(analysis_object)
```



Extract the results of the differential expression analysis

```
smoc2_res <- results(analysis_object,
                     contrast = c("condition", "fibrosis", "normal"),
                     alpha = 0.05)
```

Shrink the log2 fold change estimates to be more accurate

```
smoc2_res <- lfcShrink(analysis_object,
                       contrast =  c("condition","fibrosis" ,"normal" ),
                       res = smoc2_res, type="ashr") # 'ashr' is a package that may need to be installe

#Extract results
smoc2_res <- results(analysis_object,
                     contrast = c("condition","fibrosis","normal"),
                     alpha = 0.05,
                     lfcThreshold = 0.32)

#Shrink the log2 fold changes
smoc2_res <- lfcShrink(analysis_object,
                       contrast = c("condition","fibrosis","normal"),
                       res = smoc2_res, type='ashr') # 'ashr' is a package that may need to be installe
```

Reviewing the results

```
summary(smoc2_res)
```

```
##
## out of 29556 with nonzero total read count
## adjusted p-value < 0.05
## LFC > 0 (up)       : 3716, 13%
## LFC < 0 (down)     : 3322, 11%
## outliers [1]       : 15, 0.051%
## low counts [2]     : 7207, 24%
## (mean count < 1)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

if we look at the 'LFC' (log fold change) rows we have about 12% change for both up and down of our non-zero genes. These are sensible numbers given that we are comparing two groups of the same tissue type from the same organism. we started with about 47,000 genes, so roughly 7,000 genes is a good indicator that something is going on. if we were getting much higher percentages we would know something was wrong either in the analysis or in library preparation.

Save results as a data frame

```
smoc2_res_all <- data.frame(smoc2_res)
```

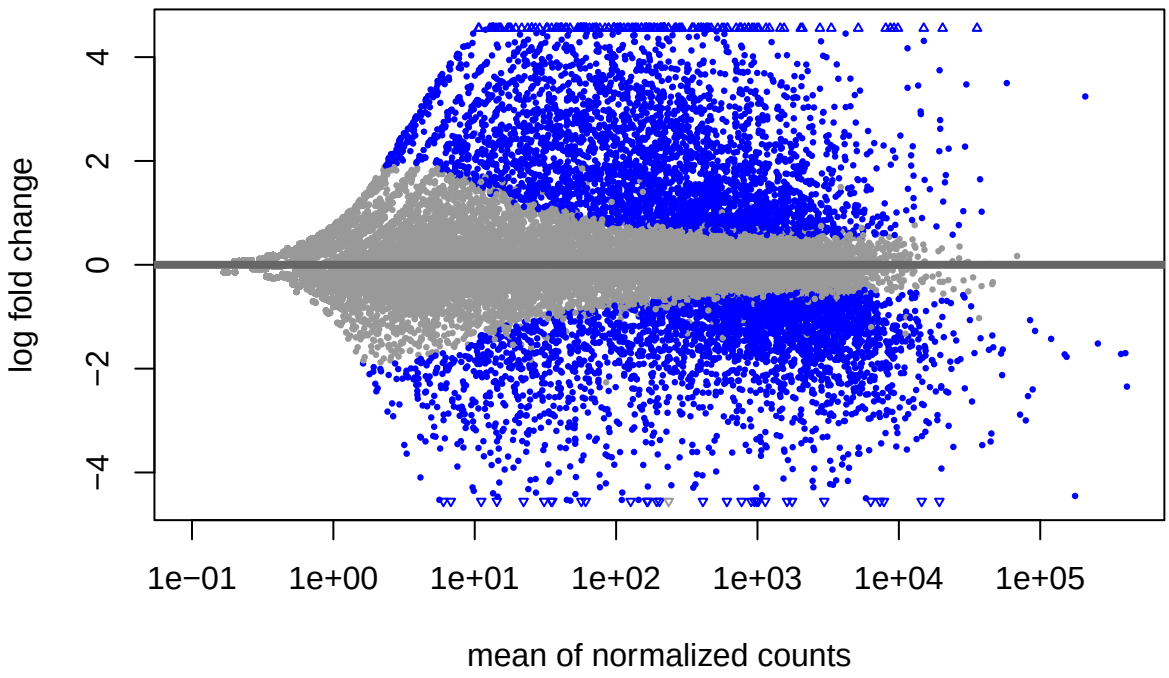
Subset the results to only return the significant genes with p-adjusted values less than 0.05

```
smoc2_res_sig <- subset(smoc2_res_all, padj < 0.05)
```

Visualizing results:

Create MA plot

```
plotMA(smoc2_res)
```

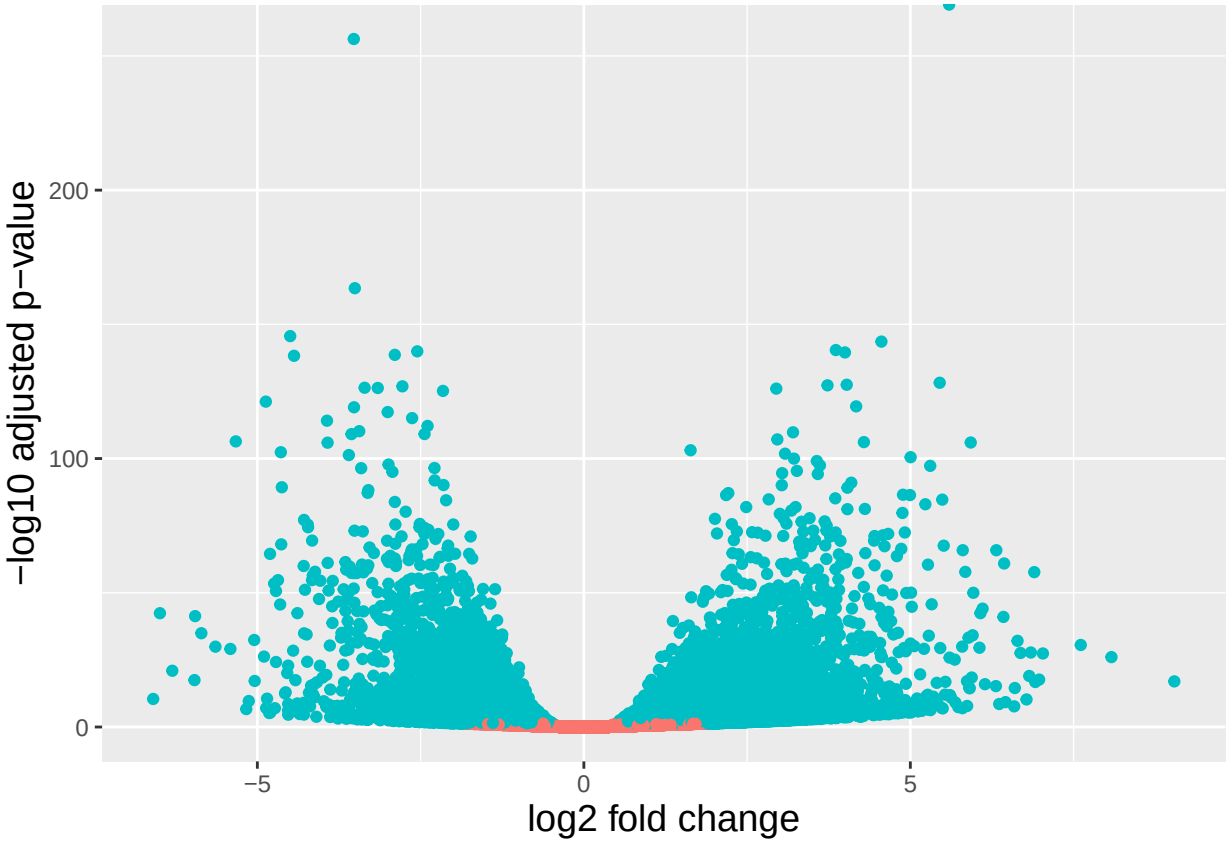


Generate logical column

```
smoc2_res_all <- data.frame(smoc2_res) %>% mutate(threshold = padj < 0.05)
```

Volcano plot

```
ggplot(smoc2_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)))
```



In the volcano plot we are looking at the amount of change in expression vs the p-value for the gene. the x-axis is the fold-change, so points to the right of zero are 1x, 2x more, and points to left -1x, -2x etc. the y-axis represents statistical significance, where up means higher significance (less likely due to chance) and down means less significance.

Heat Map

Subset normalized counts to significant genes

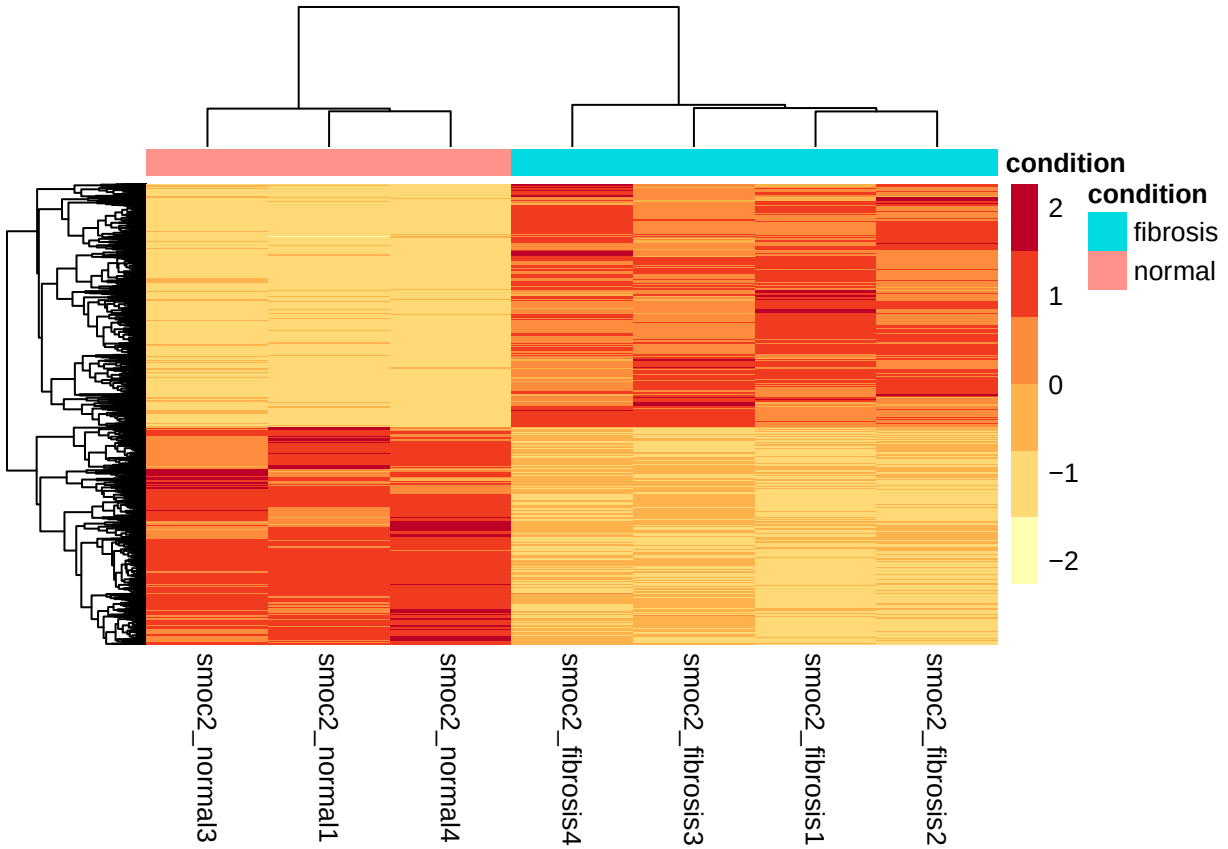
```
sig_norm_counts_smoc2 <- smoc2_normalized_counts[rownames(smoc2_res_sig),]
```

Choose heatmap color palette

```
heat_colors <- brewer.pal(n = 6, name = "YlOrRd")
```

Plot the Heatmap

```
pheatmap(sig_norm_counts_smoc2,
         color = heat_colors,
         cluster_rows = T,
         show_rownames = F,
         annotation = select(smoc2_metadata, condition),
         scale = "row")
```



just like our first heatmap, here we see some good separation in terms of color and the sample groups, the main difference is that here each individual gene is being grouped heirarchically(the rows) against the sample groups. So before where we had a symmetrical 7X7 matrix, now we have a 7038 X 7 matrix. Again though we see that the grouping is what we expect, just with individual genes and whether their expression is higher (red) or lower (yellow).

for generating a list of genes of interest

you can run this list against a database to get an idea as to what these genes do.

```
smoc2_genes_sig <- smoc2_rawcounts[,1][(as.integer(rownames(smoc2_res_sig)))]

length(smoc2_genes_sig) #how many genes there are in this list
```

```
## [1] 7038
```

```
head(smoc2_genes_sig) #the first few in the list
```

```
## [1] "ENSMUSG00000051951" "ENSMUSG00000025900" "ENSMUSG00000033845"
## [4] "ENSMUSG00000025903" "ENSMUSG00000002459" "ENSMUSG00000033793"
```

What to do next:

So we ended up with several thousand potential genes that are involved in fibrosis, and we are analysis showed that it wasn't due to chance. But what do you do now? if you are suspecting a certain gene, or

group of genes you can just look them up in the list we created and that might be the end of it. But maybe you are part of a much larger project to understand every gene, or to design a therapy to target a specific gene or a protein that gene makes. 7038 is a lot of genes to look at, way too many to do one by one in any reasonable amount of time. This is where gene ontology comes in handy. GOs (gene ontologies) are keyword phrases that are biologically relevant to their assigned gene. there are public databases that house this information and can be used to transform a list of thousands of genes into a small list of biological areas actually worth further investigation. You might discover that 45% of the significantly DE genes are involved in ion transport, or maybe cytoskeleton regulation, these are the kind of insights that can help research move more efficiently.