RNAseq Analysis Report

Noa Henig

November 20, 2018

Overview

This report includes the R script that analyzes RNA-seq samples of normal and Psoriasis lesions from GSE54456.

Loading the required R packages

```
library("edgeR")
```

```
## Loading required package: limma
```

```
suppressMessages(library("NMF"))
library("ggrepel")
```

```
## Loading required package: ggplot2
```

```
library("ggplot2")
library("ggfortify")
library("plyr")
```

Reading the counts and sample annotation tables

```
counts <- read.table("counts.txt",header=TRUE, row.names=1)
sampleAnnots <- read.table("sample-annotation.txt",row.names=1,header=TRUE)
geneAnnots <- read.delim("gene-annotation.txt", header=TRUE)
```

check consistency of the input files and report problems if exist

```
# get the sample names from the counts table and the sample annottaion table
samplesFromCounts <- colnames(counts)
samplesFromAnnots <- rownames(sampleAnnots)
genesFromCounts <- rownames(counts)
inconsistency = FALSE

# check if any name appears in only one of the files and not the other
differenceAc <- setdiff(samplesFromAnnots, samplesFromCounts)
differenceCa <- setdiff(samplesFromCounts, samplesFromAnnots)

# no differences, do nothing
if (length(differenceAc) + length(differenceCa) == 0)
    print ("Sample names in input files match.")
```

```
## [1] "Sample names in input files match."
```

```
if (length(differenceAc) > 0) ({ #there is a difference, print informative message
    print("The following sample(s) are missing from the counts file, fix it and run again:")
    print (differenceAc)
    inconsistency = TRUE
})

#there is a difference, print an informative message:
if (length(differenceCa) > 0) ({
    print("The following sample(s) are missing from the sample annotation file, fix it and run again:")
    print (differenceCa)
    inconsistency = TRUE
})
if (inconsistency == TRUE)
    quit(status=1)
```

Generate Count Per Million matrix

```
# generate an edgeR DGEList object that has the sample counts and anotations in it
group <- factor(c(sampleAnnots[,1]))
dgeList <- DGEList(counts=counts,group=group)

# cpm function from edgeR, prior.count is a starting value used to offset to prevent zero counts
cpmMatrix <- cpm(dgeList, normalized.lib.sizes=FALSE, log=FALSE, prior.count=0.25)
```

Filter the cpm matrix

```
numberOfSamples = ncol(counts)
numberOfGenes <- nrow(counts)
# for plotting of cpm per gene later
dgeListCpm <- DGEList(counts=cpmMatrix,group=group)
# run through the genes (rows), keep lines where at least 0.75 of the samples had at least 1 count.
threshold <- 0.75 * numberOfSamples
j <- 0
keep <- 0
newMatrix<-0
for (i in 1:numberOfGenes) ({
    if (sum(counts[i,] >= 1) >= threshold) ({
        j=j+1
        keep <- c(keep,i)
    })
})
newMatrix<-counts[keep,]
print(paste("Got the low expression genes filtered out, number of genes kept:", length(keep)))
```

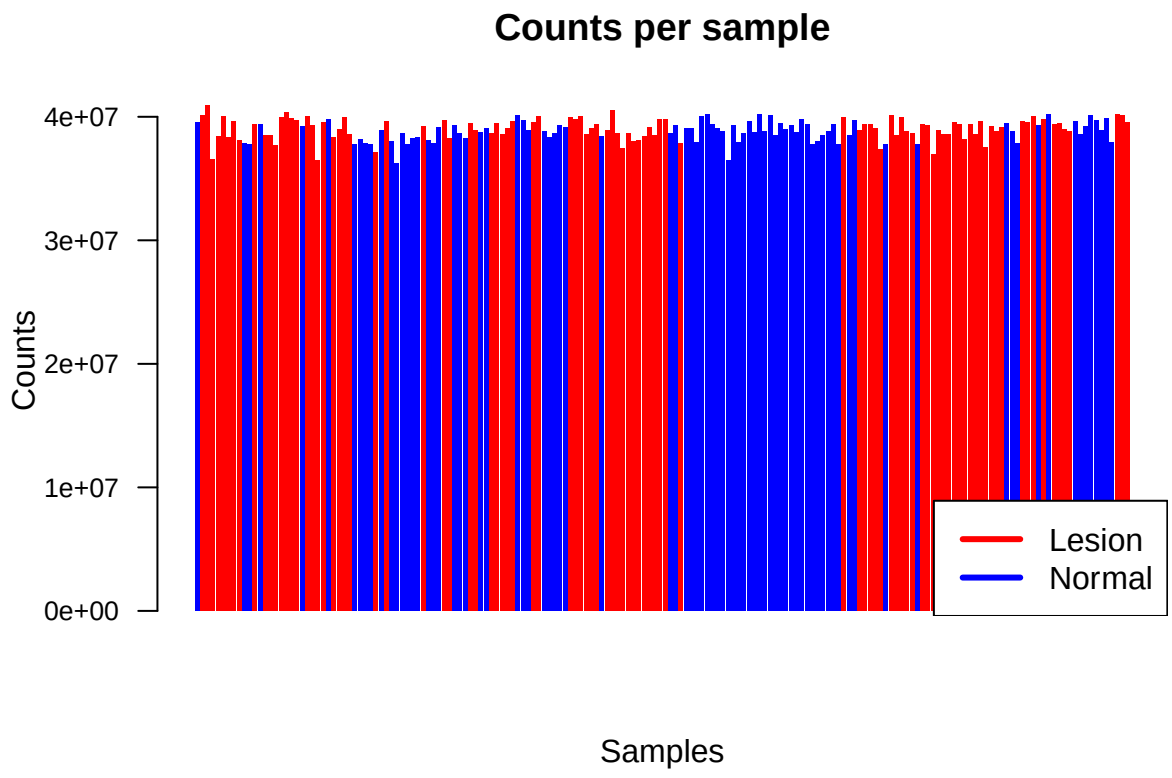
```
## [1] "Got the low expression genes filtered out, number of genes kept: 29928"
```

Make a logCPM filtered matrix and save as a binary file

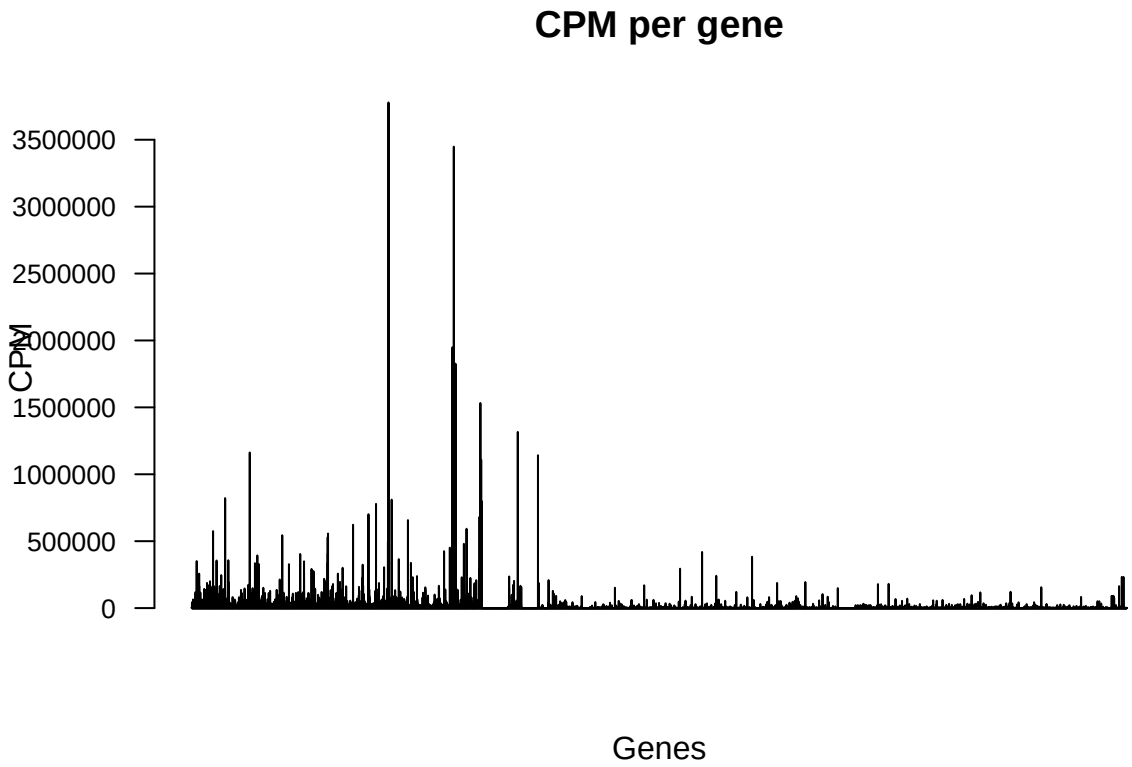
```
filteredCounts <- newMatrix
## generate a dgeList object from the filtered matrix, use logCPM this time and
filteredDgeList <- DGEList(counts=filteredCounts,group=group)
# cpm function from edgeR, prior.count is a starting value used to offset to prevent zero counts
filteredLogCpmMatrix <- cpm(filteredDgeList, normalized.lib.sizes=FALSE, log=TRUE, prior.count=0.25)
#save this into an RDS binary format:
saveRDS(filteredLogCpmMatrix, "filteredLogCpmMatrix.rds")
```

PCA and other plots, exclusion of outliers

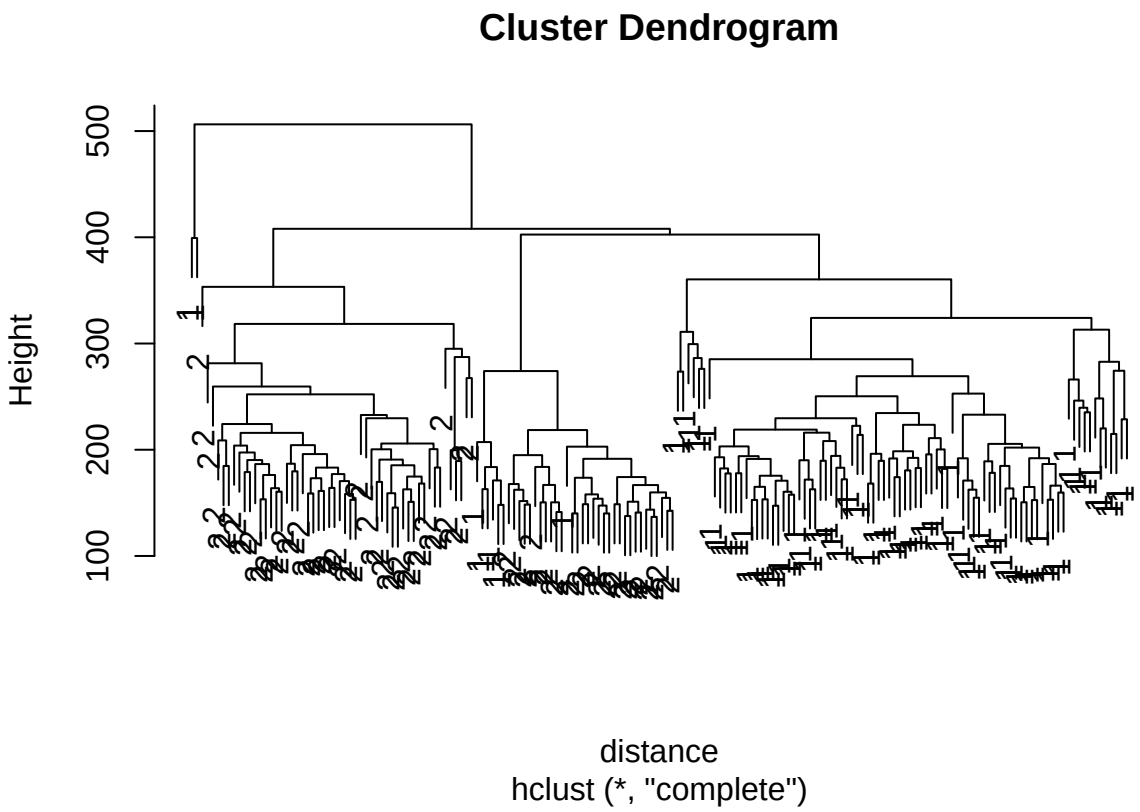
```
colors <- c("red", "blue") # red/"1"s are Lesion samples, blue/"2"s are Normal

# plot the raw counts per sample, color represents group (normal/lesion) to check for possible bias in
densities=rep("-1", length(samplesFromCounts))
barplot(colSums(dgeList$counts), las=2, main="Counts per sample", cex.axis=0.8,
        cex.names=0.5, col=colors[group], axisnames=FALSE, density=densities,
        border = NA, xlab="Samples", ylab="Counts")
legend("bottomright", c("Lesion", "Normal"), col=c("red", "blue"), lwd=3, bg="white")
```



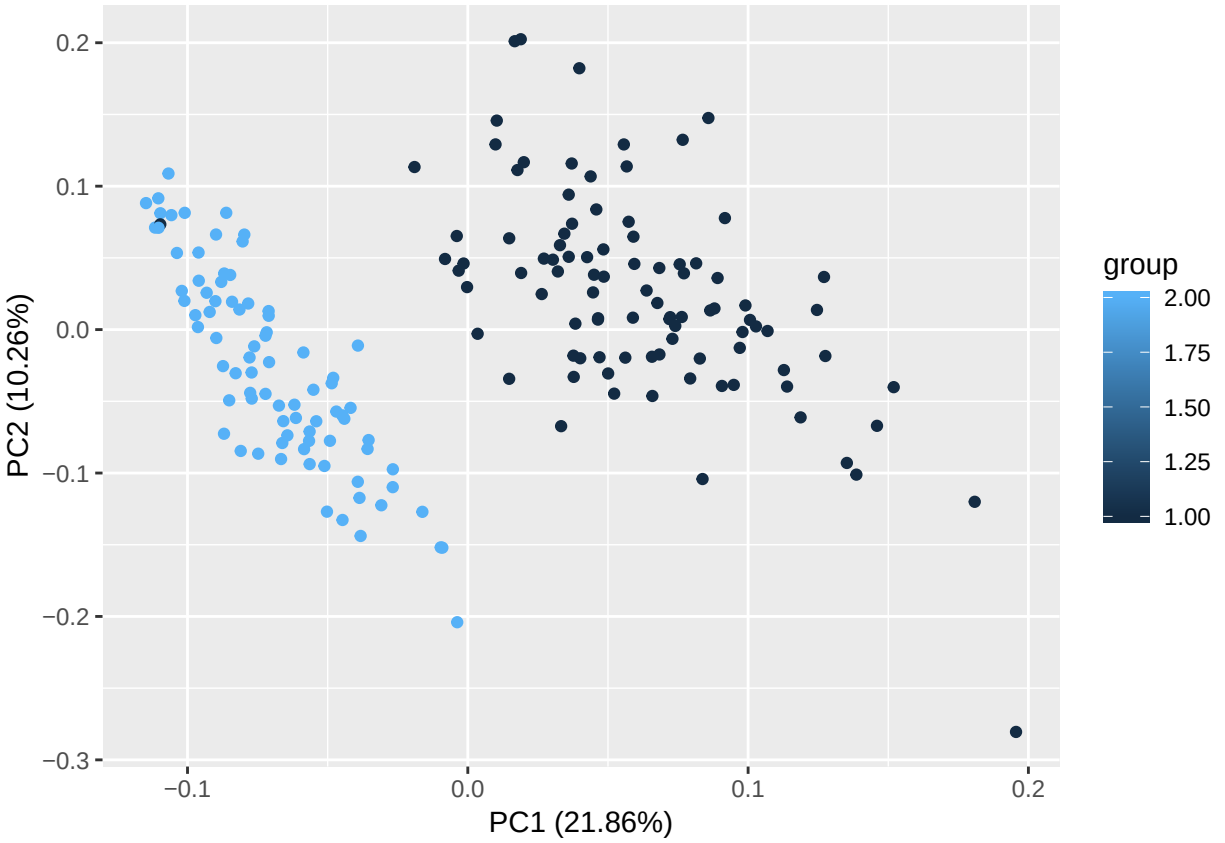
```r
# plot the gene expression levels per gene based on the cpm values
barplot(rowSums(dgeListCpm$counts), las=2, main="CPM per gene", axisnames=FALSE, cex.axis=0.8,
        ylab="CPM", xlab="Genes")
```



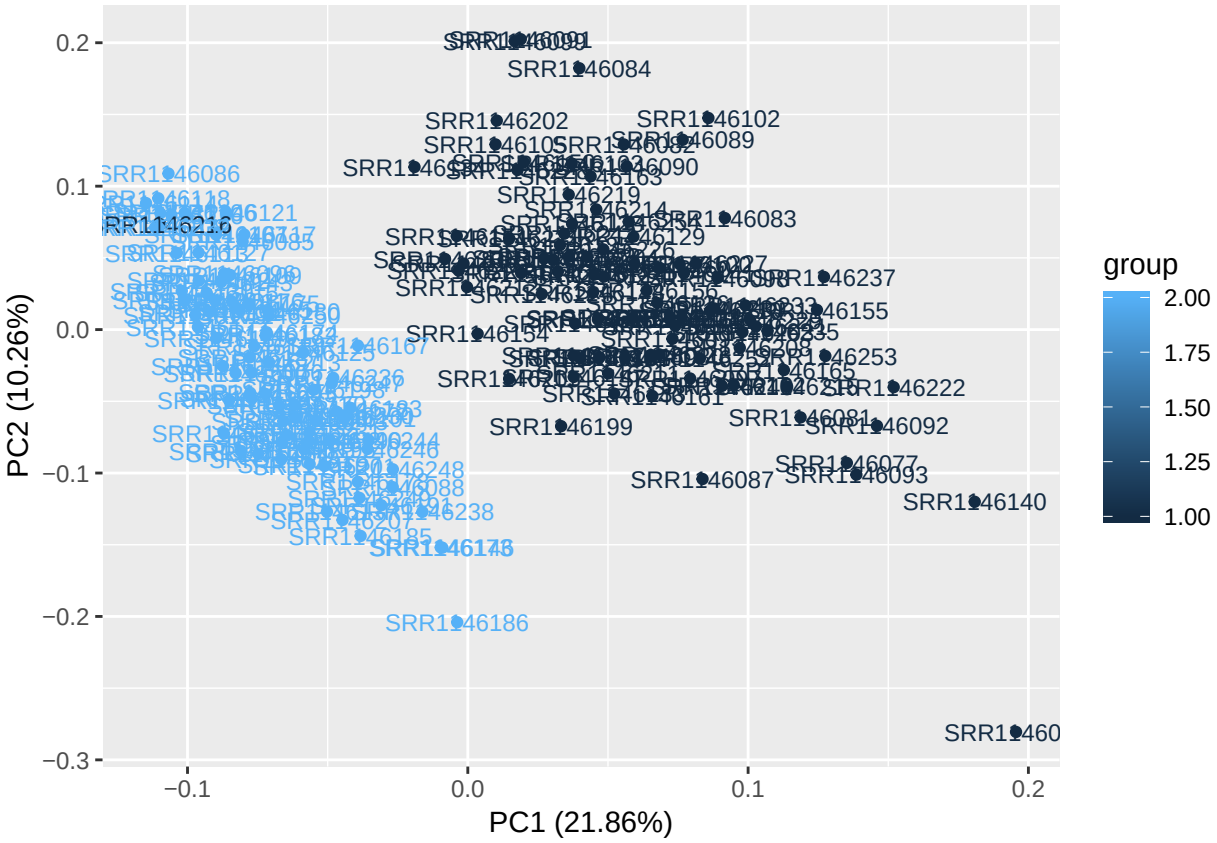

```r
# dendrogram that shows hierarchical clustering analysis, reflects the distance between samples
#based on logCPM data.
# transpose the filtered counnts matrix so each line describes an element on the plot and the
#columns are the variables
transposedFLCPM <- t(filteredLogCpmMatrix)
distance <- dist(transposedFLCPM)
clusters <- hclust(distance)
plot(clusters, labels=group)
```




```
# PCA and detection of outliers:
# add the column with group information
transposedFLCPMwGroup <- cbind(transposedFLCPM, group)
autoplot(prcomp(transposedFLCPM), data=transposedFLCPMwGroup, colour='group')
```




```
# add sample labels to identify the outlier lesion sample on the bottom right and one that
# seems mislabeled as lesion
autoplot(prcomp(transposedFLCPM), data=transposedFLCPMwGroup, colour='group', label=TRUE,
         label.size=3)
```




```
# remove the outlier and the suspected mislabeled sample:
outlier <- "SRR1146078"
mislabeled <- "SRR1146216"
samplesToRemove <- c(outlier, mislabeled)
# vector of the samples we want to keep
samplesToKeep <- setdiff(samplesFromAnnots, samplesToRemove)
# subset of the samples and their annotations that we want to keep
sampleAnnotsToKeep <- as.matrix(sampleAnnots[samplesToKeep,])
# define a new 'group' factor that does not include the 2 samples that are removed
group <- factor(c(sampleAnnotsToKeep[,1]))
#generate new objects for the downstream analysis
filteredCountsNoOutliers <- filteredCounts[,samplesToKeep]
## generate a dgeList object from the new matrix, use logCPM this time and
filteredDgeListNoOutliers <- DGEList(counts=filteredCountsNoOutliers,group=group)
# get logCPM
filteredLogCpmNoOuts <- cpm(filteredDgeListNoOutliers, normalized.lib.sizes=FALSE,
                            log=TRUE, prior.count=0.25)
```

Differential expression analysis by edgeR

```
# Normalization is not needed for single factor analysis.
# The dispersion is estimated before testing for differential expression, using the filtered
# counts. edgeR uses quantile-adjusted conditional maximum likelihood (qCML) method for
# experiments with a single factor. The design matrix defines the variables and levels
```

```
# considered in the model, in this case one factor:
design <- model.matrix(~group)
filteredDgeListNoOutliers <- estimateDisp(filteredDgeListNoOutliers, design)

# exact test for the negative binomial distribution to compute exact p-values to assess
# differential expression between the normal and lesion samples.
# The function exactTest computes exact p-values by summing over all sums of counts that
# have a probability less than the probability under the null hypothesis of the observed sum of counts.
et <- exactTest(filteredDgeListNoOutliers)
topTags(et)
```

```
## Comparison of groups:  normal-lesional
##                      logFC    logCPM PValue FDR
## ENSG00000241794 -7.097882 10.042093      0   0
## ENSG00000196805 -6.123587  9.046407      0   0
## ENSG00000198074 -6.098107  6.123885      0   0
## ENSG00000136688 -5.520329  6.912500      0   0
## ENSG00000227471 -5.009572  2.841350      0   0
## ENSG00000159337 -4.445683  7.456132      0   0
## ENSG00000115919 -4.352201  4.560336      0   0
## ENSG00000165474 -4.247575 10.039662      0   0
## ENSG00000213201 -3.579262  2.605319      0   0
## ENSG00000164687 -3.488666  8.262420      0   0
```

```
# modifying the row names of the differentially expressed genes table to be able to merge
# with the gene annotations
tempDE <- et$table
ENSEMBL <- rownames(tempDE)
rownames(tempDE) <- NULL
tempDEWithNames <- cbind(ENSEMBL, tempDE)

# that merge removes genes that do not have corresponding annotation, will use that later
annotatedDE_onlyAnnotated <- merge(tempDEWithNames, geneAnnots, by="ENSEMBL")

# the following 'join' is also merging the two table but genes that don't have annotaions
# are included in the output with 'NA's in the annotation columns; write this table to file.
annotatedDE_allGenes <- join(tempDEWithNames, geneAnnots, type="left", by="ENSEMBL")

# sort the list of DEGs by ascending p-values and write to a tab-delimited text file
sortedDE <- annotatedDE_allGenes[order(annotatedDE_allGenes$PValue),]
write.table(sortedDE, "differntialExpression.txt", sep="\t")
```

Heatmap of the top DEGs

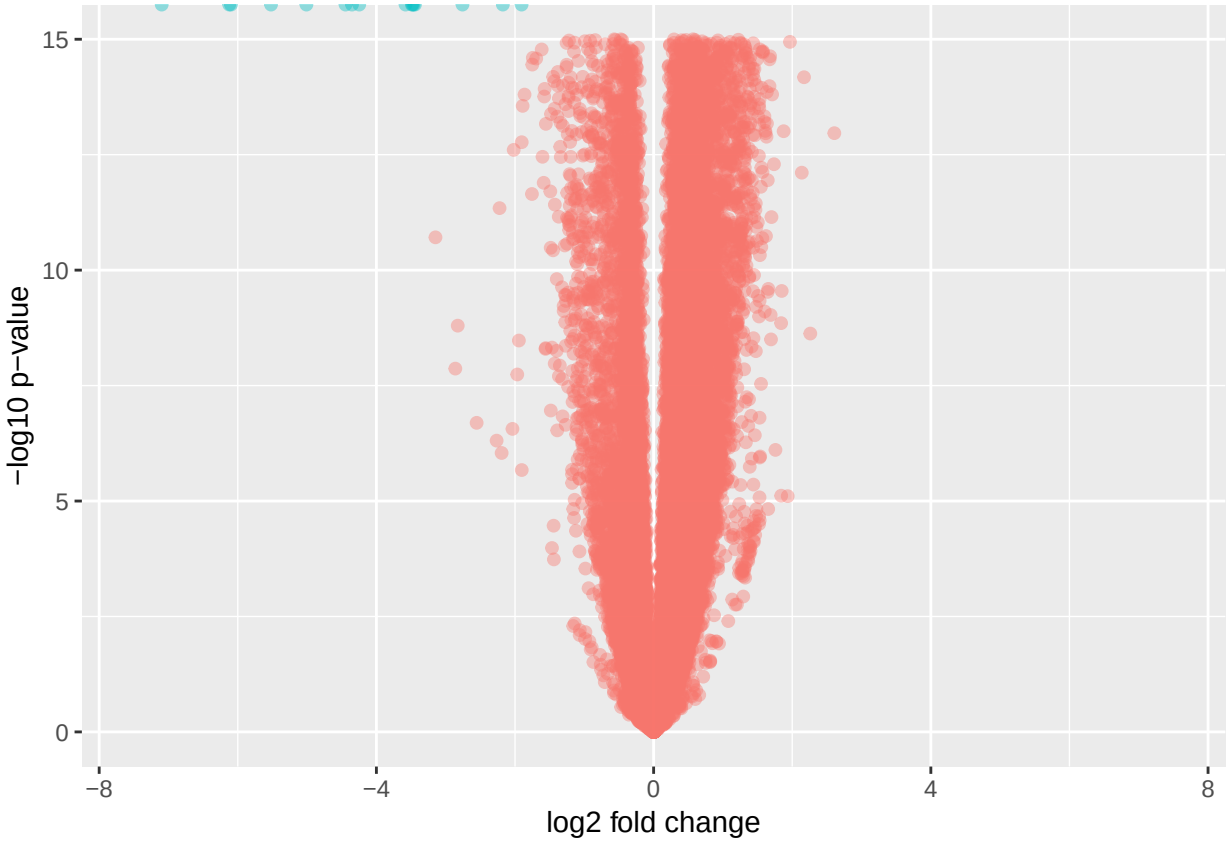
```
# add a column of absolute values of fold change to the annotated-only genes table
absLogFC <- abs(annotatedDE_onlyAnnotated$logFC)
annotatedDE_onlyAnnotatedWithAbs <- cbind(annotatedDE_onlyAnnotated, absLogFC)
# sort by p-values (ascending order) and absLogFoldChange (descending order)
sortedAnnotatedDEGsWithAbs <- annotatedDE_onlyAnnotatedWithAbs[with(annotatedDE_onlyAnnotatedWithAbs,
                                                                    order(PValue, -absLogFC)),]
```

```
# take the top 100 genes to plot a heatmap based on their CPM values
top100 <- (sortedAnnotatedDEGsWithAbs$ENSEMBL)[1:100]
matrixForHeatmap <- filteredLogCpmNoOuts[top100,]

# The column annotaion is the grouping normal/lesion. Set the cellheight and cellwidth to
# make the font readable
aheatmap(matrixForHeatmap, filename="heatmap.pdf", annCol=data.frame(group),
         main="Log(CountsPerMillion), normal and lesion samples", cellheight=7, cellwidth=8)
```

Volcano plot

```
# add a column of absolute values of fold change to 'all genes' table (not only the annotated ones)
absLogFC_allGenes <- abs(annotatedDE_allGenes$logFC)
annotatedDE_allGenesWithAbs <- cbind(annotatedDE_allGenes, absLogFC_allGenes)

# define the threshold from which genes will be colored: the FC of the 100th gene in
# the top 100
thresholdPvalue <- sortedAnnotatedDEGsWithAbs[100,4]
annotatedDE_allGenesWithAbs$thresholdPval <- as.factor(annotatedDE_allGenesWithAbs$PValue <= thresholdP
ggplot(data=annotatedDE_allGenesWithAbs, aes(x=logFC, y=-log10(PValue),
                                             colour=thresholdPval))+
  geom_point(alpha=0.4, size=1.75) +
  xlim(c(-7.5, 7.5)) + ylim(c(0, 15)) +
  xlab("log2 fold change") + ylab("-log10 p-value") +
  theme(legend.position="none")
```

```
## Warning: Removed 8072 rows containing missing values (geom_point).
```


15
10
5
0
−log10 p−value
−8
−4
0
4
8
log2 fold change