Perform Differential Expression analysis using DESeq2 package

Load Libraries

```
library(DESeq2)
library(pheatmap)
library(ggplot2)
library(tinytex)
```

Read counts data

```
read_count_data <- function(file_path){
counts_data <<- read.csv(file_path, row.names = 1)
return (head(counts_data))
}

read_count_data("pasilla_gene_exp.csv")
```

```
##             treated1 treated2 treated3 untreated1 untreated2 untreated3
## FBgn0000003        0        0        1          0          0          0
## FBgn0000008      140       88       70         92        161         76
## FBgn0000014        4        0        0          5          1          0
## FBgn0000015        1        0        0          0          2          1
## FBgn0000017     6205     3072     3334       4664       8714       3564
## FBgn0000018      722      299      308        583        761        245
##             untreated4
## FBgn0000003          0
## FBgn0000008         70
## FBgn0000014          0
## FBgn0000015          2
## FBgn0000017       3150
## FBgn0000018        310
```

Read Sample metadata

```
read_metadata <- function(file_path){
coldata <<- read.csv(file_path, row.names = 1)
return (coldata)
}

read_metadata("pasilla_meta.data.csv")
```

```
##            condition        type
## treated1     treated single-read
## treated2     treated  paired-end
## treated3     treated  paired-end
## untreated1 untreated single-read
## untreated2 untreated single-read
## untreated3 untreated  paired-end
## untreated4 untreated  paired-end
```

convert condition and types columns in coldata object to factor

```
convert_chr_to_factor <- function(){
coldata$condition <- factor(coldata$condition)
coldata$type <- factor(coldata$type)
}

convert_chr_to_factor()
```

make sure the row names in colData matches to the column names in counts_data

```
all(rownames(coldata) %in% colnames(counts_data))
```

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

IS the columns of the count matrix and the rows of the colData (information about samples) are in the same order?

```
all(rownames(coldata) == colnames(counts_data))
```

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

if Not, make them in the same order.

```
counts_data <- counts_data[, rownames(coldata)]
all(rownames(coldata) == colnames(counts_data))
```

Construct a DESeqDataSet.

```
deseqdataset <- DESeqDataSetFromMatrix(countData = counts_data,
                                       colData = coldata,
                                       design = ~ condition)
```

```
## Warning in DESeqDataSet(se, design = design, ignoreRank): some variables in
## design formula are characters, converting to factors
```

```
deseqdataset
```

```
## class: DESeqDataSet
## dim: 14599 7
## metadata(1): version
## assays(1): counts
## rownames(14599): FBgn0000003 FBgn0000008 ... FBgn0261574 FBgn0261575
## rowData names(0):
## colnames(7): treated1 treated2 ... untreated3 untreated4
## colData names(2): condition type
```

Pre-filtering: removing rows with low gene counts

keep rows that have at least 10 reads total

```
pre_filter <- function(){
keep <- rowSums(counts(deseqdataset)) >=10
deseqdataset <- deseqdataset[keep,]

return (deseqdataset)

}

pre_filter()
```

```
## class: DESeqDataSet
## dim: 9921 7
## metadata(1): version
## assays(1): counts
## rownames(9921): FBgn0000008 FBgn0000014 ... FBgn0261574 FBgn0261575
## rowData names(0):
## colnames(7): treated1 treated2 ... untreated3 untreated4
## colData names(2): condition type
```

Set the factor level

```
factor_level <- function(){

deseqdataset$condition <- relevel(deseqdataset$condition, ref = "untreated")

}

factor_level()
```

Differential expression analysis

```r
diff_expr_analysis <- function(){
deseqdataset <- DESeq(deseqdataset)
result <<- results(deseqdataset)
result01 <<- results(deseqdataset, alpha = 0.01 , lfcThreshold = 1.5)
return (result01)
}

diff_expr_analysis()
```

```
## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing

## log2 fold change (MLE): condition untreated vs treated
## Wald test p-value: condition untreated vs treated
## DataFrame with 14599 rows and 6 columns
##                baseMean log2FoldChange     lfcSE        stat    pvalue
##               <numeric>      <numeric> <numeric>   <numeric> <numeric>
## FBgn0000003    0.171569    -1.02604541  3.805503 -0.26962147  0.802971
## FBgn0000008   95.144079    -0.00215142  0.223884 -0.00960955  1.000000
## FBgn0000014    1.056572     0.49673557  2.160264  0.22994204  0.856490
## FBgn0000015    0.846723     1.88276170  2.106432  0.89381546  0.482051
## FBgn0000017 4352.592899     0.24002523  0.126024  1.90459450  1.000000
## ...                 ...            ...       ...         ...       ...
## FBgn0261571 8.73437e-02     -0.9002942  3.810165  -0.2362875  0.826890
## FBgn0261572 6.19714e+00      0.9591315  0.777017   1.2343759  0.757587
## FBgn0261573 2.24098e+03     -0.0126158  0.112701  -0.1119412  1.000000
## FBgn0261574 4.85774e+03     -0.0152569  0.193148  -0.0789905  1.000000
## FBgn0261575 1.06836e+01     -0.1635594  0.938909  -0.1742016  0.960903
##                  padj
##             <numeric>
## FBgn0000003         1
## FBgn0000008         1
## FBgn0000014         1
## FBgn0000015         1
## FBgn0000017         1
## ...               ...
## FBgn0261571         1
## FBgn0261572         1
## FBgn0261573         1
## FBgn0261574         1
## FBgn0261575         1
```

Top 10 differentail expression genes

```
ordered_result <- result01[order(result01$padj, decreasing = FALSE), ]
top10 <- head(ordered_result, n=10)
top10
```

```
## log2 fold change (MLE): condition untreated vs treated
## Wald test p-value: condition untreated vs treated
## DataFrame with 10 rows and 6 columns
##              baseMean log2FoldChange     lfcSE      stat      pvalue
##             <numeric>      <numeric> <numeric> <numeric>   <numeric>
## FBgn0039155  730.5958        4.61901 0.1687068  27.37895 1.29498e-76
## FBgn0003360 4343.0354        3.17967 0.1435262  22.15395 6.15888e-32
## FBgn0039827  261.9162        4.16252 0.2325888  17.89646 1.21275e-30
## FBgn0025111 1501.4105       -2.89986 0.1269205 -22.84788 1.37758e-28
## FBgn0034736  225.8764        3.51144 0.2146721  16.35722 3.63283e-21
## FBgn0035085  638.2326        2.56041 0.1372952  18.64896 5.65484e-15
## FBgn0034434  114.6256        3.64257 0.2782855  13.08932 6.84746e-15
## FBgn0029167 3706.1165        2.19700 0.0969889  22.65209 3.32623e-13
## FBgn0085359   68.6100        4.91813 0.4959814   9.91595 2.75756e-12
## FBgn0024288   58.8511        4.58584 0.4650117   9.86177 1.61111e-11
##                    padj
##               <numeric>
## FBgn0039155 1.60033e-72
## FBgn0003360 3.80557e-28
## FBgn0039827 4.99572e-27
## FBgn0025111 4.25602e-25
## FBgn0034736 8.97890e-18
## FBgn0035085 1.16471e-11
## FBgn0034434 1.20887e-11
## FBgn0029167 5.13819e-10
## FBgn0085359 3.78643e-09
## FBgn0024288 1.99101e-08
```

Explore results

```
summary(result)
```

```
##
## out of 12359 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)       : 540, 4.4%
## LFC < 0 (down)     : 521, 4.2%
## outliers [1]       : 1, 0.0081%
## low counts [2]     : 4035, 33%
## (mean count < 7)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

```r
summary(result01)
```

```
##
## out of 12359 with nonzero total read count
## adjusted p-value < 0.01
## LFC > 1.50 (up)     : 11, 0.089%
## LFC < -1.50 (down) : 7, 0.057%
## outliers [1]       : 1, 0.0081%
## low counts [2]     : 0, 0%
## (mean count < 0)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

How many adjusted p-values less than 0.01?

```r
sum(result01$padj < 0.01 , na.rm = TRUE)
```

```
## [1] 18
```

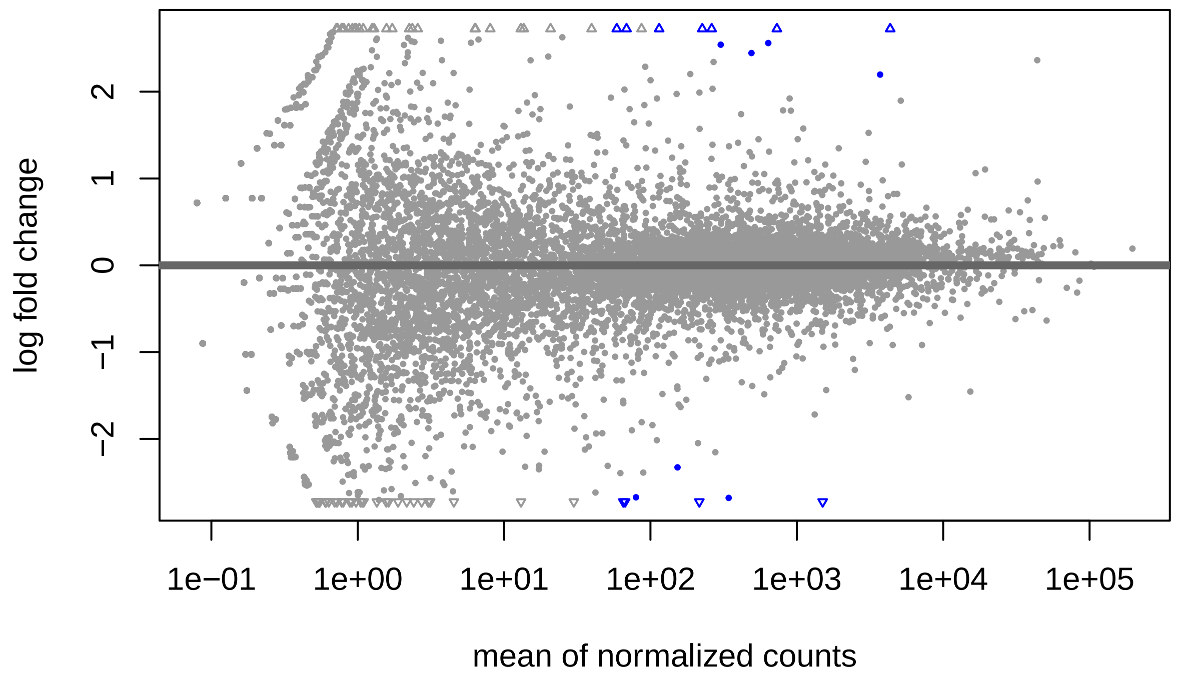
Write results to CSV file

```r
write_sig_genes <- function(out_path){
write.csv(ordered_result, file = out_path)
}

write_sig_genes("Significant genes.csv")
```

Visualizing the results

MA-plot

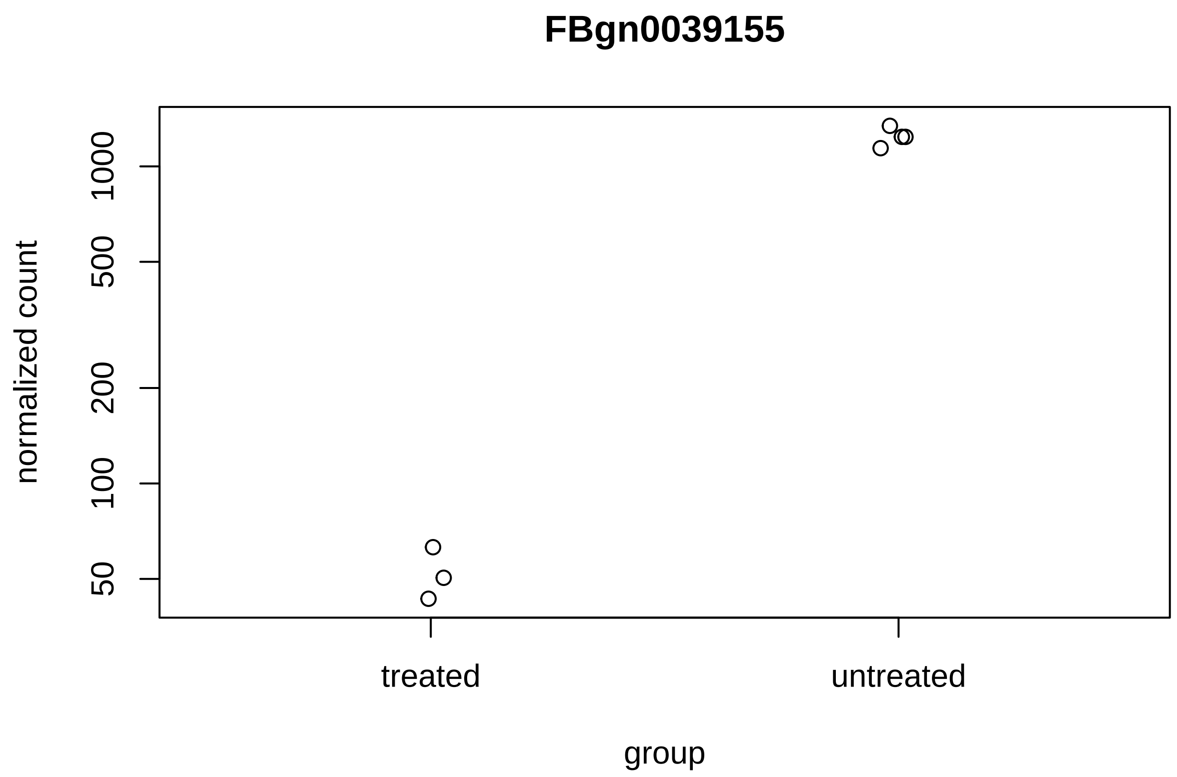
```r
ma_plot <- function(){
plotMA(result01)
}

ma_plot()
```



Plot counts

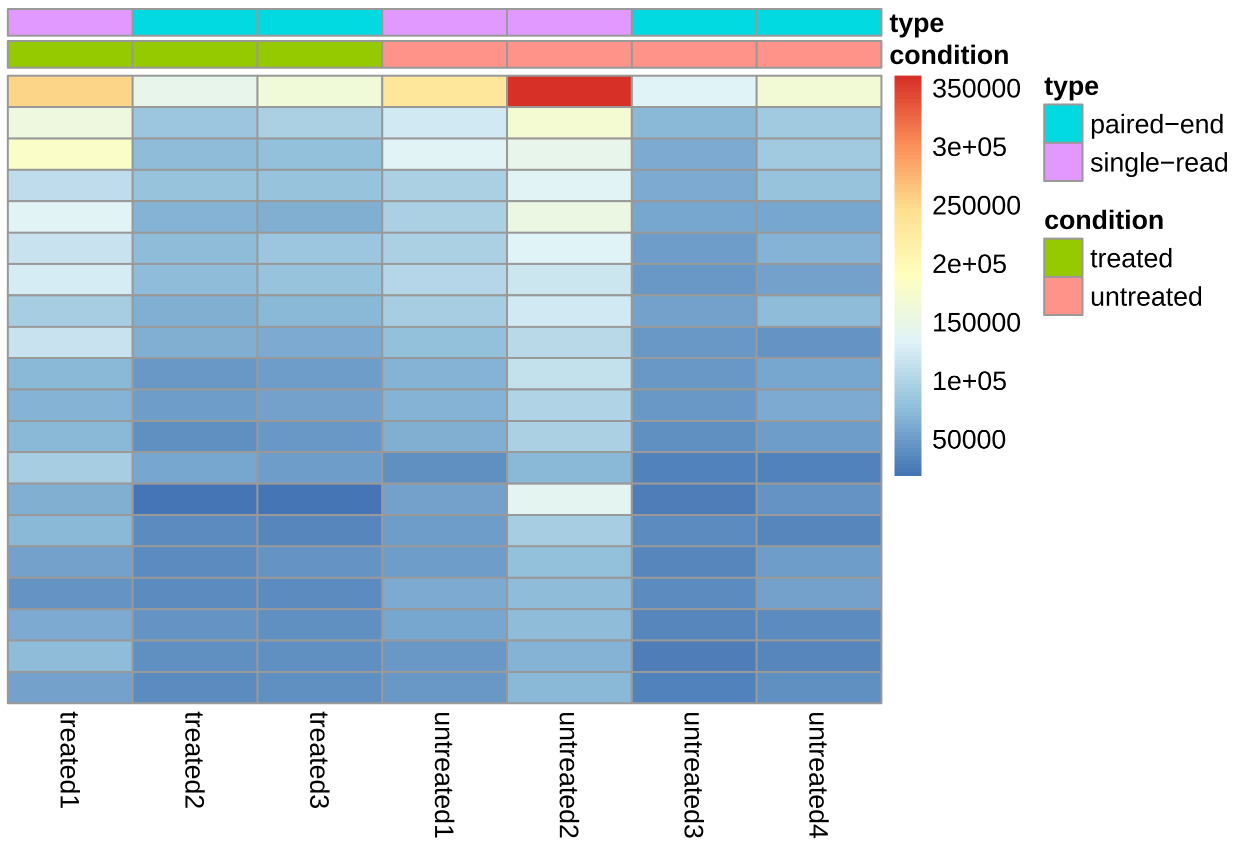
Here we specify the gene which had the smallest p value from the results table

```
plot_counts <- function(){
plotCounts(deseqdataset, gene = which.min(result01$padj), intgroup = "condition")
}

plot_counts()
```



Heatmap

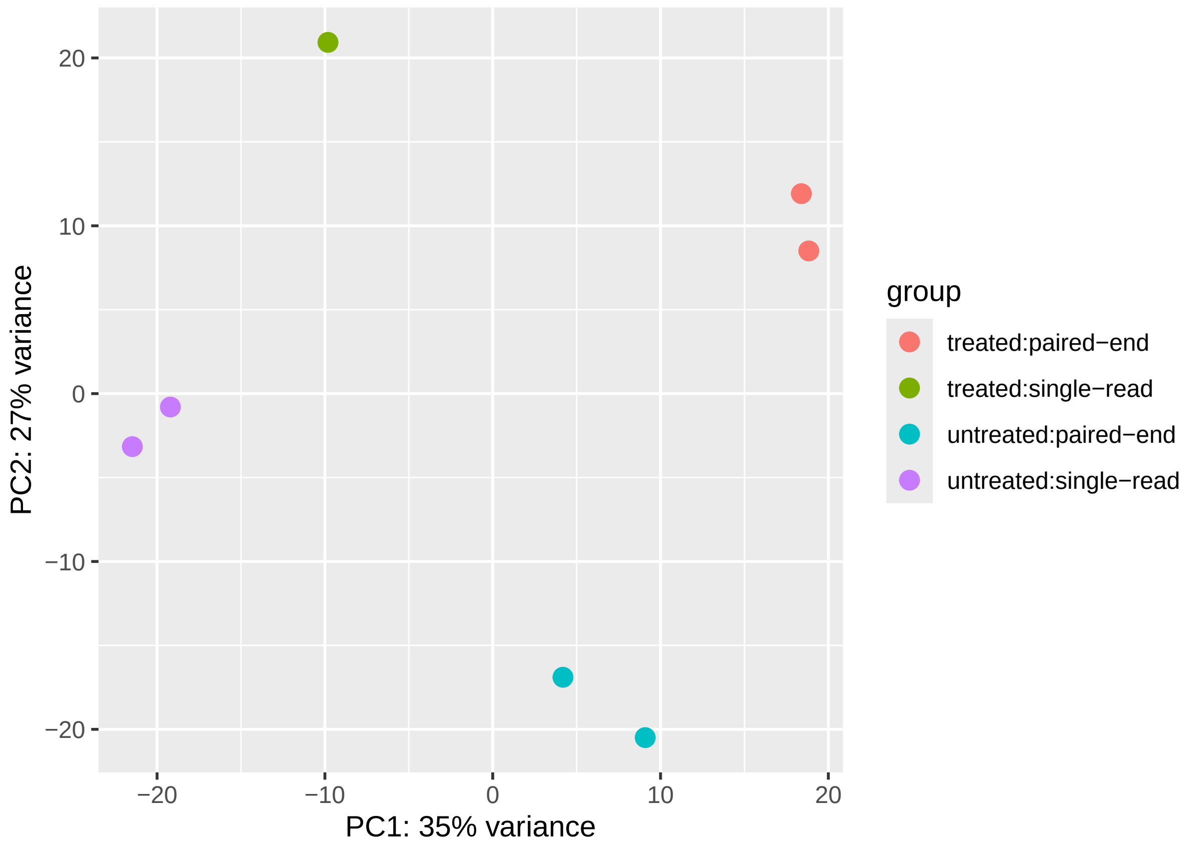
```
heatmap <- function(){
select <- order(rowMeans(counts(deseqdataset)),
                decreasing = TRUE)[1:20]

df <- as.data.frame(colData(deseqdataset)[,c("condition","type")])
pheatmap::pheatmap(assay(deseqdataset)[select,], cluster_rows = FALSE,
                   show_rownames = FALSE, cluster_cols = FALSE,
                   annotation_col = df)
}

heatmap()
```



PCA plot

```r
pca_plot <- function(){
normalized = normTransform(deseqdataset)
plotPCA(normalized, intgroup=c("condition","type"))
}

pca_plot()
```

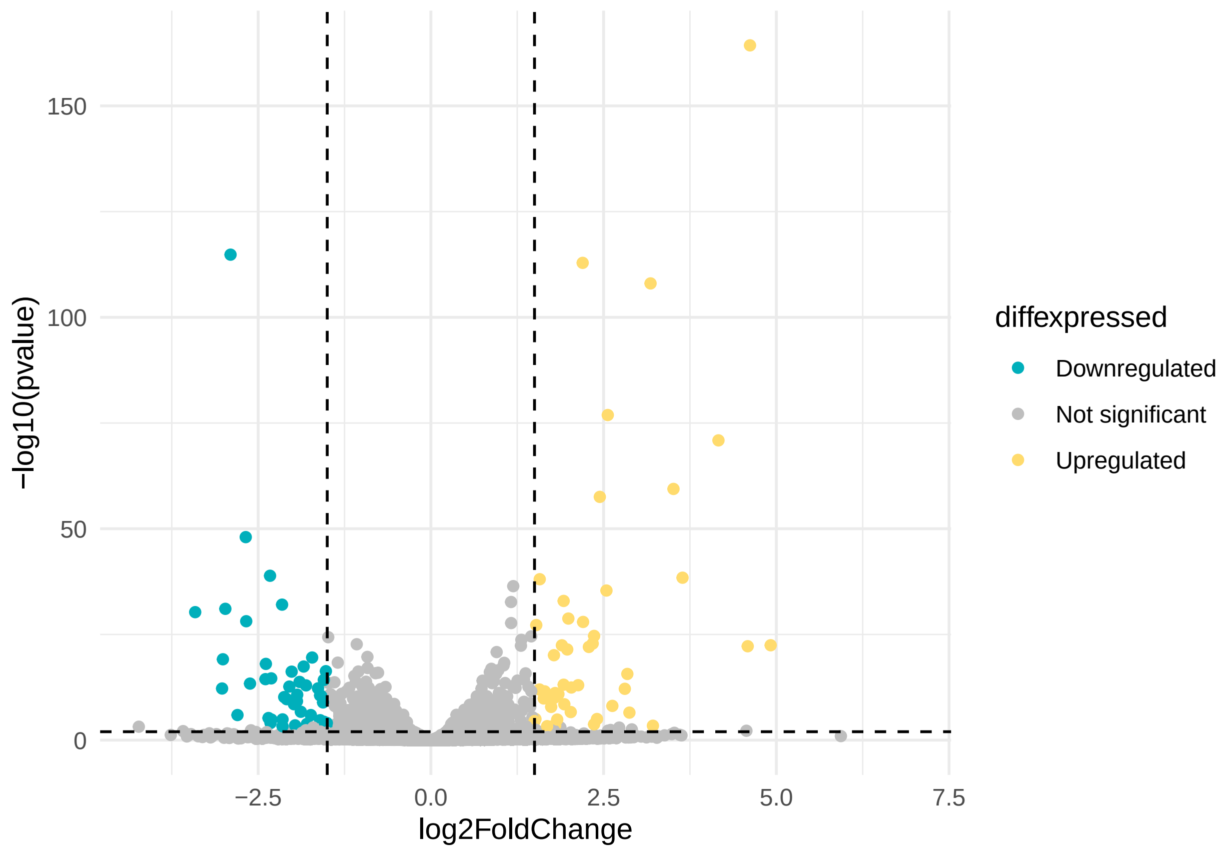
```
## using ntop=500 top features by variance
```



Volcano plot

```
vplcano_plot <- function(){

result.df <- as.data.frame(result)

result.df$diffexpressed <- "NO"
result.df$diffexpressed[result.df$log2FoldChange > 1.5 &
                        result.df$padj < 0.01] <- "UP"
result.df$diffexpressed[result.df$log2FoldChange < -1.5 &
                        result.df$padj < 0.01] <- "DOWN"


ggplot(data = result.df, aes(x = log2FoldChange, y = -log10(pvalue),
                             col = diffexpressed))+
  geom_point()+ theme_minimal()+
  geom_vline(xintercept = c(-1.5, 1.5), col = "black", linetype = 'dashed') +
  geom_hline(yintercept = -log10(0.01), col = "black", linetype = 'dashed') +
  scale_color_manual(values = c("#00AFBB", "grey", "#FFDB6D"),
  labels = c("Downregulated", "Not significant", "Upregulated"))

}

vplcano_plot()
```

```
## Warning: Removed 2241 rows containing missing values or values outside the scale range
## ('geom_point()').
```


−log10(pvalue)
log2FoldChange
diffexpressed
Downregulated
Not significant
Upregulated