DEGreport

Lorena Pantano[*1]

[1]Harvard TH Chan School of Public Health, Boston, US

[*]lorena.pantano@gmail.com

December 18, 2017

Package

DEGreport 1.14.1

Contents

```
library(DEGreport)
data(humanGender)
```

1 General QC figures from DE analysis

We are going to do a differential expression analysis with edgeR/DESeq2. We have an object that is coming from the edgeR package. It contains a gene count matrix for 85 TSI HapMap individuals, and the gender information. With that, we are going to apply the 'glmFit' function or 'DESeq2' to get genes differentially expressed between males and females.

```
library(DESeq2)
idx <- c(1:10, 75:85)
dds <- DESeqDataSetFromMatrix(assays(humanGender)[[1]][1:1000, idx],
                              colData(humanGender)[idx,], design=~group)
dds <- DESeq(dds)
res <- results(dds)
```

We need to extract the experiment design data.frame where the condition is Male or Female.

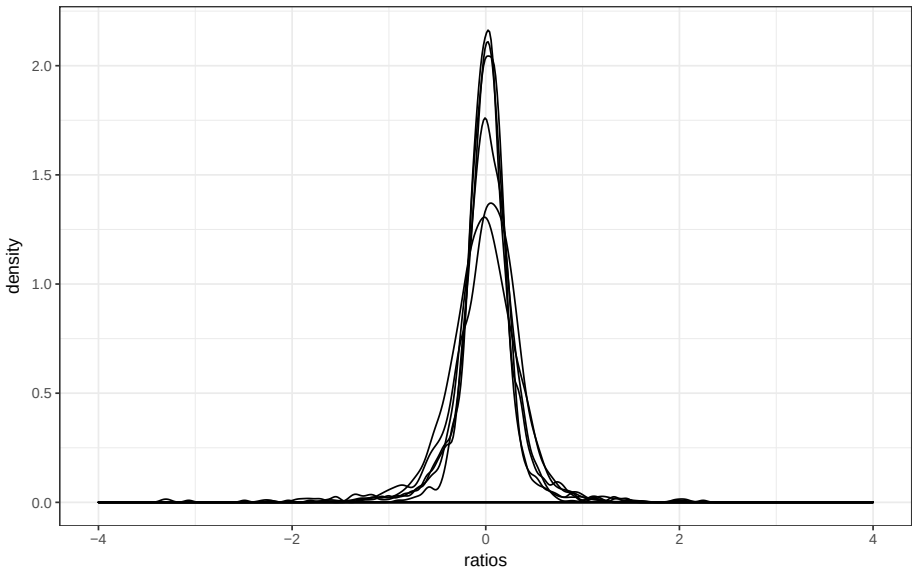
```
counts <- counts(dds, normalized = TRUE)
design <- as.data.frame(colData(dds))
```

1.1 Size factor QC

A main assumption in library size factor calculation of edgeR and DESeq2 (and others) is that the majority of genes remain unchanged. Plotting the distribution of gene ratios between each gene and the average gene can show how true this is. Not super useful for many samples because the plot becomes crowed.

```
degCheckFactors(counts[, 1:6])
```

```
## Warning in bind_rows_(x, .id): Unequal factor levels: coercing to character

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
into character vector
```



1.2 Mean-Variance QC plots

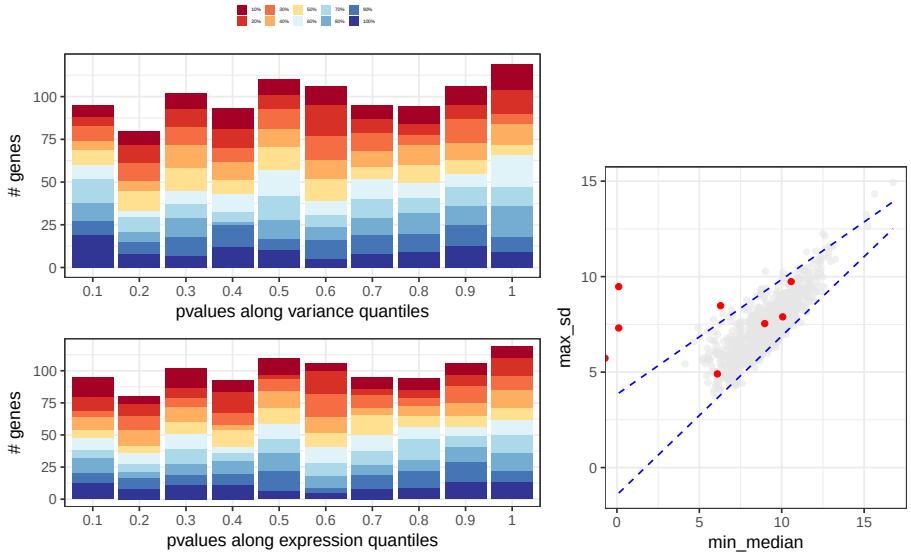
p-value distribution gives an idea on how well you model is capturing the input data and as well whether it could be some problem for some set of genes. In general, you expect to have a flat distribution with peaks at 0 and 1. In this case, we add the mean count information to check if any set of genes are enriched in any specific p-value range.

Variation (dispersion) and average expression relationship shouldn't be a factor among the differentially expressed genes. When plotting average mean and standard deviation, significant genes should be randomly distributed.

In this case, it would be good to look at the ones that are totally outside the expected correlation.

You can put this tree plots together using `degQC`.

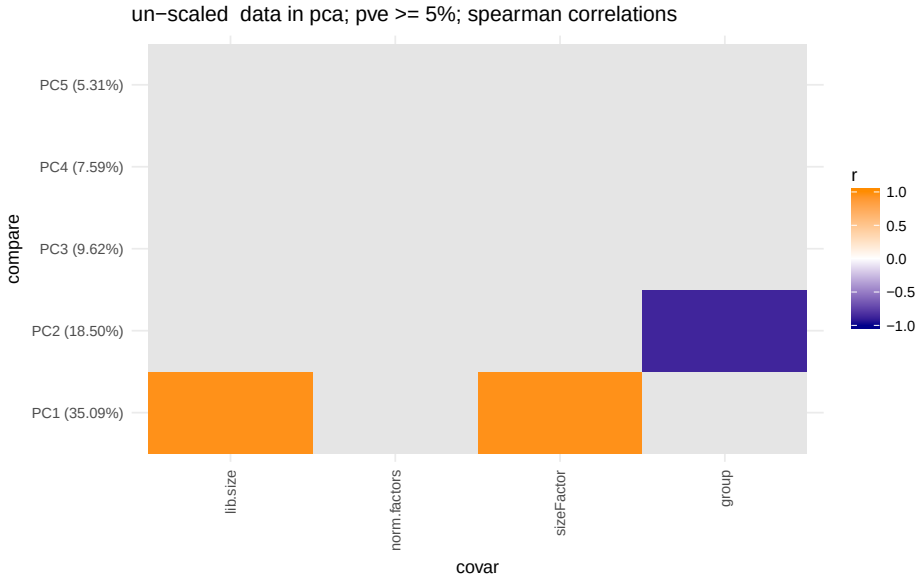
```
degQC(counts, design[["group"]], pvalue = res[["pvalue"]])
```



1.3 Covariates effect on count data

Another important analysis to do if you have covariates is to calculate the correlation between PCs from PCA analysis to different variables you may think are affecting the gene expression. This is a toy example of how the function works with raw data, where clearly library size correlates with some of the PCs.

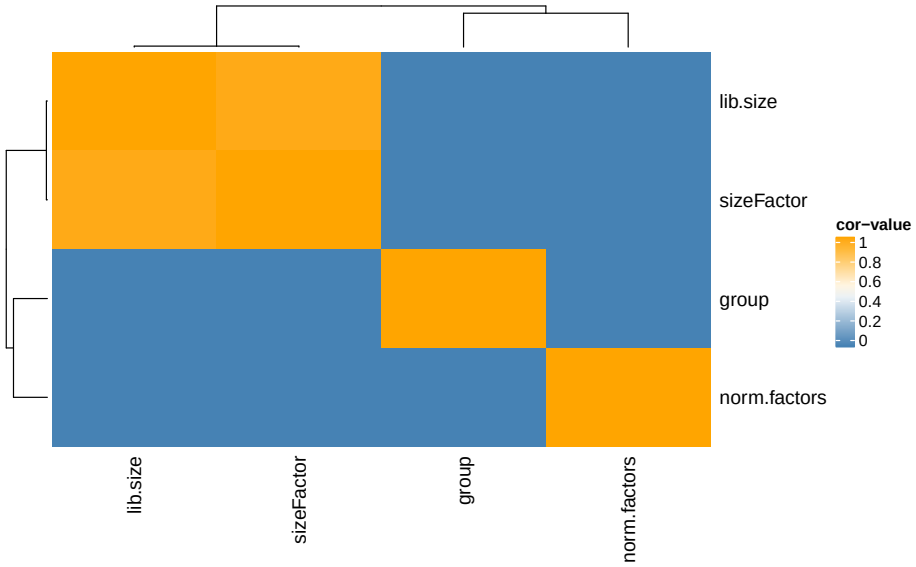
```
resCov <- degCovariates(log2(counts(dds)+0.5),
                        colData(dds))
```



1.4 Covariates correlation with metrics

Also, the correlation among covariates and metrics from the analysis can be tested. This is useful when the study has multiple variables, like in clinical trials. The following code will return a correlation table, and plot the correlation heatmap for all the covariates and metrics in a table.

```
cor <- degCorCov(colData(dds))
```




```
names(cor)
```

```
## [1] "cor"    "corMat" "fdrMat" "plot"
```

1.5 QC report

A quick HTML report can be created with `createReport` to show whether a DE analysis is biased to a particular set of genes. It contains the output of `degQC`, `degVB` and `degMB`.

```
createReport(colData(dds)[["group"]], counts(dds, normalized = TRUE),
             row.names(res)[1:20], res[["pvalue"]], path = "~/Downloads")
```

2 Report from DESeq2 analysis

Here, we show some useful plots for differentially expressed genes.

2.1 Contrasts

DEGSet is a class to store the DE results like the one from `results` function. *DESeq2* offers multiple way to ask for contrasts/coefficients. With `degComps` is easy to get multiple results in a single object:

```
degs <- degComps(dds, combs = "group",
                 contrast = list("group_Male_vs_Female",
                                 c("group", "Female", "Male")))
names(degs)
```

```
## [1] "group_Male_vs_Female" "group_Female_vs_Male"
```

`degs` contains 3 elements, one for each contrast/coefficient asked for. It contains the results output in the element `raw` and the output of `lfcShrink` in the element `shrunken`. To obtain the results from one of them, use the method `dge`:

```
deg(degs[[1]])
```

```
## log2 fold change (MAP): group Male vs Female
## Wald test p-value: group Male vs Female
## DataFrame with 1000 rows and 6 columns
##                   baseMean log2FoldChange      lfcSE         stat
##                  <numeric>      <numeric>  <numeric>    <numeric>
## ENSG00000067048 1025.03783      1.9394875 0.10069402    23.994380
## ENSG00000012817  411.54387      3.7005640 0.09898885    21.804517
## ENSG00000067646  169.81477      3.3228457 0.09951933    15.483847
## ENSG00000005889  670.86191     -0.4894347 0.09293631    -5.263708
## ENSG00000006757   92.66111     -0.4729262 0.09927492    -4.757006
## ...                    ...            ...        ...          ...
## ENSG00000068120  1214.0967  -0.0001143219 0.07972266 -0.001433998
## ENSG00000072062   935.3172   0.0005592191 0.09153522  0.006108720
## ENSG00000076770  1019.7964   0.0008500818 0.10166633  0.008361930
## ENSG00000078967   166.4221   0.0004157454 0.09597580  0.004329710
## ENSG00000079246  5226.3390   0.0001495843 0.09262051  0.001614995
##                       pvalue          padj
##                    <numeric>     <numeric>
## ENSG00000067048 3.183020e-127 3.183020e-124
## ENSG00000012817 2.102139e-105 1.051070e-102
## ENSG00000067646  4.459844e-54  1.486615e-51
## ENSG00000005889  1.411785e-07  3.529463e-05
## ENSG00000006757  1.964856e-06  3.929713e-04
## ...                       ...           ...
## ENSG00000068120     0.9988558     0.9988558
## ENSG00000072062     0.9951260     0.9988558
## ENSG00000076770     0.9933282     0.9988558
## ENSG00000078967     0.9965454     0.9988558
## ENSG00000079246     0.9987114     0.9988558
```

By default it would output the `shrunken` table always, as defined by `degDefault`, that contains the default table to get.

To get the original results table, use the parameter as this:

```
deg(degs[[1]], "raw", "tibble")
```

```
## # A tibble: 1,000 x 7
##               gene   baseMean log2FoldChange      lfcSE      stat
##  *           <chr>      <dbl>          <dbl>      <dbl>     <dbl>
##  1 ENSG00000067048 1025.03783     10.1571705 0.42331456 23.994380
##  2 ENSG00000012817  411.54387      9.2394007 0.42373792 21.804517
##  3 ENSG00000067646  169.81477     10.1874916 0.65794317 15.483847
##  4 ENSG00000005889  670.86191     -0.6919265 0.13145228 -5.263708
##  5 ENSG00000006757   92.66111     -0.7666012 0.16115204 -4.757006
##  6 ENSG00000073282  220.15603     -1.8685615 0.42061202 -4.442482
##  7 ENSG00000005302 2026.54990     -0.7418952 0.17634123 -4.207157
##  8 ENSG00000005020 1233.86316      0.3888370 0.09523118  4.083085
##  9 ENSG00000003400  393.62677      0.6803243 0.17664751  3.851310
## 10 ENSG00000069702  106.67010     -1.6323189 0.45856109 -3.559654
## # ... with 990 more rows, and 2 more variables: pvalue <dbl>, padj <dbl>
```

Note that the format of the output can be changed to tibble, or data.frame with a third parameter `tidy`.

The table will be always sorted by padj.

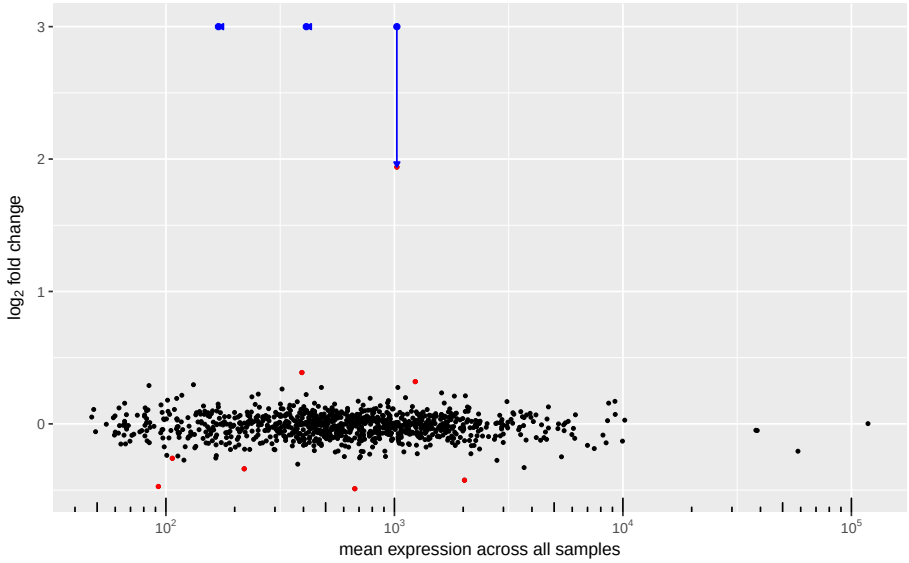
And easy way to get significant genes is:

```
significants(degs[[1]], fc = 0, fdr = 0.05)
```

```
##  [1] "ENSG00000012817" "ENSG00000067646" "ENSG00000067048" "ENSG00000005889"
##  [5] "ENSG00000006757" "ENSG00000005302" "ENSG00000003400" "ENSG00000073282"
##  [9] "ENSG00000005020" "ENSG00000069702"
```

Since log2FoldChange are shrunken, the method for DEGSet class now can plot these changes as follow:

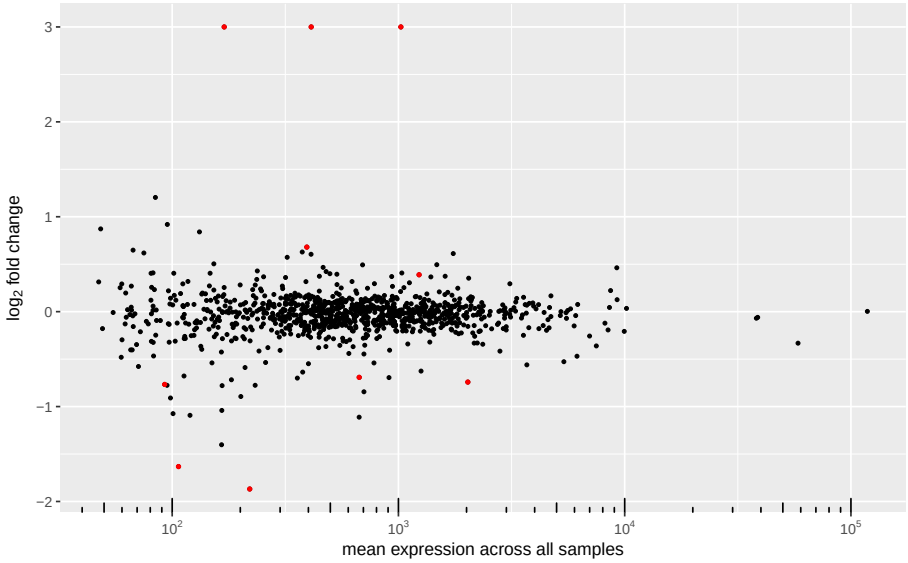
```
plotMA(degs[[1]], diff = 2, limit = 3)
```



The blue arrows indicate how foldchange is affected by this new feature.

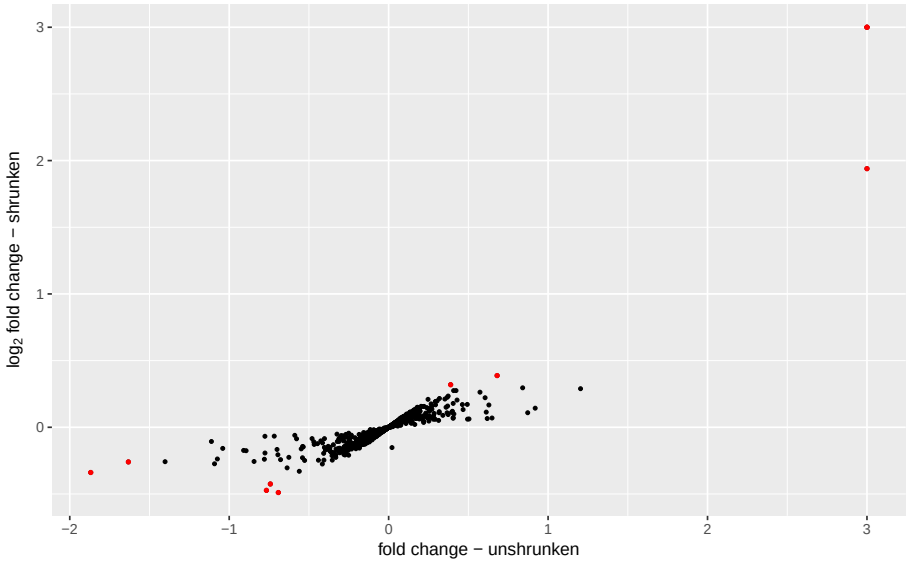
As well, it can plot the original MA plot:

```
plotMA(degs[[1]], diff = 2, limit = 3, raw = TRUE)
```



or the correlation between the original log2FoldChange and the new ones:

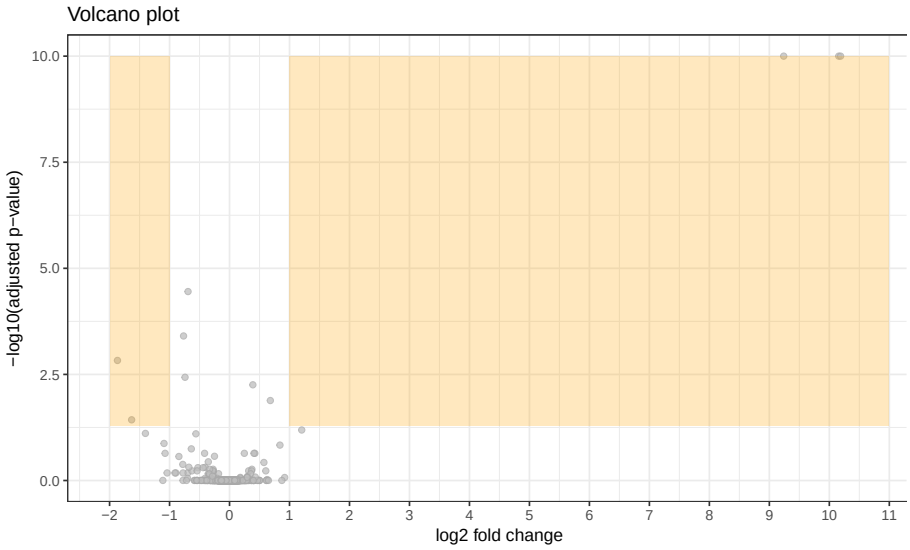
```
plotMA(degs[[1]], limit = 3, correlation = TRUE)
```



2.2 Volcano plots

Volcano plot using the output of *DESeq2*. It mainly needs data.frame with two columns (logFC and pVal). Specific genes can be plot using the option `plot_text` (subset of the previous data.frame with a 3rd column to be used to plot the gene name).

```
res[["id"]] <- row.names(res)
# show <- as.data.frame(res[1:10, c("log2FoldChange", "padj", "id")])
degVolcano(res[,c("log2FoldChange", "padj")])
```

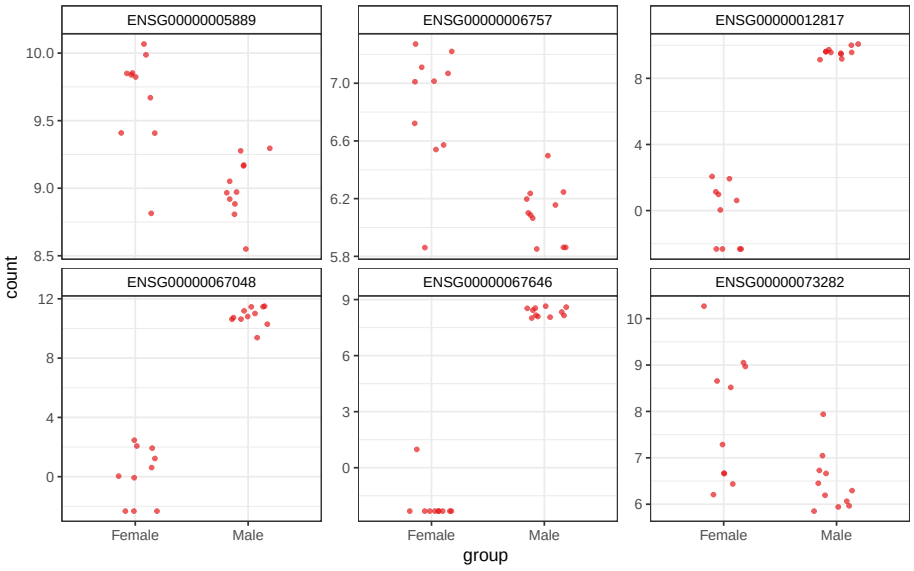


Note that the function is compatible with DEGset. Using `degVolcano(degs[[1]])` is valid.

2.3 Gene plots

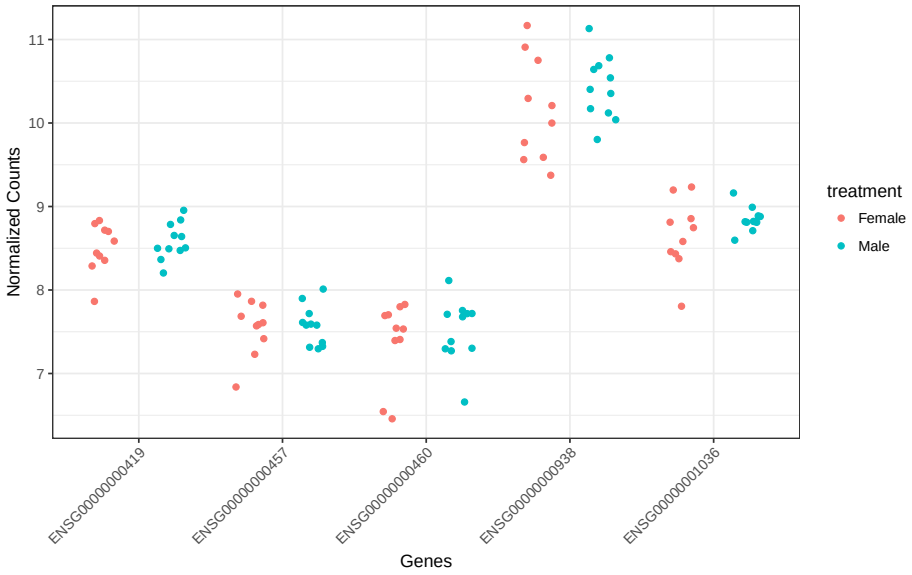
Plot top genes coloring by group. Very useful for experiments with nested groups. 'xs' can be 'time' or 'WT'/'KO', and 'group' can be 'treated'/'untreated'. Another classification can be added, like 'batch' that will plot points with different shapes.

```
degPlot(dds = dds, res = res, n = 6, xs = "group")
```



Another option for plotting genes in a wide format:

```
degPlotWide(dds, rownames(dds)[1:5], group="group")
```

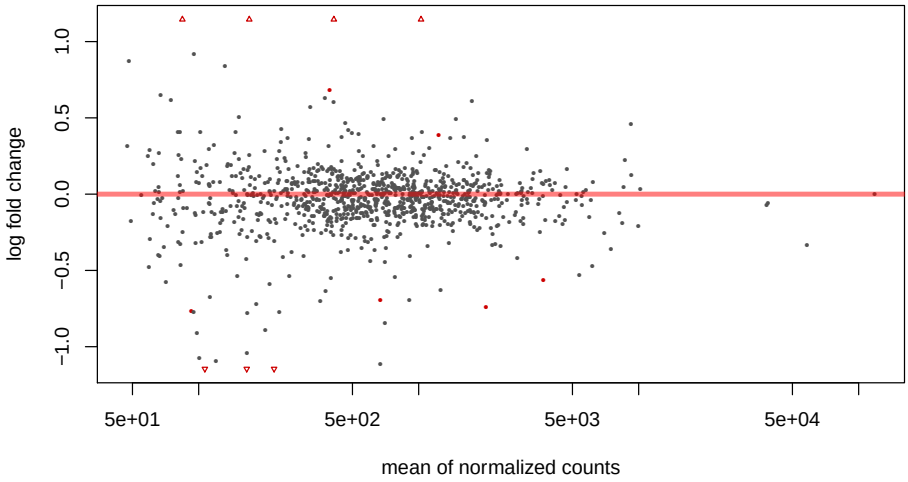


2.4 Full report

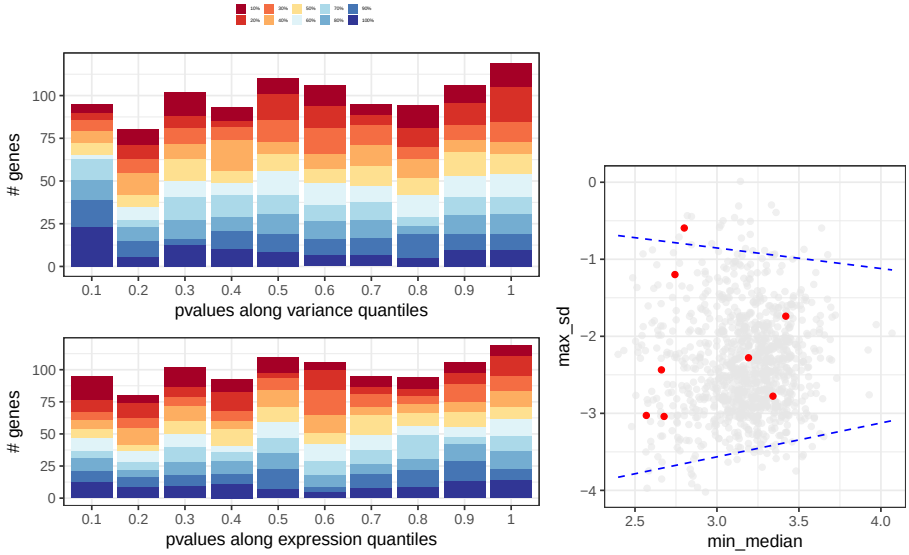
If you have a DESeq2 object, you can use degResults to create a full report with markdown code inserted, including figures and table with top de-regulated genes, GO enrichment analysis and heatmaps and PCA plots. If you set `path_results`, different files will be saved there.

```
resreport <- degResults(dds = dds, name = "test", org = NULL,
                        do_go = FALSE, group = "group", xs = "group",
                        path_results = NULL)
```

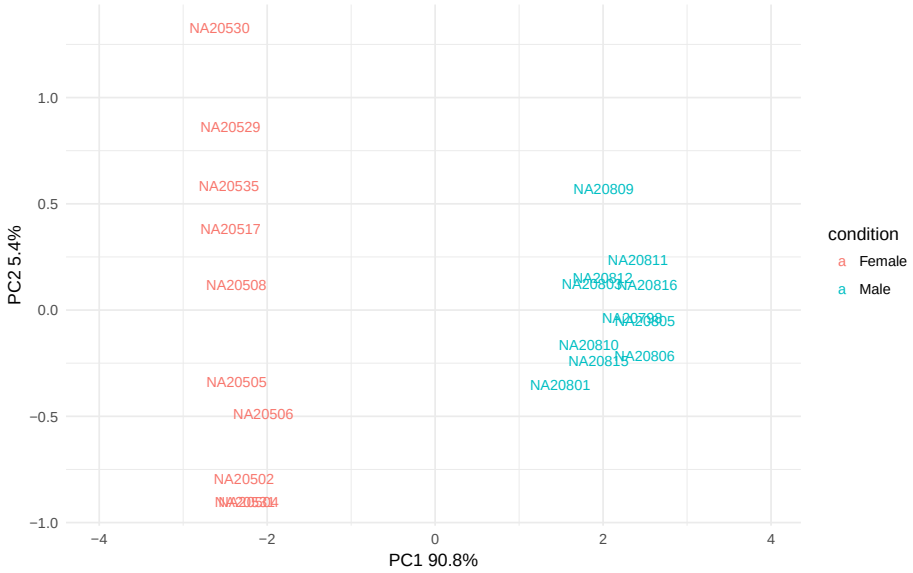
```
## ## Comparison:  test {.tabset}
##
##
##  <br>out of 1000 with nonzero total read count<br>adjusted p-value < 0.1<br>LFC > 0 (up)     : 6, 0.6% <br
##
##
## Differential expression file at:  test_de.csv
##
## Normalized counts matrix file at:  test_log2_counts.csv
##
## ### MA plot plot
```




```
##
##
## ### Volcano plot
##
##
##
## ### QC for DE genes
```




```
##
##
## ### Most significants, FDR< 0.05  and log2FC >  0.1 :  10
```




```
##
##
##
## ### Plots top 9 most significants
##
##
##
## ### Top DE table
##
##
## \begin{tabular}{l|r|r|r|r|r|r|r}
## \hline
##   & baseMean & log2FoldChange & lfcSE & stat & pvalue & padj & absMaxLog2FC\\
## \hline
## ENSG00000067048 & 1025.03783 & 10.1571705 & 0.4233146 & 23.994380 & 0.0000000 & 0.0000000 & 10.1571705\\
## \hline
## ENSG00000012817 & 411.54387 & 9.2394007 & 0.4237379 & 21.804517 & 0.0000000 & 0.0000000 & 9.2394007\\
## \hline
## ENSG00000067646 & 169.81477 & 10.1874916 & 0.6579432 & 15.483847 & 0.0000000 & 0.0000000 & 10.1874916\\
## \hline
## ENSG00000005889 & 670.86191 & -0.6919265 & 0.1314523 & -5.263708 & 0.0000001 & 0.0000353 & 0.6919265\\
## \hline
## ENSG00000006757 & 92.66111 & -0.7666012 & 0.1611520 & -4.757006 & 0.0000020 & 0.0003930 & 0.7666012\\
## \hline
## ENSG00000073282 & 220.15603 & -1.8685615 & 0.4206120 & -4.442482 & 0.0000089 & 0.0014821 & 1.8685615\\
## \hline
## ENSG00000005302 & 2026.54990 & -0.7418952 & 0.1763412 & -4.207157 & 0.0000259 & 0.0036943 & 0.7418952\\
## \hline
## ENSG00000005020 & 1233.86316 & 0.3888370 & 0.0952312 & 4.083085 & 0.0000444 & 0.0055552 & 0.3888370\\
## \hline
## ENSG00000003400 & 393.62677 & 0.6803243 & 0.1766475 & 3.851310 & 0.0001175 & 0.0130542 & 0.6803243\\
## \hline
## ENSG00000069702 & 106.67010 & -1.6323189 & 0.4585611 & -3.559654 & 0.0003713 & 0.0371343 & 1.6323189\\
```

```
## \hline
## ENSG00000010278 & 84.30823 & 1.2035871 & 0.3554857 & 3.385754 & 0.0007098 & 0.0645300 & 1.2035871\\
## \hline
## ENSG00000023171 & 165.31692 & -1.4022259 & 0.4236024 & -3.310240 & 0.0009322 & 0.0776799 & 1.4022259\\
## \hline
## ENSG00000072501 & 3694.76013 & -0.5604815 & 0.1707989 & -3.281529 & 0.0010325 & 0.0794200 & 0.5604815\\
## \hline
## ENSG00000070018 & 119.89049 & -1.0921227 & 0.3512409 & -3.109327 & 0.0018751 & 0.1339388 & 1.0921227\\
## \hline
## ENSG00000059377 & 131.98111 & 0.8405094 & 0.2744635 & 3.062372 & 0.0021959 & 0.1463935 & 0.8405094\\
## \hline
## ENSG00000008277 & 377.43955 & -0.6368732 & 0.2136506 & -2.980910 & 0.0028739 & 0.1796208 & 0.6368732\\
## \hline
## ENSG00000005059 & 479.12528 & 0.4225402 & 0.1492246 & 2.831571 & 0.0046320 & 0.2289281 & 0.4225402\\
## \hline
## ENSG00000012963 & 1829.21224 & 0.2471040 & 0.0861859 & 2.867104 & 0.0041425 & 0.2289281 & 0.2471040\\
## \hline
## ENSG00000038427 & 100.87217 & -1.0743654 & 0.3810269 & -2.819658 & 0.0048075 & 0.2289281 & 1.0743654\\
## \hline
## ENSG00000068079 & 1035.17996 & 0.4075632 & 0.1415563 & 2.879160 & 0.0039874 & 0.2289281 & 0.4075632\\
## \hline
## \end{tabular}
```

2.5 Interactive shiny-app

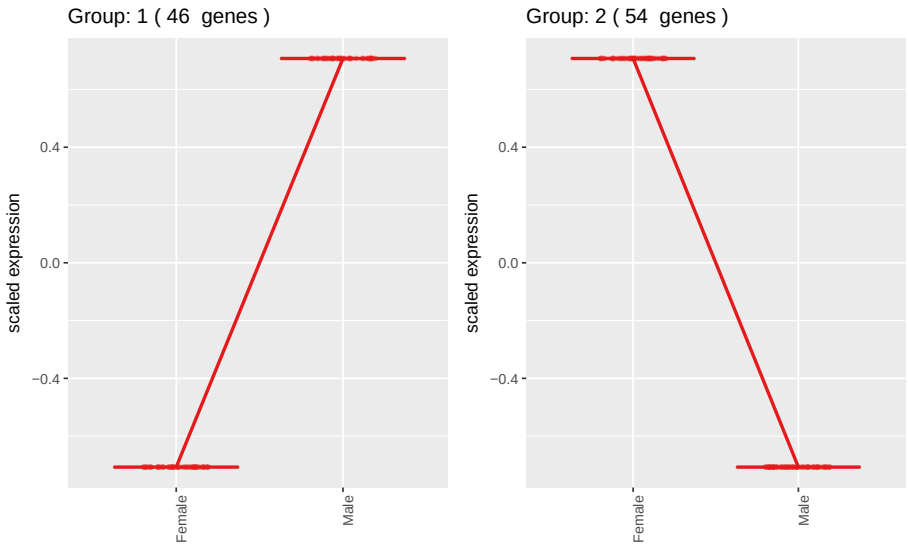
Browsing gene expression can help to validate results or select some gene for downstream analysis. Run the following lines if you want to visualize your expression values by condition:

```
degObj(counts, design, "degObj.rda")
library(shiny)
shiny::runGitHub("lpantano/shiny", subdir="expression")
```

3 Detect patterns of expression

In this section, we show how to detect pattern of expression. Mainly useful when data is a time course experiment. `degPatterns` needs a expression matrix, the design experiment and the column used to group samples.

```
ma = assay(rlog(dds))[row.names(res)[1:100],]
res <- degPatterns(ma, design, time = "group", col=NULL)
```



4 Useful functions

This section shows some useful functions during DEG analysis.

`degFilter` helps to filter genes with a minimum read count by group.

```
cat("gene in original count matrix: 1000")

## gene in original count matrix: 1000

filter_count <- degFilter(counts(dds),
                          design, "group",
                          min=1, minreads = 50)
cat("gene in final count matrix", nrow(filter_count))

## gene in final count matrix 940
```