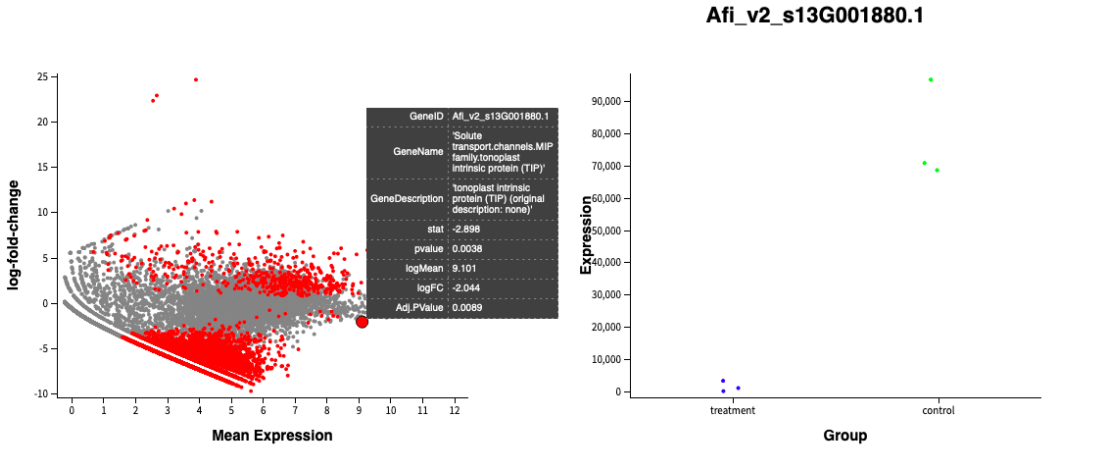
DESeq2+Glimma

Giorgio Bolchi

5/6/2022

Description

This script is a combination of the **DESeq2** Differential Gene Expression Analysis tool and the **Glimma** interactive html graphic tool available through **Bioconductor**.



Search:

GeneID	GeneName	GeneDescription	stat	pvalue	logMean	logFC	Adj.PValue
Afi_v2_s28G002430.2	'RNA biosynthesis.transcriptional regulation.transcription factor (bHLH)'	'transcription factor (bHLH) (original description: none)'	14.49	1.463e-47	7.252	6.79	2.47e-43
Afi_v2_s2124G000060.1	'not assigned.not annotated'	' no hits & (original description: none)'	13.96	2.649e-44	7.585	1.875	2.236e-40
Afi_v2_s12G001740.3	'Lipid metabolism.fatty acid biosynthesis.plastidial fatty acid synthase (ptFAS) system.fatty acid chain termination.oleoyl-ACP thioesterase'	'oleoyl-ACP thioesterase (original description: none)'	-13.11	2.79e-39	6.598	-6.91	1.57e-35

Sources:

DESeq2:

https://github.com/mikelove/DESeq2/li>

https://www.bioconductor.org/packages/release/bioc/manuals/DESeq2/man/DESeq2.pdf (manual)

Glimma:

https://github.com/Shians/Glimma

https://bioconductor.org/packages/release/bioc/manuals/Glimma/man/Glimma.pdf (manual)

Script

Load libraries

(If not installed yet)

```
install.packages("BiocManager")
BiocManager::install("DESeq2")
BiocManager::install("Glimma")
```

```
library("DESeq2")
library("Glimma")
```

Load data

Replace *YOUR_COUNTS_DATA.csv* by your raw counts .csv file. If necessary, change the separation character to match your file.

```
countData <- read.csv('YOUR_COUNTS_DATA.csv', header = TRUE, sep = ",")
```

Replace *YOUR_METADATA.csv* by your metadata .csv file (with sample ids, and treatment names).

```
colData <- read.csv('YOUR_METADATA.csv', header = TRUE, sep = ",",)
colnames(colData)<-c("id","group")
```

DESeq2

Construct DESeqDataset object

```
dds <- DESeqDataSetFromMatrix(countData=countData,
                              colData=colData,
                              design=~group,
                              tidy = TRUE)
```

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

Run DESeq function

```
dds <- DESeq(dds)
```

```
## estimating size factors
```

```
## estimating dispersions
```

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

```
## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing
```

Extract results in a new dataframe. Make sure to change the *treatment* and *control* to your own.

```
res <- results(dds, contrast=c("group", "treatment", "control"))
```

It is possible to already export these results as a .csv file for further analysis in other programs.

```
write.csv(as.data.frame(res), file="results.csv")
```

The file would look like this.

```
head(read.csv("results.csv"))
```

```
##                         X   baseMean log2FoldChange     lfcSE       stat
## 1  Afi_v2_s189G000110.2 194548.636     4.81609140 0.4254627 11.3196548
## 2   Afi_v2_s74G000210.2  45016.653     0.52233627 0.1926285  2.7116249
## 3   Afi_v2_s48G000990.2  33289.799     0.08425995 0.3600550  0.2340197
## 4   Afi_v2_s20G000170.2  10534.156     5.86013452 0.7551303  7.7604283
## 5   Afi_v2_s15G003180.3  35236.665    -0.85785455 0.6148753 -1.3951683
## 6 Afi_v2_s3215G000080.1   7514.873     5.39865423 0.5780653  9.3391769
##          pvalue         padj
## 1 1.048841e-29 1.967392e-26
## 2 6.695432e-03 1.507700e-02
## 3 8.149697e-01 9.062854e-01
## 4 8.464304e-15 6.620692e-13
## 5 1.629651e-01 2.733138e-01
## 6 9.708590e-21 2.731674e-18
```

Sanity summary check.

```
summary(res)
```

```
##
## out of 19481 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)       : 829, 4.3%
## LFC < 0 (down)     : 8054, 41%
## outliers [1]       : 775, 4%
## low counts [2]     : 1824, 9.4%
## (mean count < 0)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

Head the results by padj.

```
head(res[order(res$padj),], n=5)
```

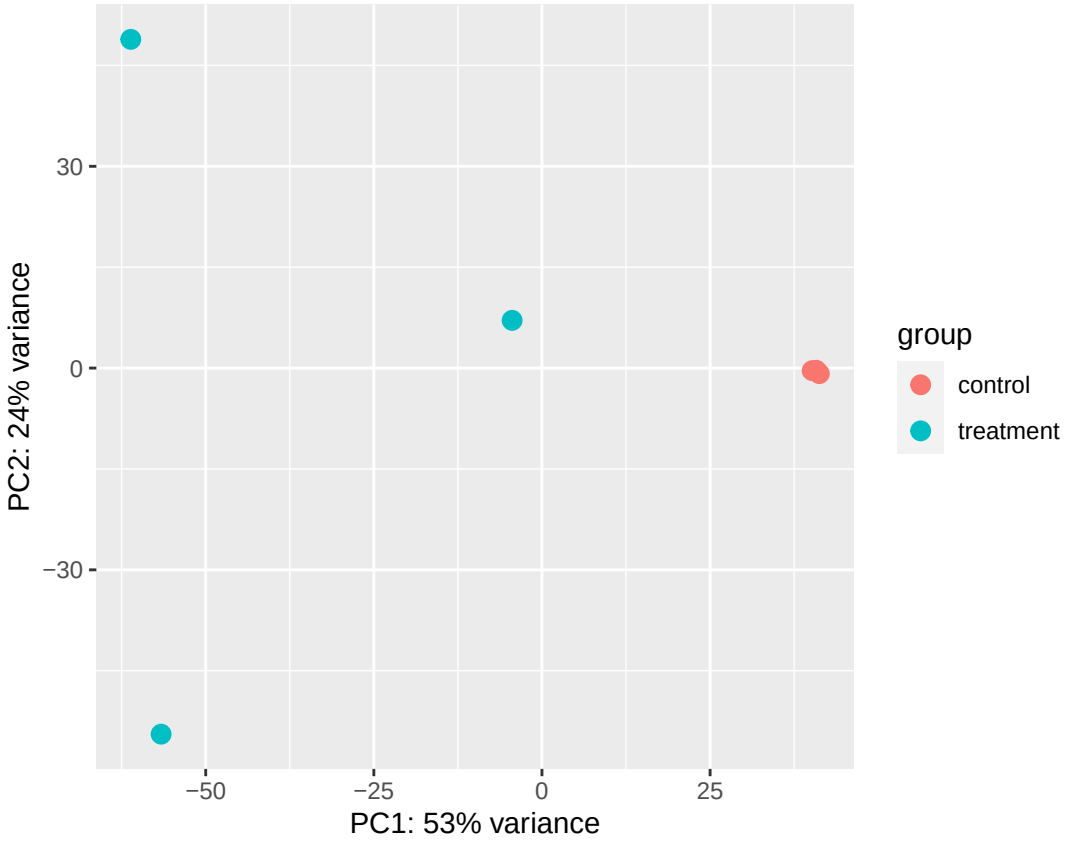
```
## log2 fold change (MLE): group treatment vs control
## Wald test p-value: group treatment vs control
## DataFrame with 5 rows and 6 columns
##                         baseMean log2FoldChange     lfcSE      stat      pvalue
##                        <numeric>      <numeric> <numeric> <numeric>   <numeric>
## Afi_v2_s28G002430.2     1410.977        6.79005  0.468698   14.4871 1.46291e-47
## Afi_v2_s2124G000060.1   1967.290        1.87511  0.134299   13.9623 2.64931e-44
## Afi_v2_s12G001740.3      733.392       -6.90966  0.526949  -13.1126 2.78983e-39
## Afi_v2_s1G001440.1       714.523       -6.91911  0.529875  -13.0580 5.72183e-39
## Afi_v2_s5G008650.1       876.314       -6.83197  0.561754  -12.1619 4.96181e-34
##                              padj
##                         <numeric>
## Afi_v2_s28G002430.2   2.46969e-43
## Afi_v2_s2124G000060.1 2.23628e-40
## Afi_v2_s12G001740.3   1.56993e-35
## Afi_v2_s1G001440.1    2.41490e-35
## Afi_v2_s5G008650.1    1.67531e-30
```

PCA

Using the DESEQ2 plotPCA

```
vsdata <- vst(dds, blind=FALSE, fitType = c("local"))
```

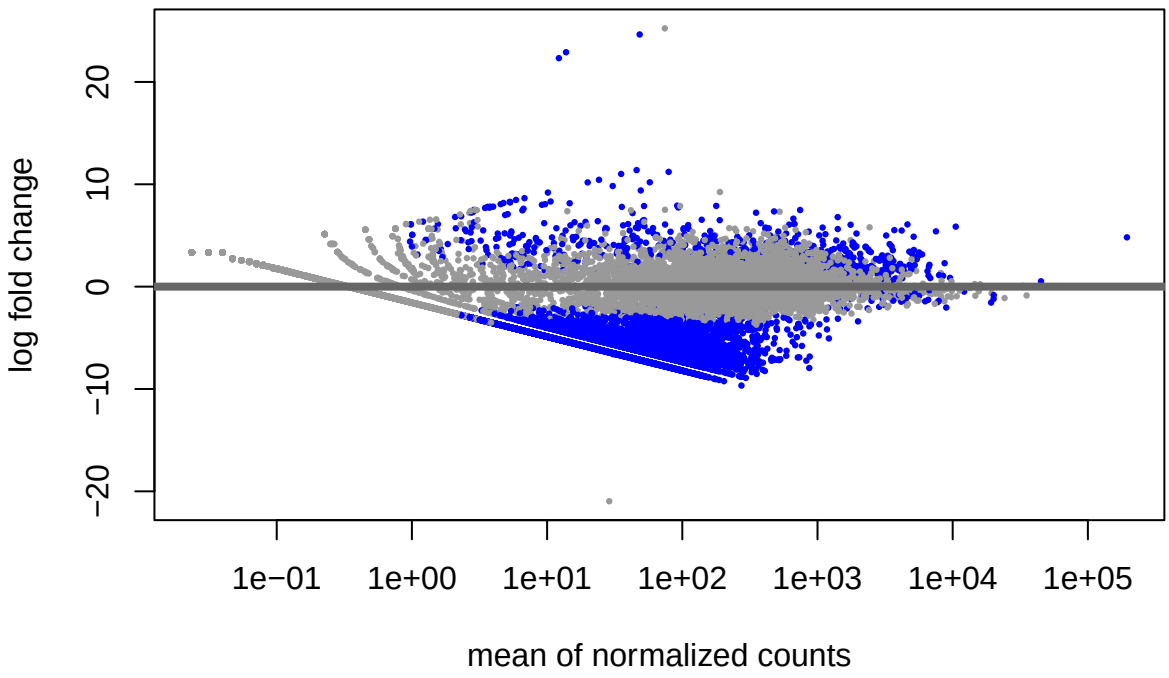
```
plotPCA(vsdata, intgroup="group")
```



Plot gene expression

The boundaries of the plot are set to the minimum and maximum log2FoldChange.

```
plotMA(res, ylim= c(min(res$log2FoldChange, na.rm=TRUE), max(res$log2FoldChange, na.rm=TRUE)))
```



Further down, it will be possible to interact and explore the data behind this figure (here static) with the Glimma html interface.

Glimma

The Glimma tool requires the data to be reshaped in a few dataframes so that they can be correctly plotted. First of all, an *anno* dataframe that will contain the data and respective annotations of the interactive plot.

```
anno <- as.data.frame(res)
```

```
anno$GeneID <- row.names.data.frame(anno) # Add GeneID as a column instead of just the rownames so that
```

Connect GeneNames to GeneIDs

As so far only the GeneIDs have been included in the datasets (e.g. *Afi_v2_s28G002430.2*), it would be more convenient to also add the respective GeneNames (e.g. *'transcription factor (bHLH)'*) and descriptions (e.g. *'RNA biosynthesis.transcriptional regulation'*).

In this case the respective names and descriptions are located in an external .csv file *GENENAMES.csv* , preferably with the following columns: GeneID, GeneName, GeneDescription.

```
gene_names <- read.csv('GENENAMES.csv', header = TRUE, sep = ",",)
```

```
anno<- cbind(anno, GeneName = "") # add a new empty column for Gene Names in the anno dataframe
anno<- cbind(anno, GeneDescription = "") # add a new empty column for Gene Descriptions in the anno dat
```

In order to connect the *GeneNames* and *GeneDescriptions* to the respectively correspondent *GeneIDs* in the *anno* dataframe, we use the following loop.

```
i=0
p=0

for (i in 1:nrow(anno)){

  anno[i,8] <- gene_names[(which(gene_names$GeneID == anno[i,7])) # the which() function gives the roum
                          ,3] # 3 = GeneID column  of the gene_names file

  anno[i,9] <- gene_names[(which(gene_names$GeneID == anno[i,7]))
                          ,4] # 4 = DESCRIPTION column  of the gene_names file

  #It should take a minute or so.
}
```

glMDPlot

A few more subsets have to be defined in order to generate the interactive Glimma plot.

```
padj <- res[, 'padj'] # subset the padj column from the results
```

```
padj[ is.na(padj) ] <- 1 # NA values are replaced by '1' values.
```

```
status <- as.numeric(padj < 0.01) # the status file gathers together all the padj values smaller than 0

# note: it is possible to modify the threshold of the padj filter. Here, only the the datapoints with a
```

Here the padj threshold is set to 0.01 (all the datapoints below this threshold will appear in red on the graph, all the others will stay grey)

```
groups <- colData(dds)[ ,'group'] # which experimental factors to take along
```

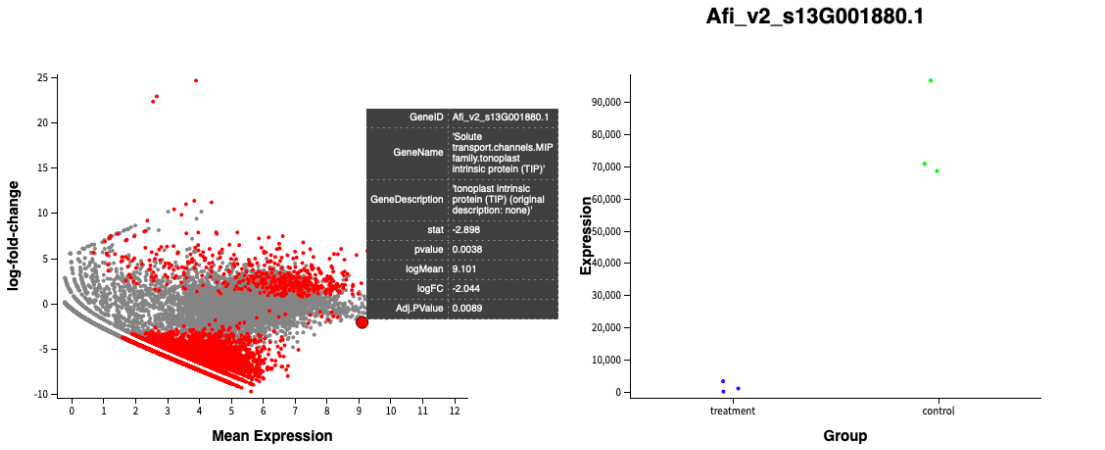
```
colors <- c('blue', 'blue', 'blue', 'green', 'green', 'green') #  color of the triplicat points. Here t
```

```
display <- c("GeneID","GeneName","GeneDescription", "stat","pvalue") # which annotation (from mcols(dds
```

Finally, we can generate the plot.

```
glMDPlot(res,
         status=status,
         counts=counts(dds),
         samples=colnames(dds),
         sample.cols=colors,
         anno=anno,
```

```
        groups=groups,
        display.columns=display,
        html='plot_Glimma', # output name
        launch= FALSE) # if TRUE, will launch the html interactive plot right after its generation.
```



Search:

GeneID	GeneName	GeneDescription	stat	pvalue	logMean	logFC	Adj.PValue
Afi_v2_s28G002430.2	'RNA biosynthesis.transcriptional regulation.transcription factor (bHLH)'	'transcription factor (bHLH) (original description: none)'	14.49	1.463e-47	7.252	6.79	2.47e-43
Afi_v2_s2124G000060.1	'not assigned.not annotated'	' no hits & (original description: none)'	13.96	2.649e-44	7.585	1.875	2.236e-40
Afi_v2_s12G001740.3	'Lipid metabolism.fatty acid biosynthesis.plastidial fatty acid synthase (ptFAS) system.fatty acid chain termination.oleoyl-ACP thioesterase'	'oleoyl-ACP thioesterase (original description: none)'	-13.11	2.79e-39	6.598	-6.91	1.57e-35

The output objecti is a folder containing the html interactive plot file.
On the plot, it is possible to browse through the datapoints, search for a specific gene, sort by foldchange or padj, etc..