Differential expression analysis of MOV10 Gene Perturbation on HEK293F mice cells

Joshua Barrs

12/8/2021

Abstract

In this project, RNA sequencing of mouse HEK293F cells are performed in order to conduct differential expression (DE) analysis to determine the impact that MOV10 gene perturbation has on other genes. By investigating this relationship, a conclusion can be made with respect to which processes are affected/correlated with MOV10 expression. We seek to identify patterns of expression resulting from MOV10 gene perturbation (addition or removal of MOV10 expression). Week seek to identify the genes whose expression levels change as MOV10 expression levels alter within the cell. MOV10 is an RNA helicase protein. It facilitates microRNA-mediated translation of certain RNAs in the brain and increases expression of other RNAs by blocking the completion of AGO2 function [1]. The data presented in this project is taken from the study by [1], in which the authors investigate how MOV10 and FMRP, Fragile X Syndrome Mental Retardation Protein, associate with and regulate the translation of a certain subset of RNAs. The RNA sequencing analysis code presented in this project was compiled following the RNA Sequencing tutorial created by [2].

Author's Note I would like to credit the DGE workshop tutorial found at [2], as this project was constructed following their tutorial made publicly available to the academic community

Materials and Methods

Data Set

For the purpose of this project, the data used in the DE analysis is taken from the study published by [2]. The dataset includes a count matrix of RNA-sequenced genes taken from mice cells. In the study by [1], HEK293F mice cells were taken and subsequently transfected with either a MOV10 transgene in order to promote MOV10 overexpression, siRNA (short-interfering RNA) to knockdown MOV10 expression levels, or an irrelevant knockdown (KD) protein to keep MOV10 at its regular expression levels. For the purposes of this project, the cells treated with the irrelevant KD protein are treated as the control group [2]. There were two replicate (sample) groups for the cells treated with knockdown siRNA, three replicates (sample groups) for the cells treated with the MOV10 transgene (overexpression), and three replicates for the cells treated with irrelevant knockdown. The data analyzed is presented below. Namely, it consists of the sequence read counts of each gene that were subjected to the three treatments described above. Higher number of sequence counts indicate higher levels of expression of that gene within the replicate.

```r
data <- read.table("../data/Mov10_full_counts.txt", header=T, row.names=1)

meta <- read.table("../meta/Mov10_full_meta.txt", header=T, row.names=1)

head(data)
```

```
##              Mov10_kd_2 Mov10_kd_3 Mov10_oe_1 Mov10_oe_2 Mov10_oe_3 Irrel_kd_1
## 1/2-SBSRNA4          57         41         64         55         38         45
## A1BG                 71         40        100         81         41         77
## A1BG-AS1            256        177        220        189        107        213
## A1CF                  0          1          1          0          0          0
## A2LD1               146         81        138        125         52         91
## A2M                  10          9          2          5          2          9
##              Irrel_kd_2 Irrel_kd_3
## 1/2-SBSRNA4          31         39
## A1BG                 58         40
## A1BG-AS1            172        126
## A1CF                  0          0
## A2LD1                80         50
## A2M                   8          4
```

Count Normalization

Differential expression analysis was then conducted by a series of steps, most of which included methods provided by the DESeq2 package, which is part of the Bioconductor Bioinformatics package network. Once the RNA sequencing data was loaded in (above), differential expression analysis of the RNA-sequenced data could begin. In the R snippet presented below, a DESeq2 data object is created from our RNA-sequenced data. We then obtain the mapped read counts for each gene and then normalize these mapped read counts in order to eliminate extraneous related factors. After the normalization, the mapped read counts of the gene will be directly proportional to the RNA expression levels.

```
# Let us now create our DeSeq2 Dataset object so we can run Prinicipal Component Analysis (PCA Analysis

## Create DESeq2Dataset object
dds <- DESeqDataSetFromMatrix(countData = data, colData = meta, design = ~ sampletype)
```

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

```
# Now that we have created our DESeq2Dataset object, let us normalize the count values by calling the e
dds <- estimateSizeFactors(dds)

# Let's make sure that the normalized factors are in our data frame
sizeFactors(dds)
```

```
## Mov10_kd_2 Mov10_kd_3 Mov10_oe_1 Mov10_oe_2 Mov10_oe_3 Irrel_kd_1 Irrel_kd_2
##  1.5646728  0.9351760  1.2016082  1.1205912  0.6534987  1.1224020  0.9625632
## Irrel_kd_3
##  0.7477715
```

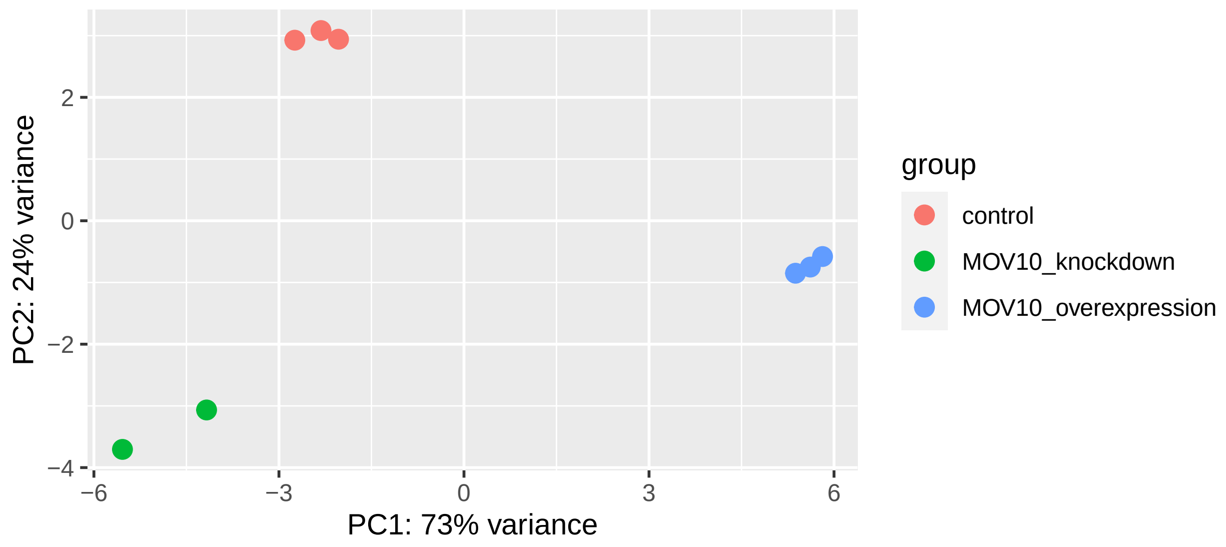
```
# Let us compute the normalized RNA sequence counts
normalized_counts <- counts(dds, normalized=TRUE)

head(normalized_counts) # output the normalized counts
```

```
##              Mov10_kd_2 Mov10_kd_3 Mov10_oe_1 Mov10_oe_2 Mov10_oe_3 Irrel_kd_1
## 1/2-SBSRNA4   36.429341  43.842016  53.261952   49.08123   58.14855  40.092585
## A1BG         45.376898  42.772698  83.221799   72.28327   62.73922  68.602869
## A1BG-AS1    163.612478 189.269189 183.087958  168.66097  163.73407 189.771571
## A1CF          0.000000   1.069317   0.832218    0.00000    0.00000   0.000000
## A2LD1        93.310241  86.614714 114.846083  111.54826   79.57170  81.076117
## A2M           6.391112   9.623857   1.664436    4.46193    3.06045   8.018517
##             Irrel_kd_2 Irrel_kd_3
## 1/2-SBSRNA4  32.205676  52.154968
## A1BG         60.255781  53.492275
## A1BG-AS1    178.689557 168.500666
## A1CF          0.000000   0.000000
## A2LD1        83.111422  66.865344
## A2M           8.311142   5.349227
```

Principal Component Analysis (PCA) of Normalized Count Data

After read count normalization, principal component analysis (PCA) of our normalized read counts were performed in order to identify factors/variables that account for the variation in our sequenced data. The PCA plot is presented below:

```r
#rlog() function will transform the normalized counts

rld <- rlog(dds, blind=TRUE)

# Now that we have moderated the variance, let us run the PCA on our normalized RNA sequence counts

plotPCA(rld, intgroup="sampletype")
```



As we can see from the graph, the replicate (sample) groups are indeed organized according to their treatment effects, indicating that the treatment effect (overexpression, knockdown, or irrelevant siRNA) serves as a key factor for variability within the data set. With this conclusion, we can further conclude that the genes that differ in their expression levels do so according to which treatment they were subjected[2].

DE Analysis using built-in DESeq function

After verifying our data in PCA, differential expression analysis is conducted using the DESeq() function, available from the DESeq2 package downloaded earlier in the project. The DESeq() function performs all steps of the Differential Expression Analysis, including dispersion estimation, gene-wise dispersion estimation, model fitting and model testing. The function then outputs a DESeq data object, which contains the genes in study in addition to several attributes of the genes, including log2 shrinkage estimates.

```
## Run analysis
dds <- DESeq(dds)
```

```
## using pre-existing size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates
## fitting model and testing
```

```
results(dds)
```

```
## log2 fold change (MLE): sampletype MOV10 overexpression vs control
## Wald test p-value: sampletype MOV10 overexpression vs control
## DataFrame with 23368 rows and 6 columns
##                 baseMean log2FoldChange     lfcSE      stat     pvalue      padj
##                <numeric>      <numeric> <numeric> <numeric>  <numeric> <numeric>
## 1/2-SBSRNA4    45.652040       0.373730  0.266671  1.401464  0.1610752 0.2750250
## A1BG           61.093102       0.264759  0.225421  1.174510  0.2401909 0.3716156
## A1BG-AS1      175.665807      -0.057622  0.139210 -0.413922  0.6789312 0.7840468
## A1CF            0.237692       0.920672  3.536275  0.260351  0.7945932        NA
## A2LD1          89.617985       0.419733  0.197240  2.128033  0.0333343 0.0774672
## ...                  ...            ...       ...       ...        ...       ...
## ZYG11B       2973.949477     -0.0675050 0.0584964 -1.154002 0.24849951 0.3813641
## ZYX          2933.105330     -0.0632865 0.0709360 -0.892163 0.37230543 0.5157840
## ZZEF1        2132.254272     -0.1580217 0.0698630 -2.261879 0.02370489 0.0582162
## ZZZ3         2215.883805     -0.1654566 0.0625671 -2.644468 0.00818194 0.0237935
## tAKR            0.343415     -1.0028891 2.9145295 -0.344100 0.73077122        NA
```

Log2 Fold Change (LFC) Shrinkage

After obtaining our DE result, we perform log 2 fold change shrinkage on our genes of interest. The main purpose of log 2 fold change shrinkage is to generate more accurate estimates of our log 2 fold change (LFC) values. In essence, genes with either low count values or high dispersion values have their LFC estimates shrunken toward 0. This way, the log 2 fold change estimates of genes with low counts or high dispersion will have more likely/lower LFC estimates [2].

We also conduct a Wald-Walfowitz test for statistical analysis with the function call below. In essence, we create a null hypothesis (in our case, the null hypothesis, or H0, is that there is no differential expression between genes from our two treatment groups) and then perform the test, which will output a p-value, indicating the likelihood of obtaining an output value as extreme as the observed value in our data set. Two separate lfcShrink() function calls are made below: one for each Walfowitz test.

Test 1: We are contrasting MOV10 Overexpression (MOV10_OE) against the control group (genes treated with irrelevant siRNA in the experiment which remain unaffected with respect to MOV10 expression) Test2: We are contrasting MOV10 gene knockdown (MOV10_KD) against the control group (genes treated with irrelevant siRNA which remain unaffected with respect to MOV10 expression)

```
# Let us now perform log2 shrinking (LFC)
res_tableOE <- lfcShrink(dds, coef = 3)
```

```
## using 'apeglm' for LFC shrinkage. If used in published research, please cite:
##     Zhu, A., Ibrahim, J.G., Love, M.I. (2018) Heavy-tailed prior distributions for
##     sequence count data: removing the noise and preserving large differences.
##     Bioinformatics. https://doi.org/10.1093/bioinformatics/bty895
```

```
head(res_tableOE)
```

```
## log2 fold change (MAP): sampletype MOV10 overexpression vs control
## Wald test p-value: sampletype MOV10 overexpression vs control
## DataFrame with 6 rows and 5 columns
##                baseMean log2FoldChange     lfcSE    pvalue      padj
##               <numeric>      <numeric> <numeric> <numeric> <numeric>
## 1/2-SBSRNA4  45.652040     0.16858121  0.209275 0.1610752 0.2750250
## A1BG         61.093102     0.13623770  0.176713 0.2401909 0.3716156
## A1BG-AS1    175.665807    -0.04016953  0.116916 0.6789312 0.7840468
## A1CF          0.237692     0.00626568  0.210057 0.7945932        NA
## A2LD1        89.617985     0.29007261  0.195380 0.0333343 0.0774672
## A2M           5.860084    -0.09623512  0.233587 0.1057823 0.1976308
```

```r
# Let's perform a Wald test and DE analysis for the MOV10_KD (knockdown) vs. the control group

# LFC Shrinking for the KnockDown genes
res_tableKD <- lfcShrink(dds, coef = 2)
```

```
## using 'apeglm' for LFC shrinkage. If used in published research, please cite:
##     Zhu, A., Ibrahim, J.G., Love, M.I. (2018) Heavy-tailed prior distributions for
##     sequence count data: removing the noise and preserving large differences.
##     Bioinformatics. https://doi.org/10.1093/bioinformatics/bty895
```

```r
head(res_tableKD)
```

```
## log2 fold change (MAP): sampletype MOV10 knockdown vs control
## Wald test p-value: sampletype MOV10 knockdown vs control
## DataFrame with 6 rows and 5 columns
##                baseMean log2FoldChange     lfcSE    pvalue      padj
##               <numeric>      <numeric> <numeric> <numeric> <numeric>
## 1/2-SBSRNA4  45.652040    -0.01256480  0.157710 0.8775736  0.930943
## A1BG         61.093102    -0.21591616  0.235252 0.0709078  0.162958
## A1BG-AS1    175.665807    -0.02139607  0.117619 0.8133011  0.892483
## A1CF          0.237692     0.00565633  0.184926 0.7362778        NA
## A2LD1        89.617985     0.10081633  0.157521 0.3105785  0.478999
## A2M           5.860084     0.00620226  0.179621 0.8880060  0.937464
```

What is outputted is a DESeqresults object (very similar to a dataframe) that contains the log2 shrinkage estimates in addition to the Walfowtiz test output (including p-values for each gene).

Extracting Significant DE Genes

Once the DESeqresults object has been outputted, we begin the work of extracting out the signficant DE genes. For the purposes of the project, "significant" with respect to the genes indicates that the gene had a significant p-value of < 0.05, allowing us to reject the null hypothesis (H0) and conclude that the genes from the two treatment groups in the Walfowtiz test are indeed differentially expressed. Because our DESeqresults object (res_tableOE and res_tableKD) contain all of the genes from the test (even the non-significant ones), we must manually parse out only the genes that are significant in order to determine which are differentially expressed (DE). We do these steps below:

```r
### Set thresholds
padj.cutoff <- 0.05
lfc.cutoff <- 0.58
```

```r
# Create a tibble from our res_tableOE
res_tableOE_tb <- res_tableOE %>%
  data.frame() %>%
  rownames_to_column(var="gene") %>%
  as_tibble()


sigOE <- res_tableOE_tb %>%
  filter(padj < padj.cutoff & abs(log2FoldChange) > lfc.cutoff)


sigOE
```

```
## # A tibble: 954 x 6
##    gene    baseMean log2FoldChange  lfcSE   pvalue     padj
##    <chr>      <dbl>          <dbl>  <dbl>    <dbl>    <dbl>
##  1 A4GALT     64.5           0.902 0.231  6.36e- 6 4.31e- 5
##  2 ABCA1     108.            0.907 0.187  8.54e- 8 8.69e- 7
##  3 ABCA6       6.12         -2.09  0.953  1.42e- 3 5.21e- 3
##  4 ABCA9      11.2          -2.06  0.649  8.89e- 5 4.49e- 4
##  5 ABCB4     137.            0.642 0.166  1.02e- 5 6.59e- 5
##  6 ABHD14B   494.            0.785 0.106  1.04e-14 3.89e-13
##  7 ABI3BP     40.8           0.754 0.327  1.20e- 3 4.50e- 3
##  8 ABL2     1756.            0.691 0.0661 1.22e-26 1.99e-24
##  9 ACADVL   1325.            0.676 0.0821 1.82e-17 1.01e-15
## 10 ACAN       19.6           1.36  0.458  1.46e- 4 6.99e- 4
## # ... with 944 more rows
```

```r
res_tableKD_tb <- res_tableKD %>%
  data.frame() %>%
  rownames_to_column(var="gene") %>%
  as_tibble()

sigKD <- res_tableKD_tb %>%
  filter(padj < padj.cutoff & abs(log2FoldChange) > lfc.cutoff)


sigKD
```

```
## # A tibble: 768 x 6
##    gene   baseMean log2FoldChange  lfcSE   pvalue     padj
##    <chr>     <dbl>          <dbl>  <dbl>    <dbl>    <dbl>
##  1 ABCA8      43.1         -0.715 0.363  2.20e- 3 1.00e- 2
##  2 ABCB10   1483.          -0.767 0.0814 3.04e-22 4.64e-20
##  3 ABCD2      43.9         -1.19  0.353  3.79e- 5 3.18e- 4
##  4 ABCG1     112.          -1.34  0.216  3.98e-11 1.32e- 9
##  5 ACCS      118.          -0.617 0.208  2.27e- 4 1.48e- 3
##  6 ACHE       88.3          0.873 0.250  2.79e- 5 2.43e- 4
##  7 ACOXL      92.1         -1.64  0.229  4.34e-14 2.42e-12
##  8 ACPP       88.5         -0.801 0.241  5.40e- 5 4.34e- 4
##  9 ACRC      103.           1.30  0.199  4.72e-12 1.85e-10
## 10 ACSBG1     20.1          1.51  0.488  1.05e- 4 7.64e- 4
## # ... with 758 more rows
```

In essence, I create a p-value cutoff value of 0.05 and a log2 fold change cutoff of 0.58. Thus, only the genes from my DESeqresults objects with p-values less than padj.cutoff (0.05) and greater than lfc.cutoff (0.58) will be extracted into a separate data frame for further analysis and visualization.

Results and Discussion

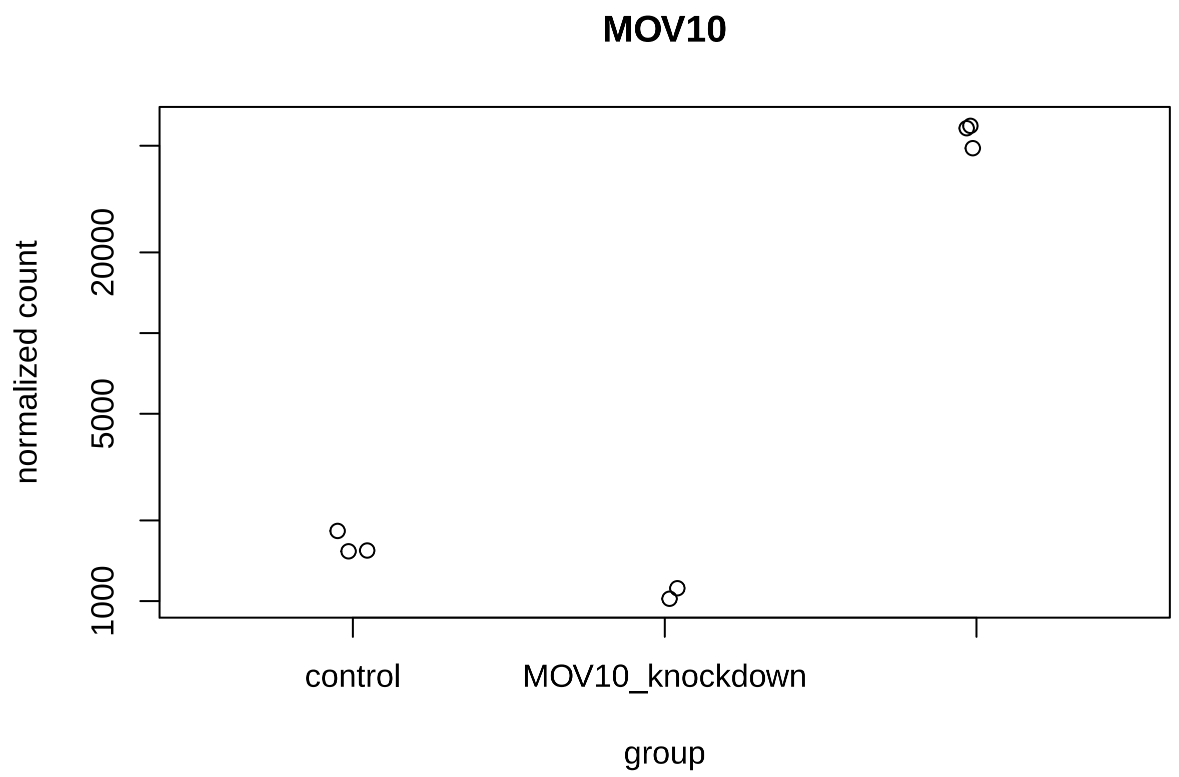
Differentially Expressed Gene Counts Plots

As the purpose of this project is DE analysis, the primary output/results of the data processing involve a series of plots that visually demonstrate the specific genes that were differentially expressed as a result of MOV10 gene perturbation in the mice cells. By investigating the following plots, we are able to determine the specific genes differentially expressed in addition to the degree with which they are differentially expressed. In the plot below, we plot the top 20 differentially expressed genes in our dataset; however, we can easily adapt the figure to include the top 30 DE genes or to include a random number of genes we wish included in the plot.

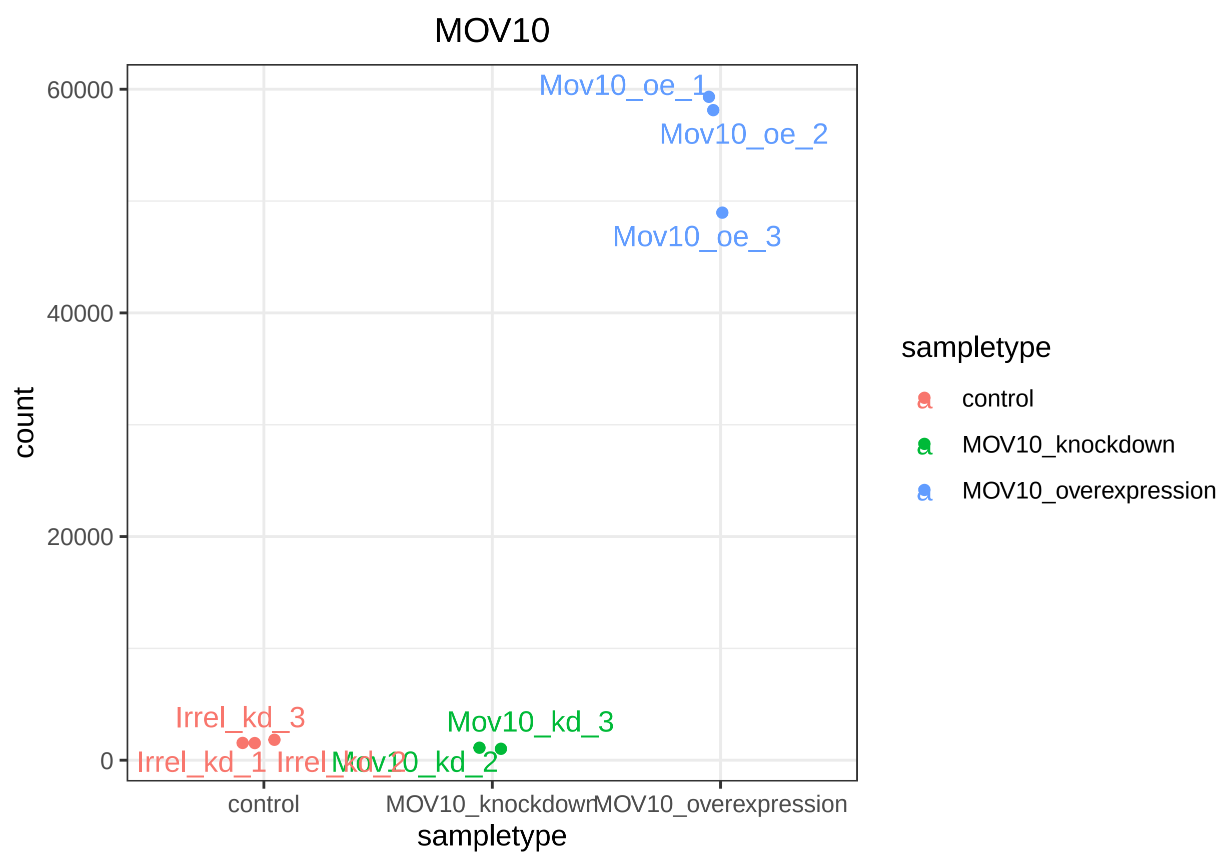
In the first plot, we simply plot the expression levels of only the MOV10 gene with respect to which treatment the MOV10 gene was subjected.

In the last plot, we visualize the top 20 differentially expressed genes and their log10 normalized read counts, grouped according to their treatment effects. We are then able to conclude that these DE genes' expression levels are a byproduct of MOV10 gene perturbation within the cells, indicating a correlation between MOV10 expression levels and expression levels of these genes seen in the plot.

```r
# Create tibbles including row names from meta and normalized_counts objects created earlier
mov10_meta <- meta %>%
  rownames_to_column(var="samplename") %>%
  as_tibble()

normalized_counts <- normalized_counts %>%
  data.frame() %>%
  rownames_to_column(var="gene") %>%
  as_tibble()

# Plot expression for single gene
plotCounts(dds, gene="MOV10", intgroup="sampletype")
```



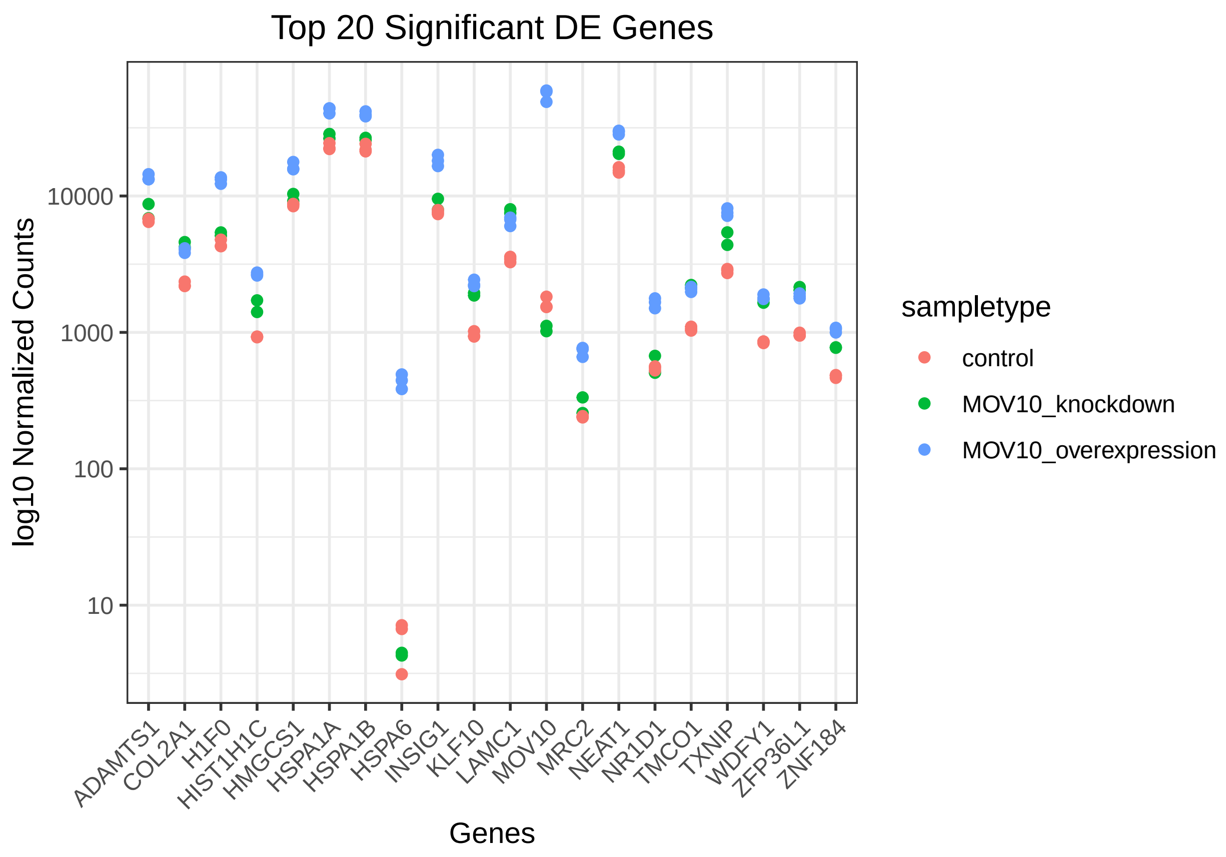
```
# Save plotcounts to a data frame object
d <- plotCounts(dds, gene="MOV10", intgroup="sampletype", returnData=TRUE)

# Plotting the MOV10 normalized counts, using the samplenames (rownames of d as labels)
ggplot(d, aes(x = sampletype, y = count, color = sampletype)) +
  geom_point(position=position_jitter(w = 0.1,h = 0)) +
  geom_text_repel(aes(label = rownames(d))) +
  theme_bw() +
  ggtitle("MOV10") +
  theme(plot.title = element_text(hjust = 0.5))
```



```
## Order results by padj values
top20_sigOE_genes <- res_tableOE_tb %>%
  arrange(padj) %>%      #Arrange rows by padj values
  pull(gene) %>%         #Extract character vector of ordered genes
  head(n=20)         #Extract the first 20 genes

## normalized counts for top 20 significant genes
top20_sigOE_norm <- normalized_counts %>%
  filter(gene %in% top20_sigOE_genes)

# Gathering the columns to have normalized counts to a single column
gathered_top20_sigOE <- top20_sigOE_norm %>%
  gather(colnames(top20_sigOE_norm)[2:9], key = "samplename", value = "normalized_counts")

## check the column header in the "gathered" data frame
View(gathered_top20_sigOE)

gathered_top20_sigOE <- inner_join(mov10_meta, gathered_top20_sigOE)
```

```
## Joining, by = "samplename"
```

```
# Let's plot now!
## plot using ggplot2
ggplot(gathered_top20_sigOE) +
  geom_point(aes(x = gene, y = normalized_counts, color = sampletype)) +
  scale_y_log10() +
  xlab("Genes") +
  ylab("log10 Normalized Counts") +
  ggtitle("Top 20 Significant DE Genes") +
  theme_bw() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  theme(plot.title = element_text(hjust = 0.5))
```



References

1. Kenny Phillip&nbsp;J, Zhou H, Kim M, Skariah G, Khetani Radhika&nbsp;S, Drnevich J, Arcila M, Kosik Kenneth&nbsp;S, Ceman S. 2014 MOV10 and FMRP regulate AGO2 association with MicroRNA recognition elements. *Cell Reports* **9**, 1729–1741. (doi:10.1016/j.celrep.2014.10.054)

2. Mistry M, Khetani R, Piper M. 2017 DGE workshop. *Github*.