Day 7 - Introduction to Gene Differential Expression Analysis using DESeq2

Authors:

Jacob Stanley (jacob.stanley@colorado.edu)

Edited and updated by:
Daniel Ramirez (daniel.ramirezhernandez@colorado.edu) - 2022
Rutendo Sigauke (rutendo.sigauke@colorado.edu) - 2023

Additional DESeq2 resources:

https://bioconductor.org/packages/release/bioc/html/DESeq2.html
https://bioconductor.org/packages/release/bioc/vignettes/DESeq2/inst/doc/DESeq2.html

Introduction:

Here will use those gene counts tables as input for the software DESeq2 to answer the question: What genes are statistically significantly changed upon an experimental condition? In particular, we will explore a dataset from a real experiment published (Andrysik 2017 et al. doi: 10.1101/gr.220533.117). You will use a gene count table that we already prepared for you, from an experiment where Human colon cancer cells (HCT116) were treated with either the vehicle DMSO, or with the p53-activator drug Nutlin.

The purpose of DESeq2 is to identify which genomic loci demonstrate a statistically significant difference in expression level between two or more conditions (referred to as “gene differential expression analysis”). It does so by modeling the variance in expression level across the full range of baseline expression levels present in the data and determines if the differential expression level for each locus is significantly greater than this variance. DESeq2 takes as an input the unnormalized count values for each (non-overlapping) loci in each sample. We recommend that you use featureCounts() to compute count values. DESeq2 performs best when provided multiple replicates per experimental condition (preferably 5+ replicates), in order to get an accurate estimation of within condition variance. DESeq2 is only to be used for non-overlapping, unique genomic loci. If one’s aim is to compute differential expression of transcripts, DESeq2 is not appropriate.

Note: All commands are executed within the R environment. We will be executing them manually, from the R command line, but they can also be compiled into a single script to be executed together.

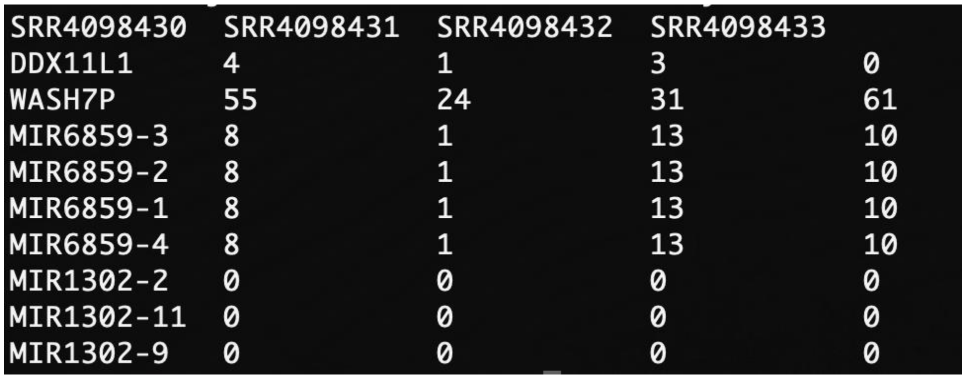
--- FILES ARE IN THE DATA FOLDER ON GITHUB FOR DAY7---

Below are the featureCounts output files as well as the condition table we will be using. Get these files onto your own local machine. We will use your local machine installation of RStudio and use these files as inputs.

/path/to/day07/data

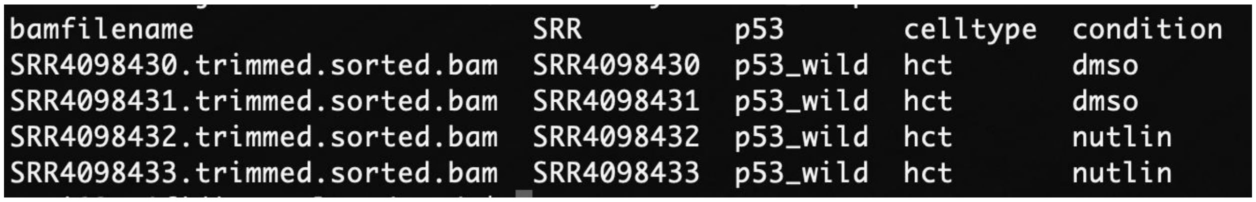
Andrysik2017_counts.tsv
Andrysik2017_samples.tsv

Shown below is the top of the file **Andrysik2017_counts.tsv**, showing the raw unnormalized read counts across the first genes of the annotation file.



```
SRR4098430  SRR4098431  SRR4098432  SRR4098433
DDX11L1     4           1           3           0
WASH7P      55          24          31          61
MIR6859-3   8           1           13          10
MIR6859-2   8           1           13          10
MIR6859-1   8           1           13          10
MIR6859-4   8           1           13          10
MIR1302-2   0           0           0           0
MIR1302-11  0           0           0           0
MIR1302-9   0           0           0           0
```

Shown below is the contents of the **Andrysik2017_samples.tsv** file. It has four columns, each with information regarding each of the RNA-seq datasets. This information was used by featureCounts, and will be used again by DESeq2 to know which datasets are replicates of each other, and against what other replicates to make any comparison.



```
bamfilename                    SRR         p53       celltype  condition
SRR4098430.trimmed.sorted.bam  SRR4098430  p53_wild  hct       dmso
SRR4098431.trimmed.sorted.bam  SRR4098431  p53_wild  hct       dmso
SRR4098432.trimmed.sorted.bam  SRR4098432  p53_wild  hct       nutlin
SRR4098433.trimmed.sorted.bam  SRR4098433  p53_wild  hct       nutlin
```

--- USING DESEQ2 WITHIN RSTUDIO ---

We will use DESeq2 with the RStudio on your local computer. First, we need to tell RStudio to load the DESeq2 library (if you have not yet installed DESeq2, let a class helper know. Be aware though that *installing DESeq2 takes some time* as it also needs several dependencies installed as well). Loading DESeq2 will prompt some red messages, and they should end with R normal “>” prompt symbol.

```
> library(DESeq2)
Loading required package: S4Vectors
Loading required package: stats4
Loading required package: BiocGenerics
Loading required package: parallel

Attaching package: ‘BiocGenerics’

The following objects are masked from ‘package:parallel’:
```

1. Load the two input files using **read.table()**:
   a. The conditions table contains the sample information.

```
> conditionsTableFile <- "~/Desktop/sr2023/day7/Andrysik2017_samples.tsv"
> conditionsTable <- read.table(conditionsTableFile,
+                               sep = "\t",
+                               header = TRUE)
```

```
> conditionsTable
                   bamfilename        SRR      p53 celltype condition
1 SRR4098430.trimmed.sorted.bam SRR4098430 p53_wild      hct      dmso
2 SRR4098431.trimmed.sorted.bam SRR4098431 p53_wild      hct      dmso
3 SRR4098432.trimmed.sorted.bam SRR4098432 p53_wild      hct    nutlin
4 SRR4098433.trimmed.sorted.bam SRR4098433 p53_wild      hct    nutlin
```

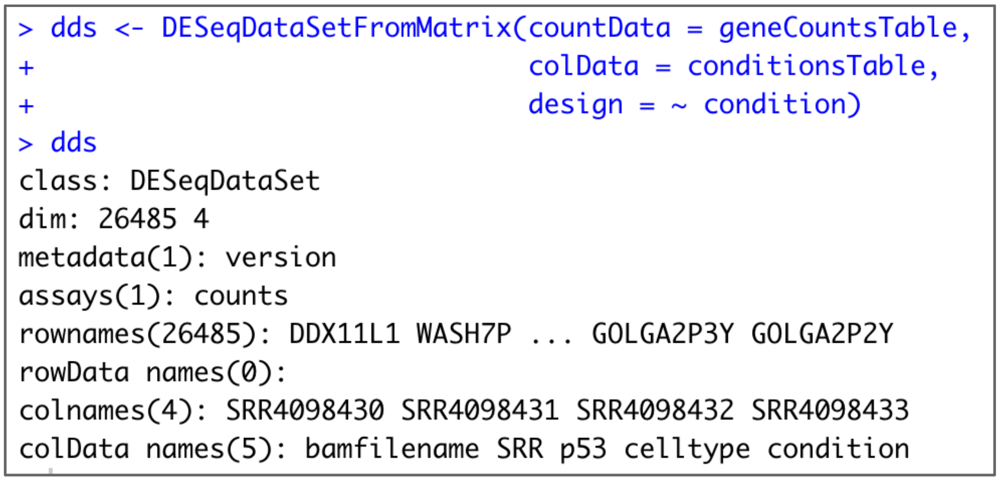
   b. Load the counts tables. The column names should match the names (SRR) in the conditions table

```
> geneCountsTableFile <- "~/Desktop/sr2023/day7/Andrysik2017_counts.tsv"
> geneCountsTable <- read.table(geneCountsTableFile,
+                               header = TRUE, sep = "\t", fill = TRUE,
+                               stringsAsFactors = FALSE, na.strings = "")
```

```
> head(geneCountsTable)
          SRR4098430 SRR4098431 SRR4098432 SRR4098433
DDX11L1            4          1          3          0
WASH7P            55         24         31         61
MIR6859-3          8          1         13         10
MIR6859-2          8          1         13         10
MIR6859-1          8          1         13         10
MIR6859-4          8          1         13         10
```

2. Next, load the two inputs onto DESeq2 using the following function. You can then type the variable **dds** and see some information of its contents, including its variable type, the number of genes that have counts (e.g. 26,485), and some of the gene and datasets labels.

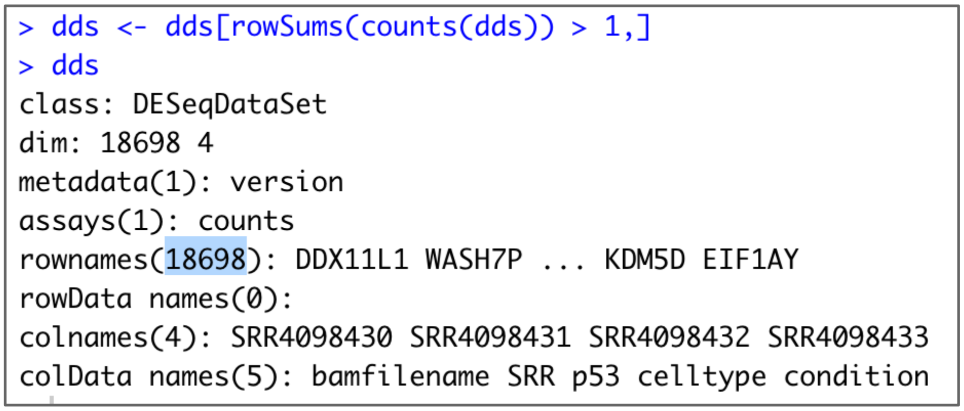
```
dds <- DESeqDataSetFromMatrix(countData = geneCountsTable,
                              colData = conditionsTable,
                              design = ~ condition)
```



```
> dds <- DESeqDataSetFromMatrix(countData = geneCountsTable,
+                               colData = conditionsTable,
+                               design = ~ condition)
> dds
class: DESeqDataSet
dim: 26485 4
metadata(1): version
assays(1): counts
rownames(26485): DDX11L1 WASH7P ... GOLGA2P3Y GOLGA2P2Y
rowData names(0):
colnames(4): SRR4098430 SRR4098431 SRR4098432 SRR4098433
colData names(5): bamfilename SRR p53 celltype condition
```

Optionally, you can remove all the gene entries that have low counts, such as those gene entries that have mostly zero counts. If you do not remove them, DESeq2 will automatically remove them internally while doing its calculations. Notice that printing the **dds** variable again gives us a smaller number of genes with counts (e.g. 18,698).

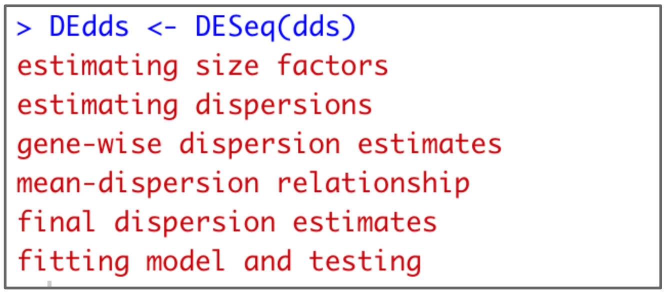
```
dds <- dds[rowSums(counts(dds)) > 1,]
```



```
> dds <- dds[rowSums(counts(dds)) > 1,]
> dds
class: DESeqDataSet
dim: 18698 4
metadata(1): version
assays(1): counts
rownames(18698): DDX11L1 WASH7P ... KDM5D EIF1AY
rowData names(0):
colnames(4): SRR4098430 SRR4098431 SRR4098432 SRR4098433
colData names(5): bamfilename SRR p53 celltype condition
```

3. Run DESeq2's main function **DESeq** on the **dds** variable you created. DESeq2 will internally do several actions:
    a. it will estimate each dataset size scale factors,
    b. it will estimate dispersion,
    c. it will fit a generalized linear model, and
    d. it will calculate each gene's fold change.

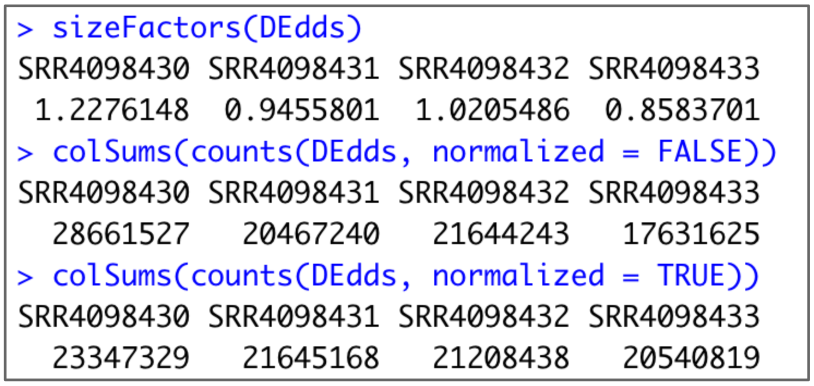
```
DEdds <- DESeq(dds)
```



```
> DEdds <- DESeq(dds)
estimating size factors
estimating dispersions
gene-wise dispersion estimates
mean-dispersion relationship
final dispersion estimates
fitting model and testing
```

Check what size factors were estimated for each of the 4 HCT116 datasets. Check how the total number of gene-assigned reads changes to a more homogeneous number with the normalization (e.g. between 20 and 23 million counts).

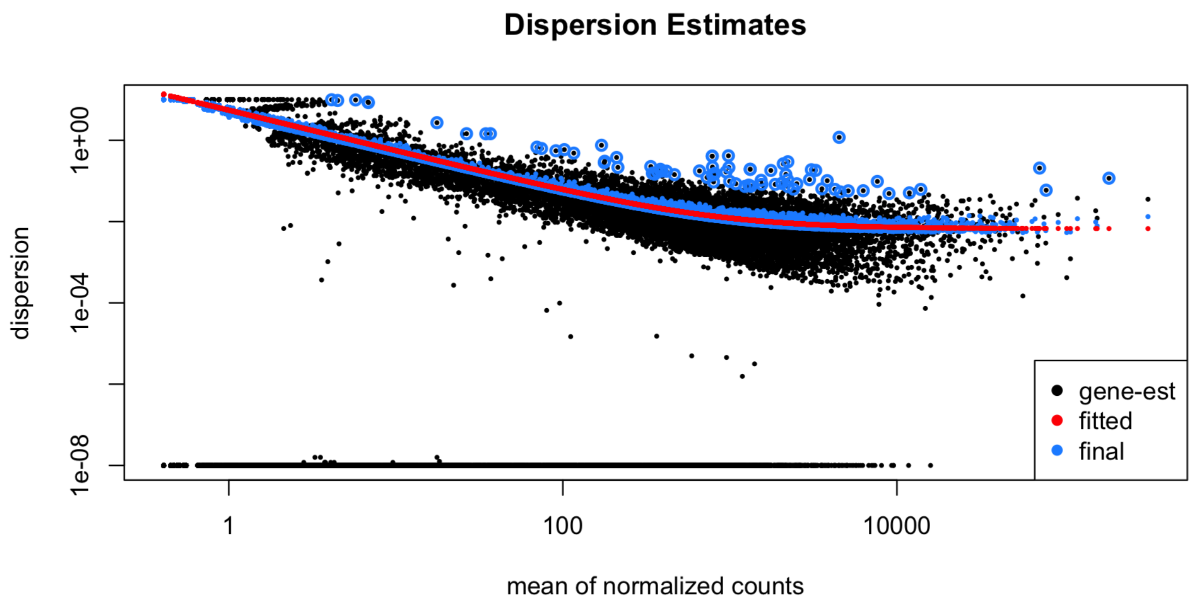
```
sizeFactors(DEdds)
colSums(counts(DEdds, normalized = FALSE))
colSums(counts(DEdds, normalized = TRUE))
```



```
> sizeFactors(DEdds)
SRR4098430 SRR4098431 SRR4098432 SRR4098433
 1.2276148  0.9455801  1.0205486  0.8583701
> colSums(counts(DEdds, normalized = FALSE))
SRR4098430 SRR4098431 SRR4098432 SRR4098433
  28661527   20467240   21644243   17631625
> colSums(counts(DEdds, normalized = TRUE))
SRR4098430 SRR4098431 SRR4098432 SRR4098433
  23347329   21645168   21208438   20540819
```

You can check the dispersion estimates with a simple DESeq2 function. You want to see that the estimates are monotonically descending and that most data points (blue) are nearby the fitted line (red).

```
plotDispEsts(DEdds, main = "Dispersion Estimates")
```

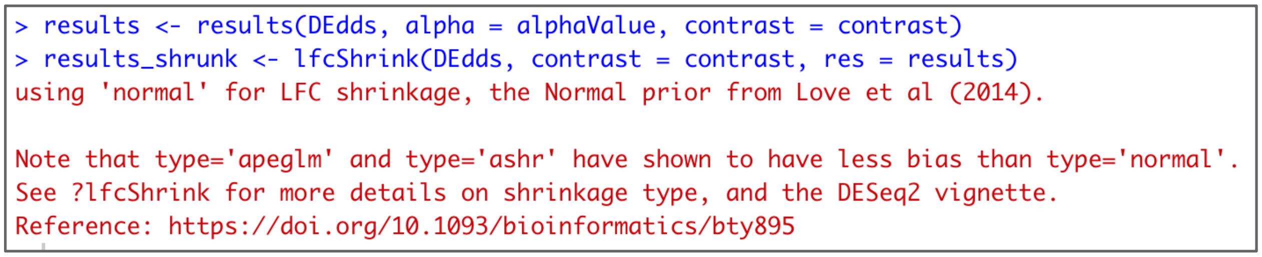


4. Define the alpha value that DESeq2 will need to assign statistical significance, as well as the names of the two experimental conditions we want to compare against each other.

```
alphaValue <- 0.05
contrast <- c("condition", "nutlin", "dmso")
```

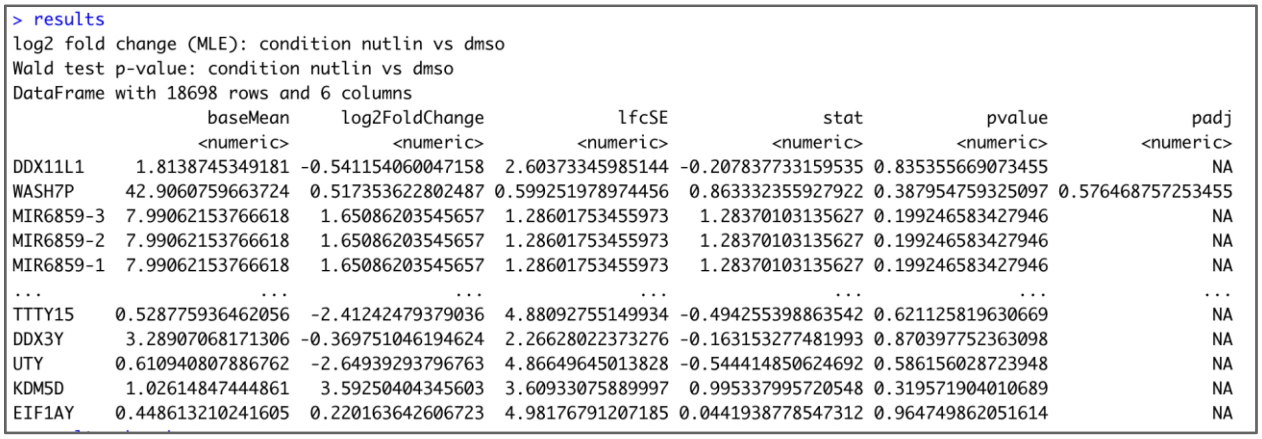
5. Extract statistically significant results, and do DESeq2 special log fold-change shrinkage, which is useful for visualization purposes. DESeq2 will let you know that it is using the normal algorithm for doing the shrinkage, and that there are newer algorithms if you want to test them out as well. They require independent library installation.

```
results <- results(DEdds,
                   alpha = alphaValue,
                   contrast = contrast)

results_shrunk <- lfcShrink(DEdds,
                            contrast = contrast,
                            res = results)
```

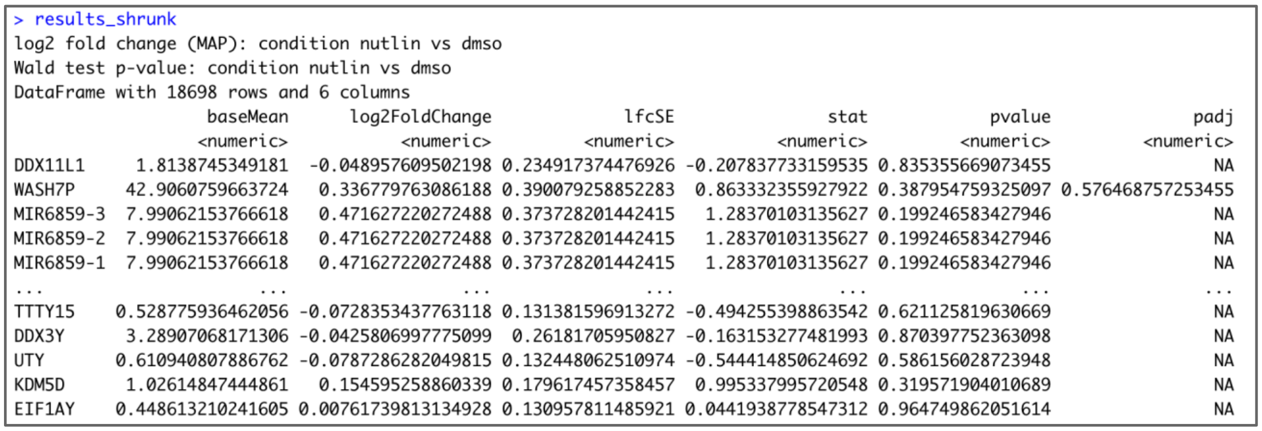


```
> results <- results(DEdds, alpha = alphaValue, contrast = contrast)
> results_shrunk <- lfcShrink(DEdds, contrast = contrast, res = results)
using 'normal' for LFC shrinkage, the Normal prior from Love et al (2014).

Note that type='apeglm' and type='ashr' have shown to have less bias than type='normal'.
See ?lfcShrink for more details on shrinkage type, and the DESeq2 vignette.
Reference: https://doi.org/10.1093/bioinformatics/bty895
```

See that both unshrunk and shrunk results have identical nominal and adjusted p-values. The shrinkage only affects the fold-change estimation values.



```
> results
log2 fold change (MLE): condition nutlin vs dmso
Wald test p-value: condition nutlin vs dmso
DataFrame with 18698 rows and 6 columns
                 baseMean    log2FoldChange             lfcSE                stat             pvalue              padj
                <numeric>         <numeric>         <numeric>           <numeric>          <numeric>         <numeric>
DDX11L1     1.8138745349181 -0.541154060047158  2.60373345985144 -0.207837733159535 0.835355669073455                NA
WASH7P     42.9060759663724  0.517353622802487 0.599251978974456  0.863332355927922 0.387954759325097 0.576468757253455
MIR6859-3  7.99062153766618   1.65086203545657  1.28601753455973   1.28370103135627 0.199246583427946                NA
MIR6859-2  7.99062153766618   1.65086203545657  1.28601753455973   1.28370103135627 0.199246583427946                NA
MIR6859-1  7.99062153766618   1.65086203545657  1.28601753455973   1.28370103135627 0.199246583427946                NA
...                     ...                ...               ...                 ...               ...               ...
TTTY15    0.528775936462056  -2.41242479379036  4.88092755149934 -0.494255398863542 0.621125819630669                NA
DDX3Y      3.28907068171306 -0.369751046194624  2.26628022373276 -0.163153277481993 0.870397752363098                NA
UTY       0.610940807886762  -2.64939293796763  4.86649645013828 -0.544414850624692 0.586156028723948                NA
KDM5D      1.02614847444861   3.59250404345603  3.60933075889997  0.995337995720548 0.319571904010689                NA
EIF1AY    0.448613210241605  0.220163642606723  4.98176791207185 0.0441938778547312 0.964749862051614                NA
```



```
> results_shrunk
log2 fold change (MAP): condition nutlin vs dmso
Wald test p-value: condition nutlin vs dmso
DataFrame with 18698 rows and 6 columns
                 baseMean     log2FoldChange             lfcSE               stat            pvalue              padj
                <numeric>          <numeric>         <numeric>          <numeric>         <numeric>         <numeric>
DDX11L1     1.8138745349181  -0.048957609502198 0.234917374476926 -0.207837733159535 0.835355669073455                NA
WASH7P     42.9060759663724   0.336779763086188 0.390079258852283  0.863332355927922 0.387954759325097 0.576468757253455
MIR6859-3  7.99062153766618   0.471627220272488 0.373728201442415   1.28370103135627 0.199246583427946                NA
MIR6859-2  7.99062153766618   0.471627220272488 0.373728201442415   1.28370103135627 0.199246583427946                NA
MIR6859-1  7.99062153766618   0.471627220272488 0.373728201442415   1.28370103135627 0.199246583427946                NA
...                     ...                 ...               ...                ...               ...               ...
TTTY15    0.528775936462056 -0.0728353437763118 0.131381596913272 -0.494255398863542 0.621125819630669                NA
DDX3Y      3.28907068171306 -0.0425806997775099  0.26181705950827 -0.163153277481993 0.870397752363098                NA
UTY       0.610940807886762 -0.0787286282049815 0.132448062510974 -0.544414850624692 0.586156028723948                NA
KDM5D      1.02614847444861   0.154595258860339 0.179617457358457  0.995337995720548 0.319571904010689                NA
EIF1AY    0.448613210241605 0.00761739813134928 0.130957811485921 0.0441938778547312 0.964749862051614                NA
```

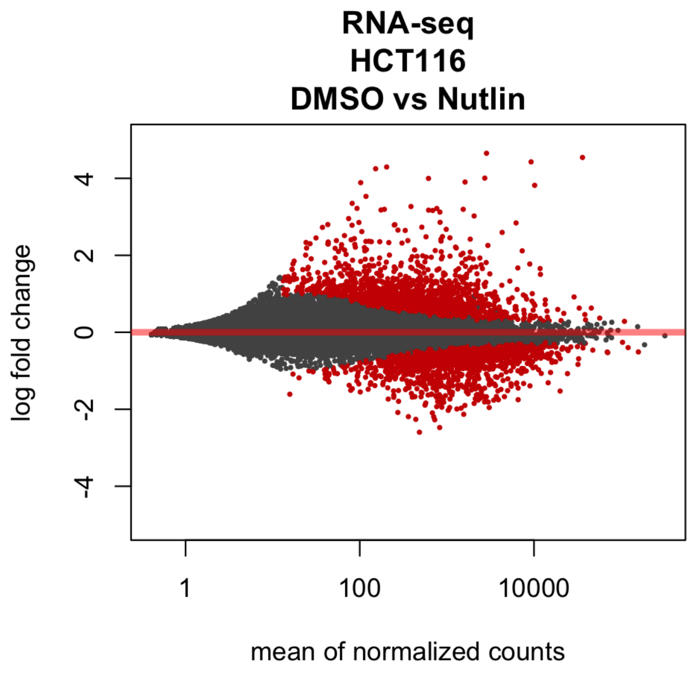
6. You can check a global visualization of how the Human genes changed upon the Nutlin treatment using DESeq2 **plotMA** function.

   Notice that you can call a given library's function by calling the name of the library followed by two colons and the name of the function. This helps R in case there are two functions with conflicting names.

   The resulting MA figure will plot each gene's fold-change in the Y-axis, and such gene's normalized counts in the X-axis. You can tell **plotMA** to color genes (red are significant, gray are non-significant) by significance by using the same alpha level threshold you defined previously.

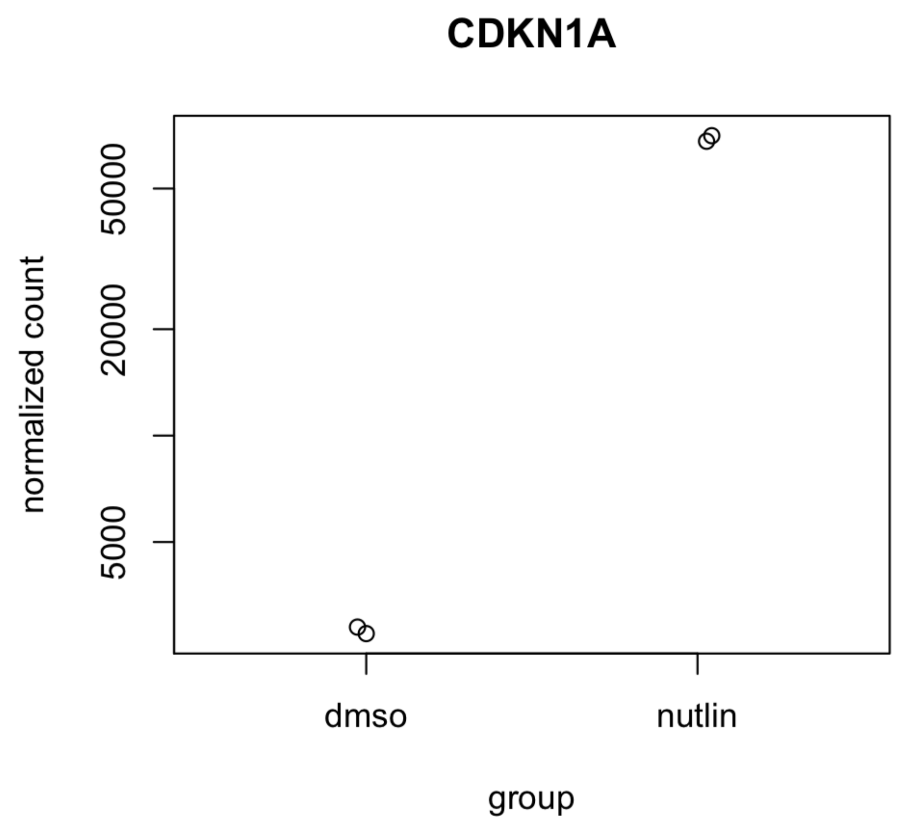
You can observe that there are more red dots with positive than negative fold-change. This behavior will depend on the treatment that the cells of your experiment are exposed to.

```
DESeq2::plotMA(results_shrunk,
               alpha = alphaValue,
               main = "RNA-seq\nHCT116\nDMSO vs Nutlin",
               xlab = "mean of normalized counts",
               ylab = "log fold change",
               ylim = c(-5,5))
```



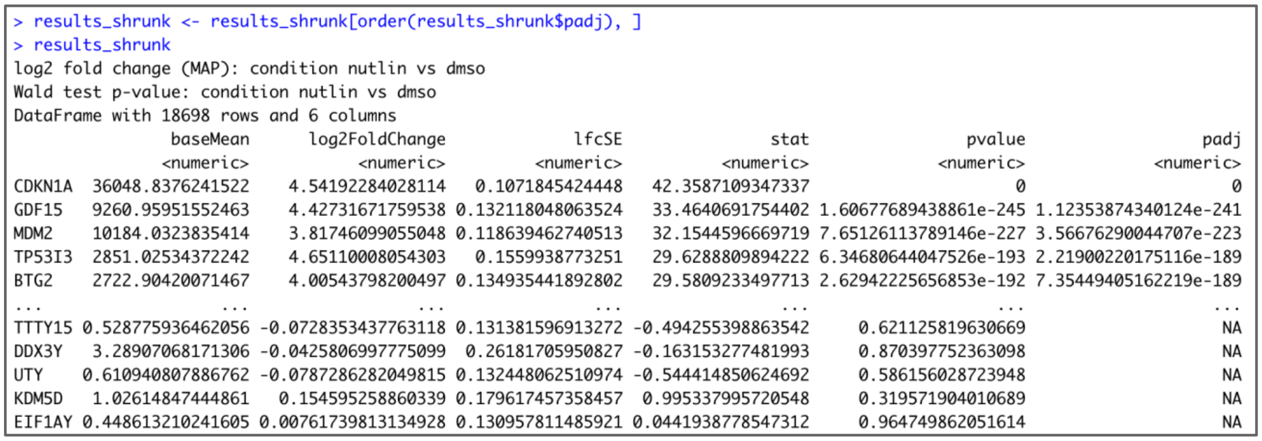
7. You can plot the normalized read counts for a given gene using DESeq2 built-in function **plotCounts**. You need to tell it the name of the gene as it appears in the original annotation file, as well as the name of the column in the condition file that denotes either “Nutlin” or “DMSO” as the conditions.

```
gene <- "CDKN1A"
plotCounts(DEdds, gene,
           intgroup = "condition",
           normalized = TRUE)
```



8. Order the gene results in descending order by their adjusted p-values, so that the most significant genes will be on the top of your results table. You can see that the very top gene is CDKN1A or p21, a known gene that controls cell cycle progression directly controlled by the transcription factor p53, which itself is activated by the drug Nutlin.

```
results_shrunk <- results_shrunk[order(results_shrunk$padj), ]
```



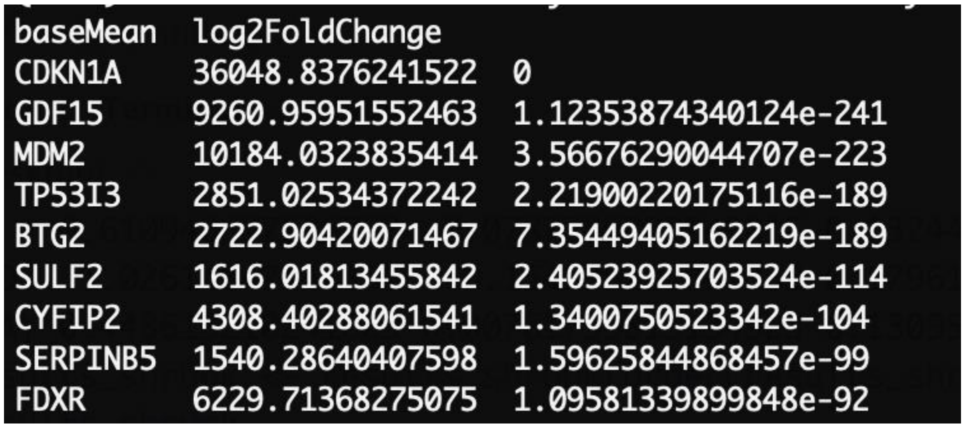
```
> results_shrunk <- results_shrunk[order(results_shrunk$padj), ]
> results_shrunk
log2 fold change (MAP): condition nutlin vs dmso
Wald test p-value: condition nutlin vs dmso
DataFrame with 18698 rows and 6 columns
                baseMean      log2FoldChange              lfcSE                stat               pvalue                 padj
               <numeric>           <numeric>          <numeric>           <numeric>            <numeric>            <numeric>
CDKN1A  36048.8376241522    4.54192284028114   0.1071845424448    42.3587109347337                    0                    0
GDF15   9260.95951552463    4.42731671759538 0.132118048063524    33.4640691754402 1.60677689438861e-245 1.12353874340124e-241
MDM2    10184.0323835414    3.81746099055048 0.118639462740513    32.1544596669719 7.65126113789146e-227 3.56676290044707e-223
TP53I3  2851.02534372242    4.65110008054303   0.1559938773251    29.6288809894222 6.34680644047526e-193 2.21900220175116e-189
BTG2    2722.90420071467    4.00543798200497 0.134935441892802    29.5809233497713 2.62942225656853e-192 7.35449405162219e-189
...                  ...                 ...               ...                 ...                  ...                  ...
TTTY15 0.528775936462056 -0.0728353437763118 0.131381596913272  -0.494255398863542    0.621125819630669                   NA
DDX3Y   3.28907068171306 -0.0425806997775099  0.26181705950827  -0.163153277481993    0.870397752363098                   NA
UTY    0.610940807886762 -0.0787286282049815 0.132448062510974  -0.544414850624692    0.586156028723948                   NA
KDM5D   1.02614847444861   0.154595258860339 0.179617457358457   0.995337995720548    0.319571904010689                   NA
EIF1AY 0.448613210241605 0.00761739813134928 0.130957811485921 0.0441938778547312    0.964749862051614                   NA
```

9. Filter out only those genes whose adjusted p-value are less than the defined alpha value. Finally, store your significant genes onto a text output file. You can use this file for any downstream analysis you deem appropriate, including gene set enrichment software such as GSEA and others.

```
results_shrunk_sig <- subset(results_shrunk, padj < alphaValue)

write.table(results_shrunk_sig,
           sep = "\t",
           quote = FALSE,
           row.names = TRUE,
           col.names = TRUE,
           "PATH/TO/DIR/Andrysik2017_RNAseq_Nutlin_results.tsv")
```

- You can then see the output text file from your bash terminal.

```
cat Andrysik2017_RNAseq_Nutlin_results.tsv | cut -f1,2,7 | head
```



```
baseMean  log2FoldChange
CDKN1A    36048.8376241522  0
GDF15     9260.95951552463  1.12353874340124e-241
MDM2      10184.0323835414  3.56676290044707e-223
TP53I3    2851.02534372242  2.21900220175116e-189
BTG2      2722.90420071467  7.35449405162219e-189
SULF2     1616.01813455842  2.40523925703524e-114
CYFIP2    4308.40288061541  1.3400750523342e-104
SERPINB5  1540.28640407598  1.59625844868457e-99
FDXR      6229.71368275075  1.09581339899848e-92
```