DE analysis with DESeq2

Princess Rodriguez

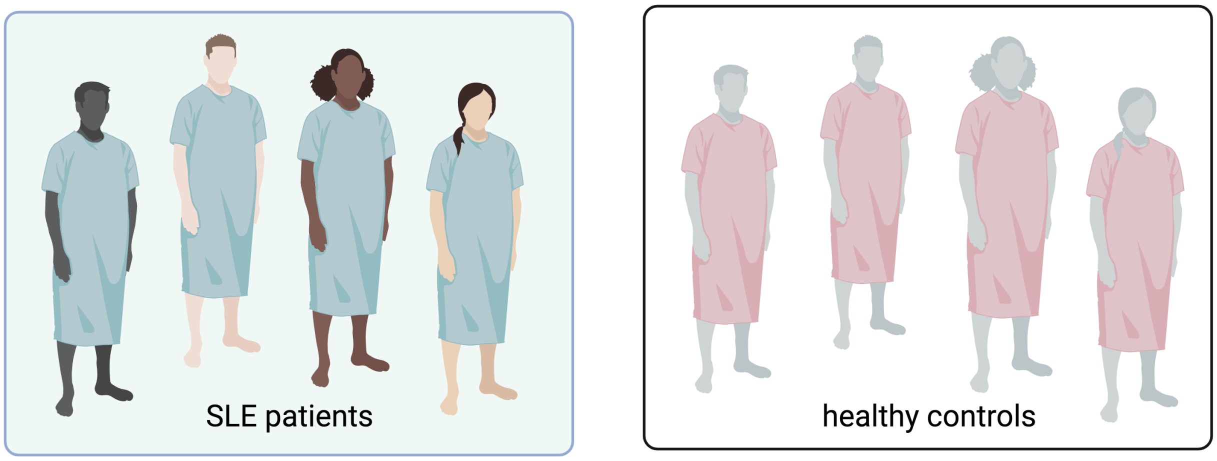
11/9/2023

About the Dataset

For this example analysis, we will use this systemic lupus erythematosus dataset (https://www.refine.bio/experiments/SRP136102/systemic-lupus-erythematosus-patient-blood-with-controls).

Design: Whole blood was collected in PaxGene tubes from 31 SLE and 29 healthy donors. RNA libraries were prepared for sequencing using standard Illumina protocols. Sequencing was performed on a Illumina HiSeq2000 platform. We will only be working with a subset of the dataset.

- Overall our goal is to compare the transcriptional changes that occur in individuals with SLE versus healthy donors.



Install the required R libraries

```
# Install CRAN packages
#install.packages(c("BiocManager", "RColorBrewer", "tidyverse", "devtools", "pheatmap", "gprofiler2"))

#install.packages("genekitr", dependencies = TRUE, INSTALL_opts = '--no-lock')

# Install Bioconductor packages
#BiocManager::install(c("DESeq2", "org.Hs.eg.db", "EnhancedVolcano", "hypeR"))
```

Load the required libraries

```
library(DESeq2)
library(RColorBrewer)
library(pheatmap)
library(ggplot2)
library(genekitr)
library(EnhancedVolcano)
```

Introduction

We spent last week reviewing the quality of the dataset. Now, our next step is to analyze the counts data (i.e. counts matrix) for the detection of differentially expressed genes (DEGs). **A gene is declared as differentially expressed if an observed difference or change in read counts between two experimental conditions is statistically significant.**

DEGs are selected based on a combination of thresholds, including log2FC and false discovery rate (padj).

- Fold Change:
  - For a given comparison, a positive fold change value indicates an increase of expression, while a negative fold change indicates a decrease in expression.
  - The value is typically reported in logarithmic scale (base 2). For example, log2 fold change of 1.5 for a specific gene in the “SLE vs Normal comparison” means that the expression of that gene is increased in SLE relative to Normal by a multiplicative factor of 2^1.5 ≈ 2.82.
- P-value: Indicates whether the gene analysed is likely to be differentially expressed in that comparison.
- Adjusted (or, corrected for multiple genes testing) p-value: The p-value obtained for each gene above is re-calculated to corrected after running many statistical tests.

As a result, we can say that genes with adjusted p-value < 0.05 and an absolute log2 fold change < 1 are significantly differentially expressed between these two samples.

Several R packages are available for expression analysis, including `DEGseq2`. The package `DESeq2` provides methods to test for differential expression by use of negative binomial generalized linear models.

Overview

To use `DESeq2` it requires the following:

1. counts matrix
2. table of sample information
3. design indicates how to model the samples

The DESeqDataSet

The *object* used by the `DESeq2` package to store the (1) read counts, (2) table of sample information , and (3) design is the `DESeqDataSet`, this is usually represented as `dds`.

dds = DESeqDataSet(1 + 2 + 3)
dds = DESeqDataSet(readcounts, table of sample, design)

There are multiple ways of constructing a *DESeqDataSet*, depending on what pipeline was used upstream of DESeq2 to generated counts or estimated counts.

a. From transcript abundance files and tximport - `DESeqDataSetFromTximport()`

b. From a count matrix - DESeqDataSetFromMatrix
c. From htseq-count files - `DESeqDataSetFromHTSeqCount`

Input files

First, we will input the counts files and sample table. The sample table is simply the annotation file.

```
meta <- read.table(file = "metadata_SRP136102.tsv",
                   sep = "\t",
                   stringsAsFactors = FALSE,
                   header = TRUE)


meta <- meta[, c("refinebio_accession_code", "refinebio_sex")]

colnames(meta) <- c('Sample', 'Sex')

meta$Treatment <- c("sle", "sle", "sle", "sle", "sle", "sle", "normal", "normal", "normal", "normal", "nor
mal", "normal")

meta$Replicate <- c(1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 5, 6)

meta
```

```
##        Sample    Sex Treatment Replicate
## 1  SRR6870284 female       sle         1
## 2  SRR6870286 female       sle         2
## 3  SRR6870292 female       sle         3
## 4  SRR6870298 female       sle         4
## 5  SRR6870302 female       sle         5
## 6  SRR6870304 female       sle         6
## 7  SRR6870356 female    normal         1
## 8  SRR6870358 female    normal         2
## 9  SRR6870360 female    normal         3
## 10 SRR6870364 female    normal         4
## 11 SRR6870366 female    normal         5
## 12 SRR6870370   male    normal         6
```

```
# read-in counts matrix
data <- read.table(file = "SRP136102.tsv",
                   sep = "\t",
                   stringsAsFactors = FALSE,
                   header = TRUE,
                   row.names = 1)

meta$ColNames <- paste(meta$Treatment, meta$Replicate, sep = "_")
colnames(data) <- meta$ColNames
colnames(data)
```

```
##  [1] "sle_1"    "sle_2"    "sle_3"    "sle_4"    "sle_5"    "sle_6"
##  [7] "normal_1" "normal_2" "normal_3" "normal_4" "normal_5" "normal_6"
```

```
head(data)
```

```
##                       sle_1      sle_2     sle_3      sle_4      sle_5
## ENSG00000000003   46.39945   24.51837   37.5936   44.98844   34.44015
## ENSG00000000005    0.00000    0.00000    0.0000    0.00000    0.00000
## ENSG00000000419  229.43710  213.17186  206.5260  145.13773  193.27979
## ENSG00000000457  529.40980  441.44977  401.6713  581.02700  472.23468
## ENSG00000000460   84.78243   78.22288  120.5951   95.55912  120.39508
## ENSG00000000938 3811.94040 3944.22310 6929.4775 2112.83840 4020.41260
##                       sle_6   normal_1   normal_2   normal_3  normal_4
## ENSG00000000003   39.04514   19.49657   37.34583   24.32008   15.3744
## ENSG00000000005    0.00000    0.00000    0.00000    0.00000    0.0000
## ENSG00000000419  237.87276  166.87498  142.88065  202.96245  204.9115
## ENSG00000000457  651.12090  405.19418  539.30914  605.55975  351.6534
## ENSG00000000460   77.85001   83.13609  129.71800  141.67328  100.7051
## ENSG00000000938 3335.41140 3803.87100 4677.96500 3465.53780 4443.0034
##                    normal_5    normal_6
## ENSG00000000003   17.38905   26.844183
## ENSG00000000005    0.00000    2.720015
## ENSG00000000419  103.15216  237.559280
## ENSG00000000457  265.86420  481.142120
## ENSG00000000460   50.99625  136.560960
## ENSG00000000938 1339.46720 3685.132600
```

```
# round all expression counts
data <- round(data)
#data
```

Creating a DESeq Object

```
dds <- DESeqDataSetFromMatrix(countData = data,
                                       colData = meta,
                                       design= ~ Treatment)
```

```
## converting counts to integer mode
```

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

```
dds
```

```
## class: DESeqDataSet
## dim: 43363 12
## metadata(1): version
## assays(1): counts
## rownames(43363): ENSG00000000003 ENSG00000000005 ... ENSG00000286271
##   ENSG00000286272
## rowData names(0):
## colnames(12): sle_1 sle_2 ... normal_5 normal_6
## colData names(5): Sample Sex Treatment Replicate ColNames
```

The *DESeqDataSet* object must have an associated *design formula*. The design formula expresses the variables which will be used in modeling. The formula should be a tilde (~) followed by the variables with plus signs between them if required.

Note: The design can be changed later, however then all differential analysis steps should be repeated, as the design formula is used to estimate the log2 fold changes of the model.

Note: In order to benefit from the default settings of the package, one should put the variable of interest at the end of the formula and make sure the control level is the first level.

Pre-filtering

While it is not necessary to pre-filter low count genes before running the DESeq2 functions, there are two reasons which make pre-filtering useful: by removing rows in which there are very few reads, we reduce the memory size of the `dds` data object, and we increase the speed of the transformation and testing functions within DESeq2. It can also improve visualizations, as features with no information for differential expression are not plotted.

Here we perform a minimal pre-filtering to keep only rows that have at least 10 reads total.

```
##pre-filter to remove low-read rows
keep <- rowSums(counts(dds)) >= 10
dds <- dds[keep,]
```

Note on factor levels

By default, R will choose a *reference level* for factors based on alphabetical order. Therefore, if you never tell the DESeq2 functions which level you want to compare against (e.g. which level represents the control group), the comparisons will be based on the alphabetical order of the levels.

use *relevel* to specify the reference level:

```
dds$Treatment <- relevel(dds$Treatment, ref = "normal")
```

Now let’s run the Differential expression analysis

The standard differential expression analysis steps are wrapped into a single function, *DESeq*. The estimation steps performed by this function are described in the manual page for `?DESeq`.

Results table will be generated using the function *results*, which extracts a results table with log2 fold changes, *p* values and adjusted *p* values.

Using the *results* function, the user can specify the comparison to build a results table for.

Details about the comparison are printed to the console, directly above the results table. The text, `group YS veh vs WT veh`, tells you that the estimates are of the logarithmic fold change log2(YS/WT) i.e. log2(treated/untreated).

```
##run Differential Expression Analysis
dds <- DESeq(dds)
```

```
## estimating size factors
```

```
## estimating dispersions
```

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

```
## mean-dispersion relationship
```

```
## final dispersion estimates
```

```
## fitting model and testing
```

```
res <- results(dds)
head(res)
```

```
## log2 fold change (MLE): Treatment sle vs normal
## Wald test p-value: Treatment sle vs normal
## DataFrame with 6 rows and 6 columns
##                  baseMean log2FoldChange     lfcSE      stat     pvalue
##                 <numeric>      <numeric> <numeric> <numeric>  <numeric>
## ENSG00000000003   29.6859       0.681638  0.250879  2.716996 0.00658774
## ENSG00000000419  186.1323       0.168331  0.187677  0.896918 0.36976260
## ENSG00000000457  461.2052       0.185962  0.121268  1.533481 0.12515749
## ENSG00000000460   98.6559      -0.131870  0.207218 -0.636385 0.52452541
## ENSG00000000938 3695.9225       0.204306  0.286646  0.712748 0.47600146
## ENSG00000000971   72.1094      -0.346477  0.511997 -0.676716 0.49858621
##                      padj
##                 <numeric>
## ENSG00000000003 0.0643199
## ENSG00000000419 0.6391759
## ENSG00000000457 0.3601913
## ENSG00000000460 0.7559490
## ENSG00000000938 0.7237027
## ENSG00000000971 0.7383242
```

```
dim(res)
```

```
## [1] 26055     6
```

Data transformation and visualization

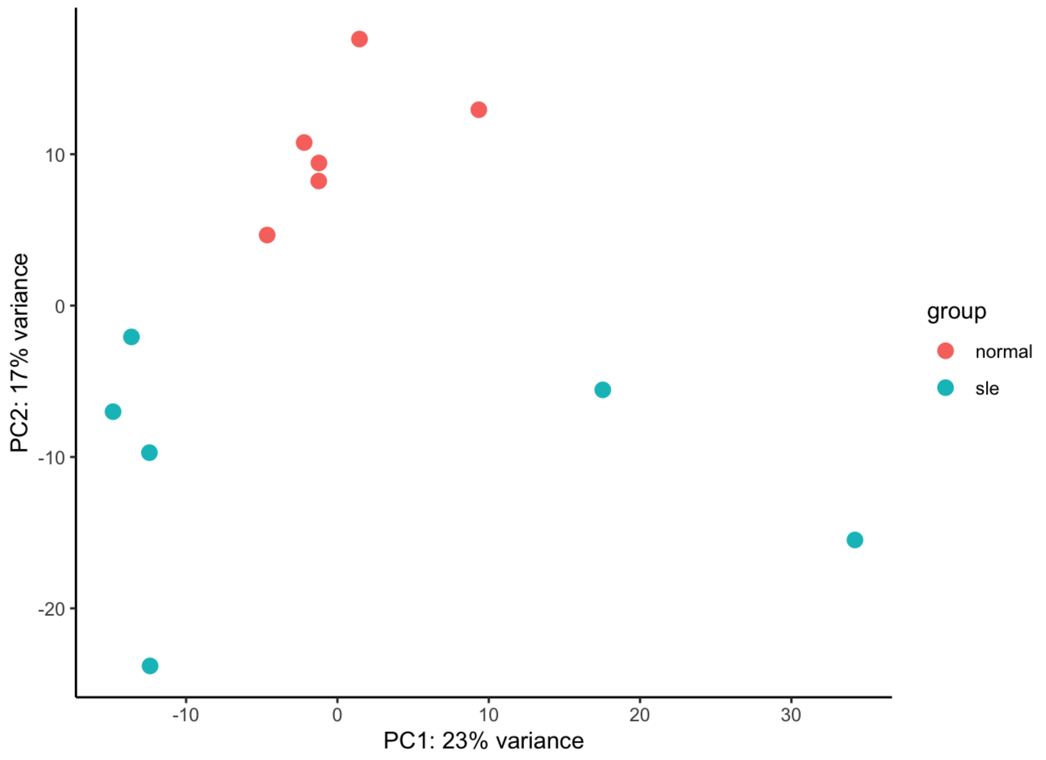
Principal components analysis (PCA)

In order to test for differential analysis, we used raw counts. However, for other downstream analysis such as visualization and clustering, it is useful to transform the data. Below we are using either *variance stabilizing transformation* (vst) (Tibshirani 1988; Huber et al. 2003; Anders and Huber 2010) or regularized logarithm (rlog) (Love, Huber, and Anders 2014). Both transformations will produce transformed data on the log2 scale which has been normalized with respect to library size or other normalization factors.

While using either function, one can specify the argument *blind* for whether the transformation should be blind to the sample information specified by the design formula. A blind dispersion estimate is not appropriate if one expects that many or the majority of genes (rows) will have large differences in counts which are explained by the experimental design. These differences due to experimental design will be interpreted as **unwanted noise**, and will result in overly shrinking the transformed values towards each other. By setting blind to FALSE, the dispersions already estimated will be used to perform transformations, or if not present, they will be estimated using the current design formula.

Now using the transformed data we will generate a PCA plot. The PCA plot will show the samples in a 2D plane spanned by the first two principal components. This type of plot is useful for visualizing the overall effect of experimental covariates (ex. age) and batch effects (ex. day sample was prepared).

```
vsd <- vst(dds, blind=FALSE)
plotPCA(vsd, intgroup=c("Treatment")) + theme_classic()
```

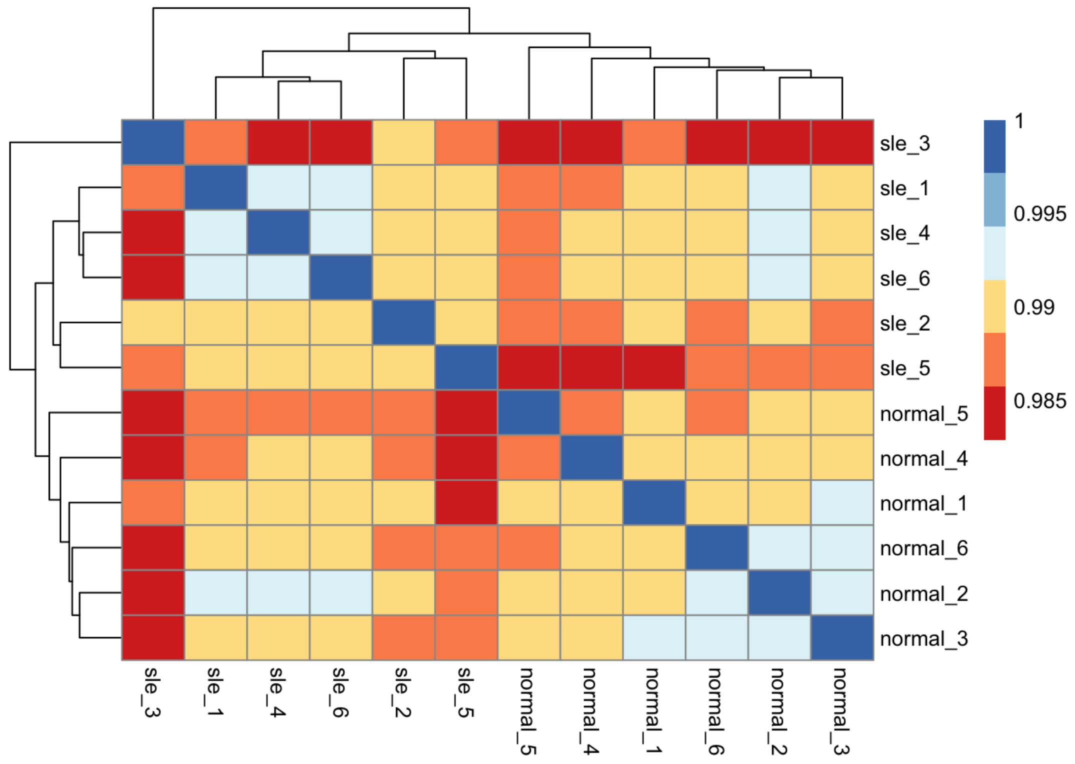


website: https://ggplot2.tidyverse.org/reference/ggtheme.html (https://ggplot2.tidyverse.org/reference/ggtheme.html)

Interpretation: PCA plot gives us a way to bring out strong patterns from large and complex datasets. Each dot represents one sample. There are a total of 12 samples, so we expect 12 dots. We expect biological replicates to have similar variance, so these should cluster together. PCA-1 dimension captures the most variation occuring the sample set. PCA-2 describes the next strongest source of variance in the sample set.

Hierarchical Clustering Heatmap

```
rld <- rlog(dds, blind=TRUE)
rld_mat <- assay(rld)
rld_cor <- cor(rld_mat)

heat.colors <- brewer.pal(6, "RdYlBu")
pheatmap(rld_cor, color = heat.colors)
```



website: https://r-graph-gallery.com/38-rcolorbrewers-palettes.html (https://r-graph-gallery.com/38-rcolorbrewers-palettes.html)

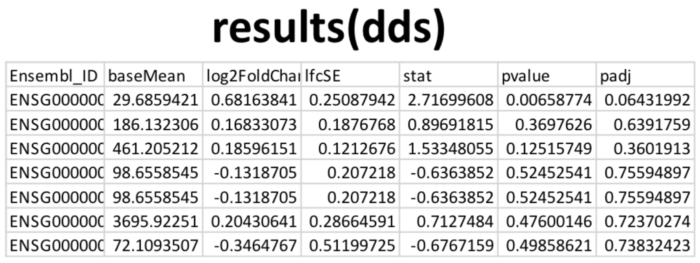
Output results in tabular format

```
# SLE vs normal
sle_n_res <- results(dds, contrast = c("Treatment", "sle", "normal"))
sle_n_sigs <- na.omit(sle_n_res)
sle_n_sigs <- subset(sle_n_sigs, padj < 0.05 & abs(log2FoldChange) > 1)
sle_n_sig_data <- merge(as.data.frame(sle_n_sigs),
                        as.data.frame(counts(dds, normalized = TRUE)),
                        by = "row.names", sort = FALSE)

names(sle_n_sig_data)[1] <- "Ensembl_ID"

sle_n_res_data <- merge(as.data.frame(sle_n_res),
                        as.data.frame(counts(dds, normalized=TRUE)),
                        by="row.names", sort=FALSE)

names(sle_n_res_data)[1] <- "Ensembl_ID"

head(res)
```

```
## log2 fold change (MLE): Treatment sle vs normal
## Wald test p-value: Treatment sle vs normal
## DataFrame with 6 rows and 6 columns
##                  baseMean log2FoldChange     lfcSE      stat     pvalue
##                 <numeric>      <numeric> <numeric> <numeric>  <numeric>
## ENSG00000000003   29.6859       0.681638  0.250879  2.716996 0.00658774
## ENSG00000000419  186.1323       0.168331  0.187677  0.896918 0.36976260
## ENSG00000000457  461.2052       0.185962  0.121268  1.533481 0.12515749
## ENSG00000000460   98.6559      -0.131870  0.207218 -0.636385 0.52452541
## ENSG00000000938 3695.9225       0.204306  0.286646  0.712748 0.47600146
## ENSG00000000971   72.1094      -0.346477  0.511997 -0.676716 0.49858621
##                      padj
##                 <numeric>
## ENSG00000000003 0.0643199
## ENSG00000000419 0.6391759
## ENSG00000000457 0.3601913
## ENSG00000000460 0.7559490
## ENSG00000000938 0.7237027
## ENSG00000000971 0.7383242
```

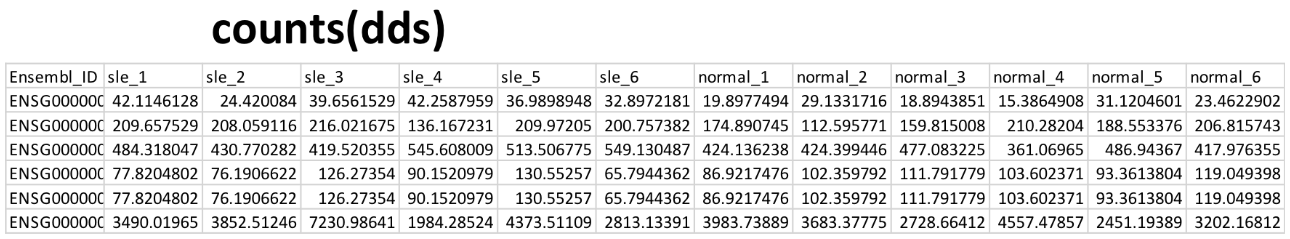
We are doing a few things here:

1. from the "results" - we are specifying a contrast
2. if there are any "NA" values in our dataframe they are being removed and placed in new object called sle_n_sigs
3. This dataframe still 43,363 rows. We now want to filter this dataframe and by log2fc and padj cutoffs
4. sle_n_sig_data contains information from two merged dataframes.



results(dds)

Ensembl_ID	baseMean	log2FoldChar	lfcSE	stat	pvalue	padj
ENSG00000C	29.6859421	0.68163841	0.25087942	2.71699608	0.00658774	0.06431992
ENSG00000C	186.132306	0.16833073	0.1876768	0.89691815	0.3697626	0.6391759
ENSG00000C	461.205212	0.18596151	0.1212676	1.53348055	0.12515749	0.3601913
ENSG00000C	98.6558545	-0.1318705	0.207218	-0.6363852	0.52452541	0.75594897
ENSG00000C	98.6558545	-0.1318705	0.207218	-0.6363852	0.52452541	0.75594897
ENSG00000C	3695.92251	0.20430641	0.28664591	0.7127484	0.47600146	0.72370274
ENSG00000C	72.1093507	-0.3464767	0.51199725	-0.6767159	0.49858621	0.73832423



counts(dds)

Ensembl_ID	sle_1	sle_2	sle_3	sle_4	sle_5	sle_6	normal_1	normal_2	normal_3	normal_4	normal_5	normal_6
ENSG00000C	42.1146128	24.420084	39.6561529	42.2587959	36.9898948	32.8972181	19.8977494	29.1331716	18.8943851	15.3864908	31.1204601	23.4622902
ENSG00000C	209.657529	208.059116	216.021675	136.167231	209.97205	200.757382	174.890745	112.595771	159.815008	210.28204	188.553376	206.815743
ENSG00000C	484.318047	430.770282	419.520355	545.608009	513.506775	549.130487	424.136238	424.399446	477.083225	361.06965	486.94367	417.976355
ENSG00000C	77.8204802	76.1906622	126.27354	90.1520979	130.55257	65.7944362	86.9217476	102.359792	111.791779	103.602371	93.3613804	119.049398
ENSG00000C	77.8204802	76.1906622	126.27354	90.1520979	130.55257	65.7944362	86.9217476	102.359792	111.791779	103.602371	93.3613804	119.049398
ENSG00000C	3490.01965	3852.51246	7230.98641	1984.28524	4373.51109	2813.13391	3983.73889	3683.37775	2728.66412	4557.47857	2451.19389	3202.16812

GeneID conversion

Our objective is now to convert from one gene id type to another. There are many Gene ID types, below are some of the most common:

- Entrez Gene - 1457, 2002, 1950
- Gene Symbol - Ikzf1, Tcf1, Cd19
- Ensembl ID - ENSMUSG00000018654

```
ids <- sle_n_res_data$Ensembl_ID
gene_symbol_map <- transId(ids, transTo = "symbol") #default is human!
```

```
## Some ID occurs one-to-many match, like "ENSG00000000460, ENSG00000004866, ENSG00000063587"...
```

```
## 79.95% genes are mapped to symbol
```

```
head(gene_symbol_map)
```

```
##          input_id   symbol
## 1 ENSG00000000003   TSPAN6
## 2 ENSG00000000419     DPM1
## 3 ENSG00000000457    SCYL3
## 4 ENSG00000000460 C1orf112
## 5 ENSG00000000460    FIRRM
## 6 ENSG00000000938      FGR
```

```
#write.csv(gene_symbol_map, "gene_symbol_map.csv", quote=F)
```

Let's look at the structure of `sle_n_res_data` :

```
str(sle_n_res_data)
```

```
## 'data.frame':    26055 obs. of  19 variables:
##  $ Ensembl_ID    : 'AsIs' chr  "ENSG00000000003" "ENSG00000000419" "ENSG00000000457" "ENSG00000000460"
...
##  $ baseMean      : num  29.7 186.1 461.2 98.7 3695.9 ...
##  $ log2FoldChange: num  0.682 0.168 0.186 -0.132 0.204 ...
##  $ lfcSE         : num  0.251 0.188 0.121 0.207 0.287 ...
##  $ stat          : num  2.717 0.897 1.533 -0.636 0.713 ...
##  $ pvalue        : num  0.00659 0.36976 0.12516 0.52453 0.476 ...
##  $ padj          : num  0.0643 0.6392 0.3602 0.7559 0.7237 ...
##  $ sle_1         : num  42.1 209.7 484.3 77.8 3490 ...
##  $ sle_2         : num  24.4 208.1 430.8 76.2 3852.5 ...
##  $ sle_3         : num  39.7 216 419.5 126.3 7231 ...
##  $ sle_4         : num  42.3 136.2 545.6 90.2 1984.3 ...
##  $ sle_5         : num  37 210 514 131 4374 ...
##  $ sle_6         : num  32.9 200.8 549.1 65.8 2813.1 ...
##  $ normal_1      : num  19.9 174.9 424.1 86.9 3983.7 ...
##  $ normal_2      : num  29.1 112.6 424.4 102.4 3683.4 ...
##  $ normal_3      : num  18.9 159.8 477.1 111.8 2728.7 ...
##  $ normal_4      : num  15.4 210.3 361.1 103.6 4557.5 ...
##  $ normal_5      : num  31.1 188.6 486.9 93.4 2451.2 ...
##  $ normal_6      : num  23.5 206.8 418 119 3202.2 ...
```

We would like to add the gene names to this data.frame. To do this we will use `dplyr::left_join`

```
sle_n_res_data_gene <- dplyr::left_join(sle_n_res_data,gene_symbol_map,
                        by=c('Ensembl_ID'='input_id'))
```

```
str(sle_n_res_data_gene)
```

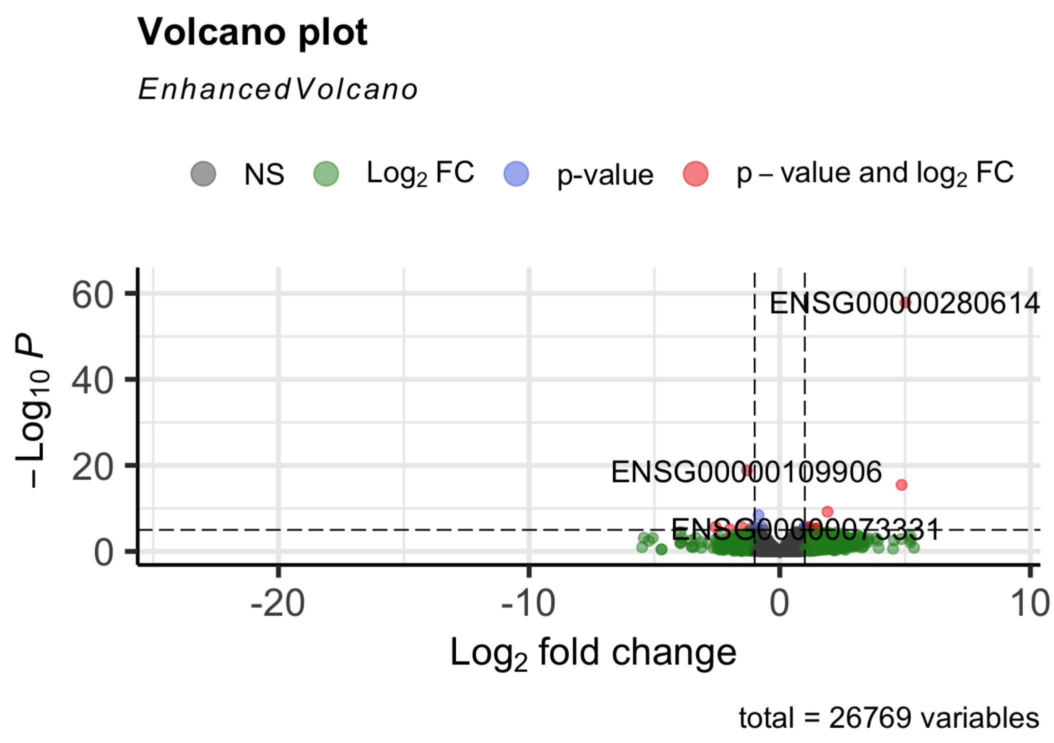
```
## 'data.frame':    26769 obs. of  20 variables:
##  $ Ensembl_ID    : 'AsIs' chr  "ENSG00000000003" "ENSG00000000419" "ENSG00000000457" "ENSG00000000460"
...
##  $ baseMean      : num  29.7 186.1 461.2 98.7 98.7 ...
##  $ log2FoldChange: num  0.682 0.168 0.186 -0.132 -0.132 ...
##  $ lfcSE         : num  0.251 0.188 0.121 0.207 0.207 ...
##  $ stat          : num  2.717 0.897 1.533 -0.636 -0.636 ...
##  $ pvalue        : num  0.00659 0.36976 0.12516 0.52453 0.52453 ...
##  $ padj          : num  0.0643 0.6392 0.3602 0.7559 0.7559 ...
##  $ sle_1         : num  42.1 209.7 484.3 77.8 77.8 ...
##  $ sle_2         : num  24.4 208.1 430.8 76.2 76.2 ...
##  $ sle_3         : num  39.7 216 419.5 126.3 126.3 ...
##  $ sle_4         : num  42.3 136.2 545.6 90.2 90.2 ...
##  $ sle_5         : num  37 210 514 131 131 ...
##  $ sle_6         : num  32.9 200.8 549.1 65.8 65.8 ...
##  $ normal_1      : num  19.9 174.9 424.1 86.9 86.9 ...
##  $ normal_2      : num  29.1 112.6 424.4 102.4 102.4 ...
##  $ normal_3      : num  18.9 159.8 477.1 111.8 111.8 ...
##  $ normal_4      : num  15.4 210.3 361.1 103.6 103.6 ...
##  $ normal_5      : num  31.1 188.6 486.9 93.4 93.4 ...
##  $ normal_6      : num  23.5 206.8 418 119 119 ...
##  $ symbol        : chr  "TSPAN6" "DPM1" "SCYL3" "C1orf112" ...
```

```
#output tabular files
write.csv(sle_n_res_data_gene,
          quote = FALSE,
          file = "sle_vs_normal_normalized_matrix.csv")

sle_n_res_data_gene_sig <- subset(sle_n_res_data_gene,
                                  padj < 0.05 & abs(log2FoldChange) > 1)

write.csv(sle_n_res_data_gene_sig,
          quote = FALSE,
          file = "sle_vs_normal_deg_padj0.05_log2fc1.csv")
```

Volcano Plot

A common plot used to represent a global view of the data is a volcano plot. Here, log transformed adjusted p-values are plotted on the y-axis while the log2 fold change is plotted on the x-axis. We will be generating volcano plots using `EnhancedVolcano` a package created by Kevin Blighe, Sharmila Rana, and Myles Lewis to generate publication-ready volcano plots.

```
EnhancedVolcano(sle_n_res_data_gene,
                lab = as.character(sle_n_res_data_gene$Ensembl_ID),
                x = 'log2FoldChange',
                y = 'padj')
```

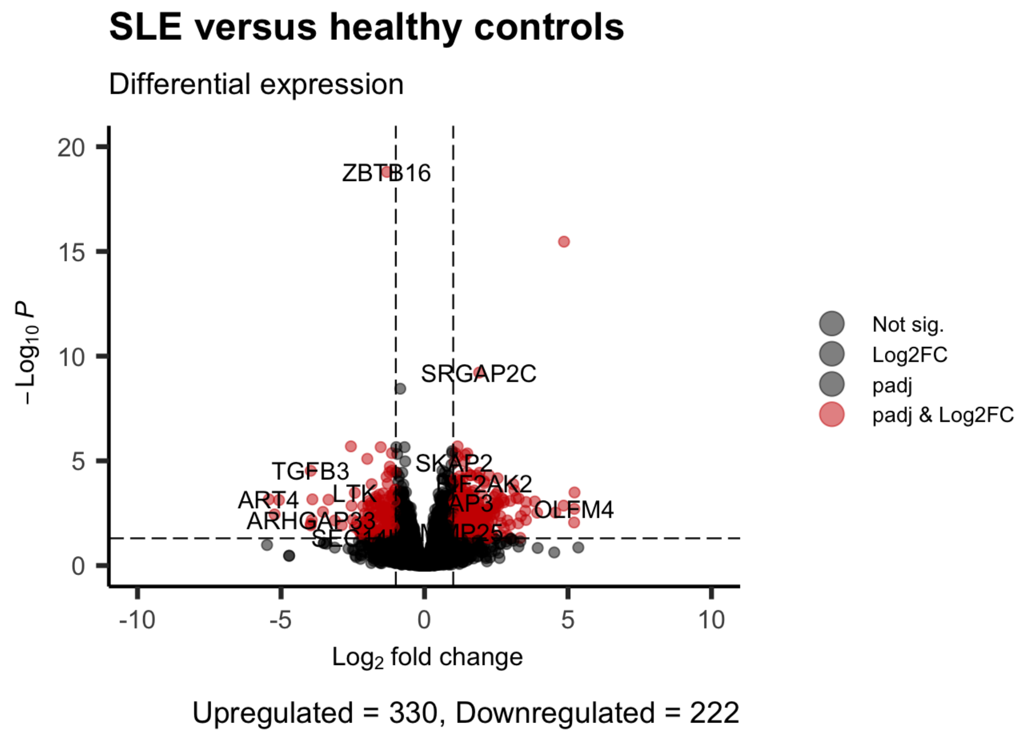


Class Exercise:

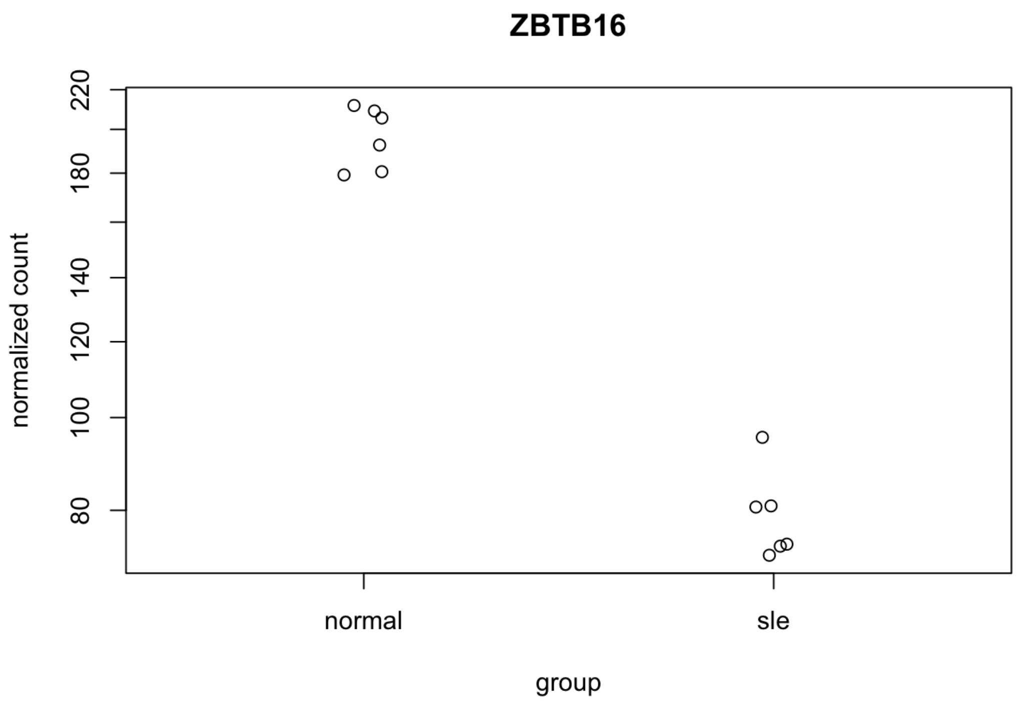
Using the EnhancedVolcano website (https://github.com/kevinblighe/EnhancedVolcano) and/or help page, add the following arguments:

1. title = SLE versus healthy controls
2. subtitle = Differential expression
3. caption = “Upregulated = 330, Downregulated = 222”
4. set the x axis limits to -10, 10
5. set the y axis limits to 0, 20
6. remove the major and minor gridlines
7. change the legend position to the right

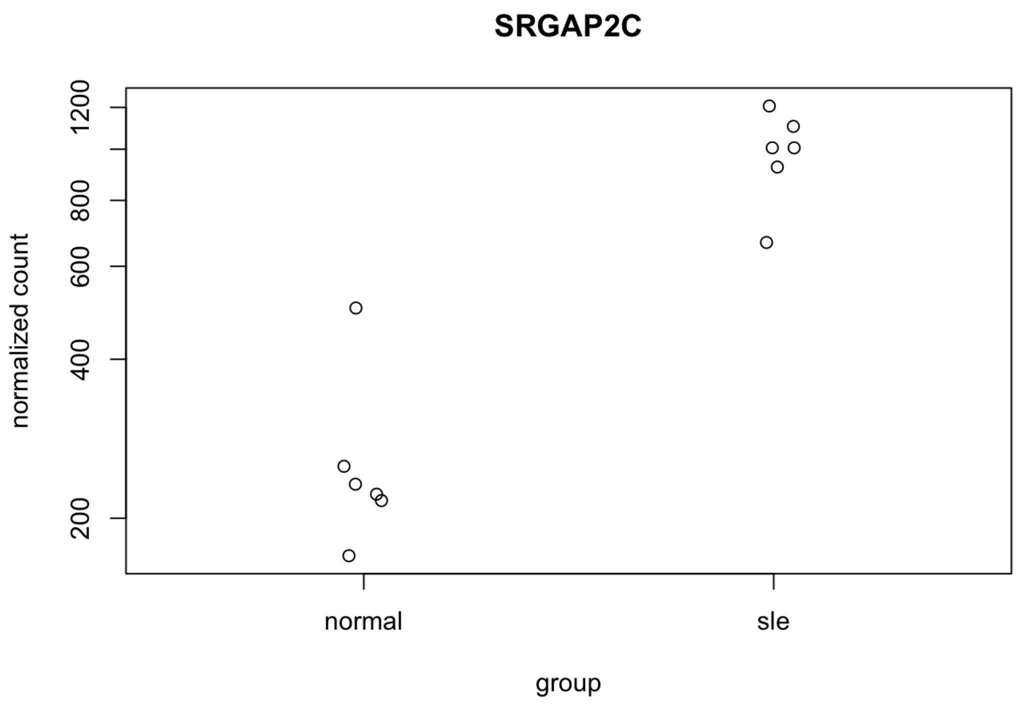
```
EnhancedVolcano(sle_n_res_data_gene,
                lab = as.character(sle_n_res_data_gene$symbol),
                x = 'log2FoldChange',
                y = 'padj',
                title = "SLE versus healthy controls",
                subtitle = "Differential expression",
                caption = paste0("Upregulated = 330, Downregulated = 222"),
                xlim = c(-10,10),
                ylim = c(0,20),
                FCcutoff = 1,
                pCutoff = 0.05,
                labSize = 4,
                axisLabSize = 12,
                col=c('black', 'black', 'black', 'red3'),
                legendLabels=c('Not sig.','Log2FC','padj', 'padj & Log2FC'),
                legendPosition = 'right',
                legendLabSize = 10,
                legendIconSize = 5.0,
                gridlines.major = FALSE,
                gridlines.minor = FALSE)
```



```
# plotting ZBTB16
plotCounts(dds, gene = "ENSG00000109906",
           intgroup = "Treatment",
           main = "ZBTB16")
```



```
# plotting SRGAP2C
plotCounts(dds, gene = "ENSG00000171943",
           intgroup = "Treatment",
           main = "SRGAP2C")
```



Citation

title: “Analyzing RNA-Seq with DESeq2” author: “Michael I. Love, Simon Anders, and Wolfgang Huber”

Note: if you use DESeq2 in published research, please cite:

> Love, M.I., Huber, W., Anders, S. (2014) Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biology*, **15**:550. 10.1186/s13059-014-0550-8 (http://dx.doi.org/10.1186/s13059-014-0550-8)