VI. Bioinformatics in R (presentation)

Center for Health Data Science, University of Copenhagen

07 September, 2021

Bioconductor

Bioconductor provides tools for computational biology and bioinformatics analysis in R - it is open source and open development and it has an active user community.
Mostly when we install R-packages we use `install.packages('name_of_package')`. When we use this command we refer to the CRAN repository of packages, however sometimes we want a package from `Bioconductor` instead. For this we use the command `BiocManager::install('name_of_package')`. In order to use this installer, you need to download the R-package `BiocManager` e.g. `install.packages('BiocManager')`.

Gene Expression Analysis in R with `DEseq2`

DEseq2 is one of the many packages/frameworks which exists for analysis of bulk gene expression data in R. For more information on DEseq2, please have a look at the original publication here.

Other highly used packages for differential expression analysis *DEA* are:

- limma
- edgeR
- NOIseq

`DEseq2` has many advantages over classical models and post hoc tests, as it is specifically developed for handling common issues and biases in expression data, including differences in sequencing depth and highly variable dispersion of counts between genes.
In brief, `DEseq2` fits a generalized linear model (GLM) for each gene in the dataset. In the case where we compare two groups i.e. treatment vs control, the GLM fit returns coefficients indicating the overall expression strength of a gene, along with the log2 fold change between groups. `DEseq2` adjusts variable gene dispersion estimates using an empirical Bayes approach which borrows information across genes and shrinks gene-wise dispersions towards a common dispersion trend to increase accuracy of differential expression testing.

About the Dataset

The dataset used for this presentation was acquired from the following github tutorial on RNAseq analysis: https://combine-australia.github.io/RNAseq-R/06-rnaseq-day1.html.

RNA sequencing data generated from luminal and basal cell sub-populations in the mammary gland of three groups of mice:
- Control
- Pregnant
- Lactating

The objective of the original study (found here) was to identify genes specifically expressed in lactating mammary glands, the gene expression profiles of luminal and basal cells from different developmental stages were compared.

Load R-packages:

```
# Data Wrangling
# install.packages("tidyverse")
# install.packages("readxl")
library(tidyverse)
library(readxl)

# For Plotting
# install.packages("ggplot2")
library(ggplot2)


# For DEA
# install.packages("BiocManager")
# BiocManager::install("DESeq2")
library(BiocManager)
library(DESeq2)
```

Importing Data

Reading in data:

```
exprDat <- read_excel("MouseRNAseq.xlsx")
exprInfo <- read_excel("MouseSampleInfo.xlsx")

# Look at the data:
head(exprDat, n=5)
```

```
## # A tibble: 5 x 14
##   EntrezGeneID GeneName MCL1.DG MCL1.DH MCL1.DI MCL1.DJ MCL1.DK MCL1.DL MCL1.LA
##   <chr>        <chr>      <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 497097       Xkr4         438     300      65     237     354     287       0
## 2 19888        Rp1            1       1       0       0       0       0      10
## 3 20671        Sox17        106     182      82     105      43      82      16
## 4 27395        Mrpl15       309     234     337     300     290     270     560
## 5 18777        Lypla1       652     515     948     935     928     791     826
## # ... with 5 more variables: MCL1.LB <dbl>, MCL1.LC <dbl>, MCL1.LD <dbl>,
## #   MCL1.LE <dbl>, MCL1.LF <dbl>
```

```
dim(exprDat)
```

```
## [1] 23151    14
```

```
head(exprInfo)
```

```
## # A tibble: 6 x 4
##   SampleName CellType Status   CellType.colors
##   <chr>      <chr>    <chr>    <chr>
## 1 MCL1.DG    basal    control  #79ADDC
## 2 MCL1.DH    basal    control  #79ADDC
## 3 MCL1.DI    basal    pregnant #79ADDC
## 4 MCL1.DJ    basal    pregnant #79ADDC
## 5 MCL1.DK    basal    lactate  #79ADDC
## 6 MCL1.DL    basal    lactate  #79ADDC
```

Convert character columns to factor types:

```
exprInfo <- exprInfo %>%
  mutate(CellType = as.factor(CellType),
         Status = as.factor(Status))

head(exprInfo)
```

```
## # A tibble: 6 x 4
##   SampleName CellType Status   CellType.colors
##   <chr>      <fct>    <fct>    <chr>
## 1 MCL1.DG    basal    control  #79ADDC
## 2 MCL1.DH    basal    control  #79ADDC
## 3 MCL1.DI    basal    pregnant #79ADDC
## 4 MCL1.DJ    basal    pregnant #79ADDC
## 5 MCL1.DK    basal    lactate  #79ADDC
## 6 MCL1.DL    basal    lactate  #79ADDC
```

Initial Data Check & Filtering:

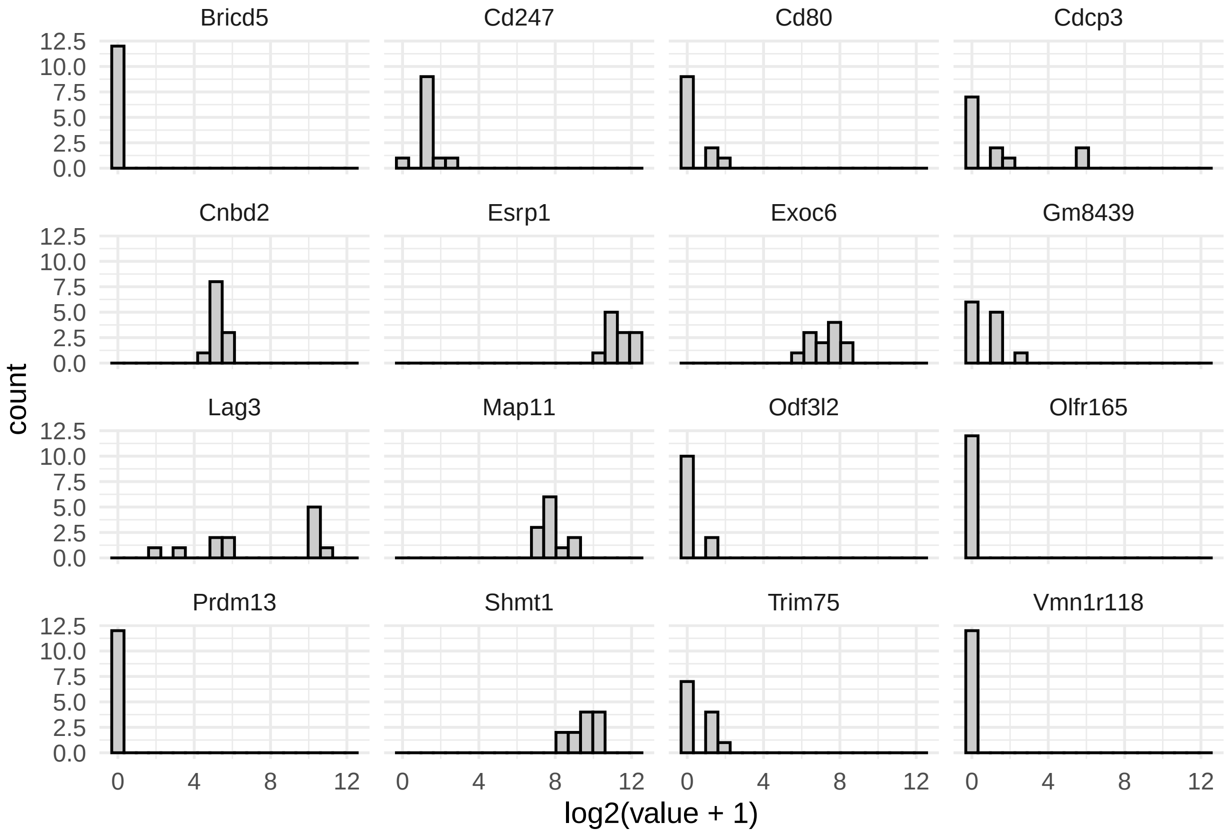
Let's try to sample 16 (n) random genes and plot their count distribution.

```
# Sample 16 random rows
expr16 <- exprDat %>%
  sample_n(.,16)

# Extract genenames
GeneName <- expr16$GeneName

# Gather counts
expr16 <- expr16 %>%
  dplyr::select(-EntrezGeneID, -GeneName) %>%
  t() %>%
  as_tibble() %>%
  rename_at(vars(names(.)), ~GeneName) %>%
  gather()

# Give it a look:
expr16
```

```
## # A tibble: 192 x 2
##    key   value
##    <chr> <dbl>
##   1 Exoc6   196
##   2 Exoc6   166
##   3 Exoc6    75
##   4 Exoc6    52
##   5 Exoc6    81
##   6 Exoc6    84
##   7 Exoc6   393
##   8 Exoc6   358
##   9 Exoc6   256
##  10 Exoc6   198
## # ... with 182 more rows
```

Plot:

```
ggplot(expr16, aes(log2(value+1))) +
  geom_histogram(color="black", fill="grey80", bins=20) +
  theme_minimal() +
  facet_wrap(~key)
```



We will filter out genes with too many zero counts. We will exclude the columns with gene information:

```
# Count number of 0s across samples. Filter samples where at least four samples has a count great than

exprDat <- exprDat %>%
  mutate(nzeros = rowSums(dplyr::select(.,-EntrezGeneID, -GeneName)==0)) %>%
  filter(nzeros <= 8) %>%
  dplyr::select(-nzeros)
```

```
# How many genes do we have left:
dim(exprDat)
```

```
## [1] 17308    14
```

Differential Expression Analysis- DESeq2

We will now make a DESeq2 object. For this we use the function `DESeqDataSetFromMatrix` from the DEseq2 package. As input we give our count matrix, our gene IDs and our meta data (exprInfo). Additionally we include a design for DE contrasts. In this case we add CellType (luminal or basal) and Status (control, pregnant or lactating).

Convert to exprDat to a dataframe and make GeneNames column into rownames:

```
# Pull out GeneNames and EntrezGeneID for later use
GeneNames <- exprDat %>%
  dplyr::select(EntrezGeneID, GeneName)



exprDat <- exprDat %>%
  dplyr::select(-EntrezGeneID) %>%
  column_to_rownames(., var = "GeneName")
```

Make a DESeq2 object:

```
exprObj <- DESeqDataSetFromMatrix(countData = exprDat,
                                  colData = exprInfo,
                                  design= ~CellType+Status)
exprObj
```

```
## class: DESeqDataSet
## dim: 17308 12
## metadata(1): version
## assays(1): counts
## rownames(17308): Xkr4 Rp1 ... Uty Gm47283,
## rowData names(0):
## colnames(12): MCL1.DG MCL1.DH ... MCL1.LE MCL1.LF
## colData names(4): SampleName CellType Status CellType.colors
```

Preliminary analysis:

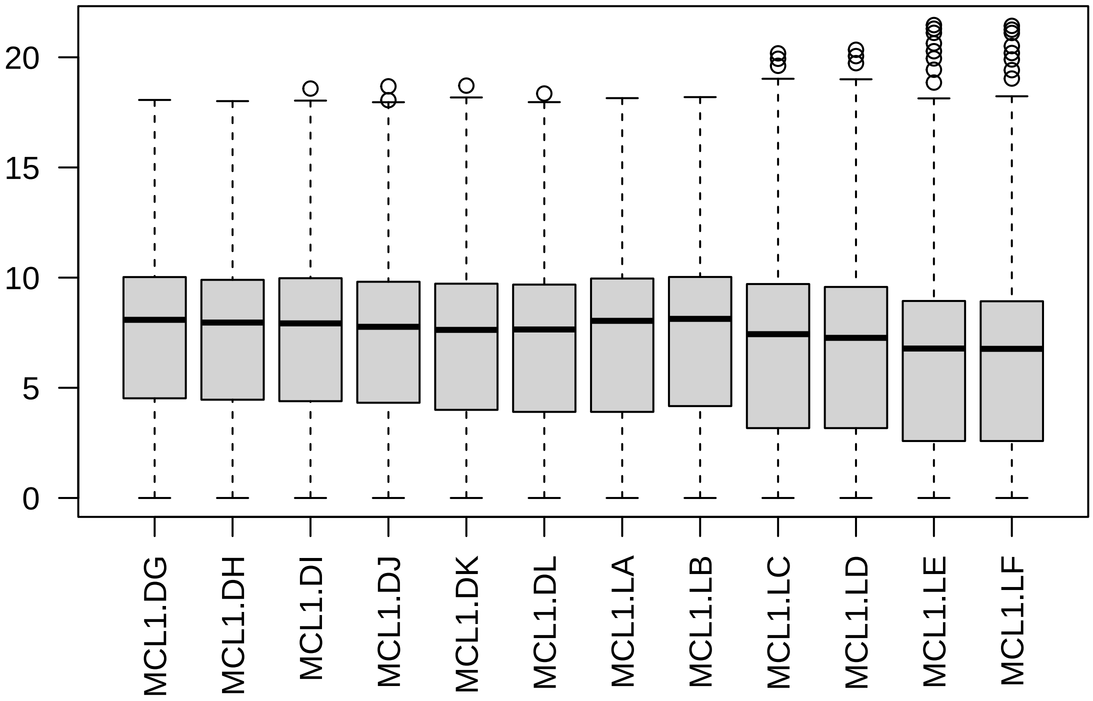
Let's have a look at the library sizes:

```
colSums(assay(exprObj))
```

```
##  MCL1.DG  MCL1.DH  MCL1.DI  MCL1.DJ  MCL1.DK  MCL1.DL  MCL1.LA  MCL1.LB
## 22634514 21155013 23488082 22100122 21057113 19583106 19698631 20944796
##  MCL1.LC  MCL1.LD  MCL1.LE  MCL1.LF
## 21675050 21457888 24419457 24366629
```

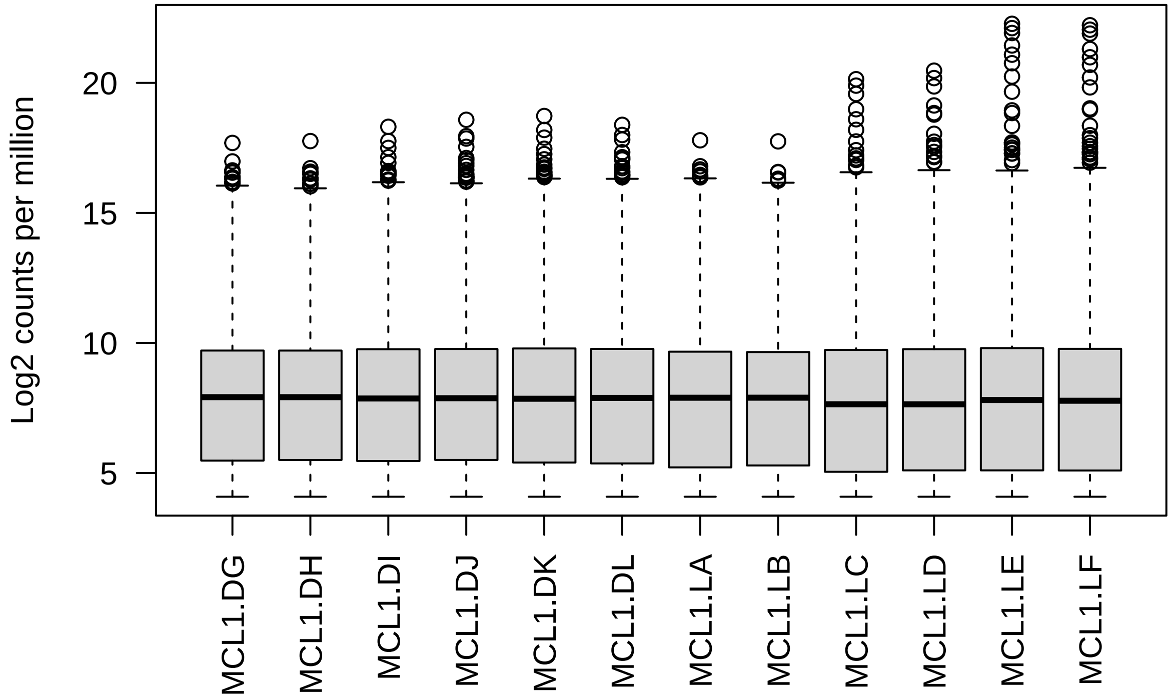
The count distributions may be dominated by a few genes with very large counts. These genes will drive plotting e.g. heatmaps, PCA analysis etc. Let's see if we have any "outlier" genes in our dataset and at the same time inspect the sample library sizes. For convenience I am using the base R boxplot function:

```r
#boxplot(assay(exprObj), las=2)
boxplot(log2(assay(exprObj)+1), las=2)
```



We perform variance stabilizing transformation to obtain log2 counts per million read mapped, overcoming issues with outlier genes and sequencing depth:

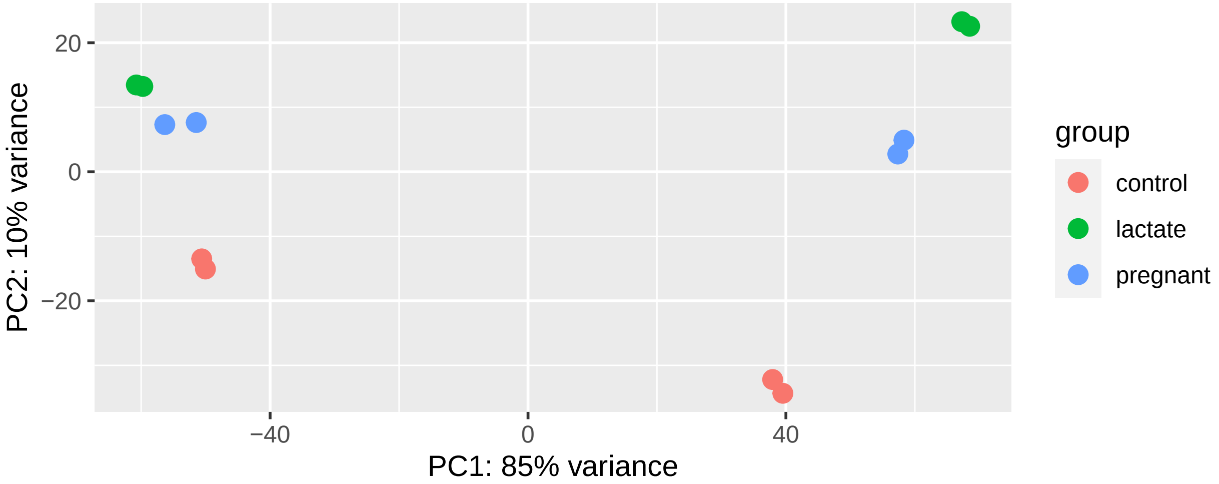
```r
exprObjvst <- vst(exprObj,blind=FALSE)
boxplot(assay(exprObjvst), xlab="", ylab="Log2 counts per million",las=2)
```



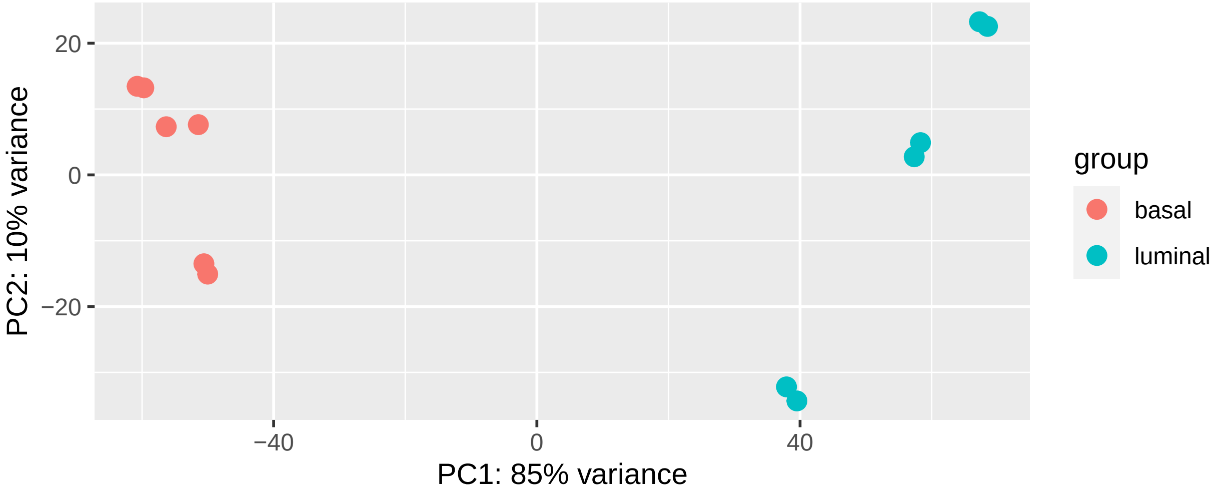
Principal Component Analysis

Before performing DEA it is a good idea to explore how samples cluster together based on there gene expression profile. The expectation here is that samples from the same group (treatment vs control, condition A vs condition B, etc.) will cluster together. A principal component analysis (PCA) plot can also help us to identify outlier samples which might need to be removed from the analysis. We use our vst counts for principal component analysis:

```
plotPCA(exprObjvst,intgroup="Status")
```



```
plotPCA(exprObjvst,intgroup="CellType")
```



DESeq function for DEA

Next, we use `DEseq()` to estimate dispersion, gene-wise and mean-dispersion, fitting model(s):

```
exprObj <- DESeq(exprObj)
```

Testing

Have a look at the group comparisons:

```
resultsNames(exprObj)
```

```
## [1] "Intercept"                  "CellType_luminal_vs_basal"
## [3] "Status_lactate_vs_control"  "Status_pregnant_vs_control"
```

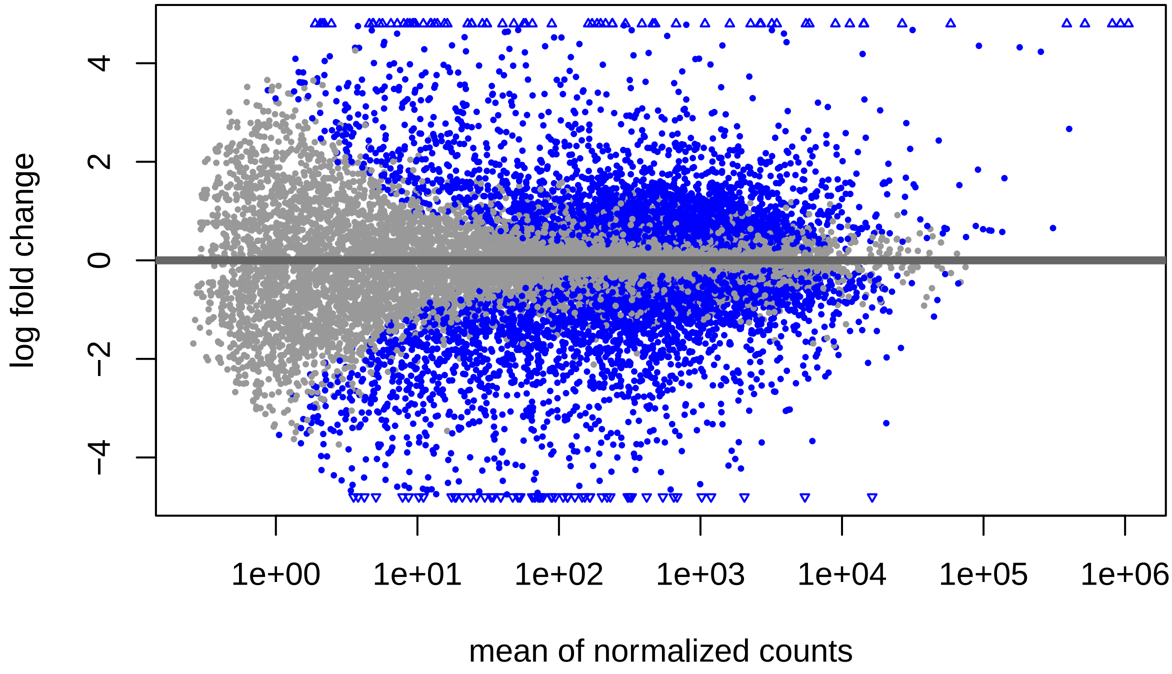
Test for DE genes between the three groups of mice, adjusted for cell type:

(I) lactating and control mice:

```
resLC <- results(exprObj, contrast = c("Status", "lactate", "control"), independentFiltering = FALSE)
```

Summary and plot of DE analysis results:

```
DESeq2::plotMA(resLC)
```



```
summary(resLC)
```

```
##
## out of 17308 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)       : 3550, 21%
## LFC < 0 (down)     : 3474, 20%
## outliers [1]       : 0, 0%
## low counts [2]     : 0, 0%
## (mean count < 0)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

Below we perform the same steps as above to get the DE genes between (II) pregnant and control mice and (III) lactating and pregnant mice:

(II) pregnant and control mice:

```
resPC <- results(exprObj, contrast = c("Status", "pregnant", "control"),  independentFiltering = FALSE)

#DESeq2::plotMA(resPC)
#summary(resPC)
```

(III) lactating and pregnant mice:

```
resLP <- results(exprObj, contrast = c("Status", "lactate", "pregnant"),  independentFiltering = FALSE)

#DESeq2::plotMA(resLP)
#summary(resLP)
```

We filter the results of the DEA to only include those genes which are differentially expressed based on logFC (>= 1.0 or <= -1.0) and adjusted p-value (< 0.01).
Firstly, bind the three DE genesets together and convert to a tibble. Then, add a column with GeneNames. Lastly, filter and arrange rows (genes) based on logFC and adjusted p-values.

```
resDE <- rbind(resLC, resPC, resLP) %>%
  as_tibble() %>%
  mutate(GeneName = rep(GeneNames$GeneName, 3)) %>%
  filter((log2FoldChange >= 1.0 | log2FoldChange <= -1.0) & padj <= 0.01) %>%
  arrange(padj, desc(abs(log2FoldChange)))

# DE genes
dim(resDE)
```

```
## [1] 4122    7
```

```
length(unique(resDE$GeneName))
```

```
## [1] 2792
```

Heatmap Visualization

To visually inspect if DE genes identified in our DESeq2 analysis successfully separate the three groups of mice (control, pregnant and lactating), we will make a heatmap. For this we use the `heatmap` function.

It will not make sense to include all DE genes in this heatmap (almost 3000 unique genes). Instead pick the top 100 most significant DE genes, based on adj. p-value and logFC.

Make a vector of unique GeneNames (top100):

```
# Make a vector of unique EntrezGeneIDs (top100):

top100 <- resDE[1:100,] %>%
  pull(GeneName) %>%
  unique()
```

```
length(top100)
```

```
## [1] 82
```

The expression counts themselves (not logFC) are needed for the heatmap. We use the topDE vector to extract these from the vst normalized DESeq2 object.

```
head(assay(exprObjvst), n=5)
```

```
##         MCL1.DG  MCL1.DH  MCL1.DI  MCL1.DJ  MCL1.DK  MCL1.DL  MCL1.LA  MCL1.LB
## Xkr4   8.536754 8.157576 6.401025 7.987204 8.608154 8.350271 4.088346 4.088346
## Rp1    4.395113 4.408625 4.088346 4.088346 4.088346 4.088346 5.049024 4.605457
## Sox17  6.823999 7.536586 6.631003 7.018342 6.198459 6.849080 5.290970 5.528569
## Mrpl15 8.084835 7.843863 8.288204 8.288468 8.346705 8.271139 8.881522 8.550341
## Lypla1 9.068321 8.867045 9.679391 9.821810 9.921005 9.717343 9.409346 9.384315
##         MCL1.LC  MCL1.LD  MCL1.LE   MCL1.LF
## Xkr4   4.088346 4.088346 4.088346  4.088346
## Rp1    5.155437 4.603001 4.088346  4.088346
## Sox17  5.496313 5.102135 4.880034  5.487699
## Mrpl15 8.993366 8.614985 9.157331  9.288091
## Lypla1 9.418084 9.531917 9.945641 10.020675
```

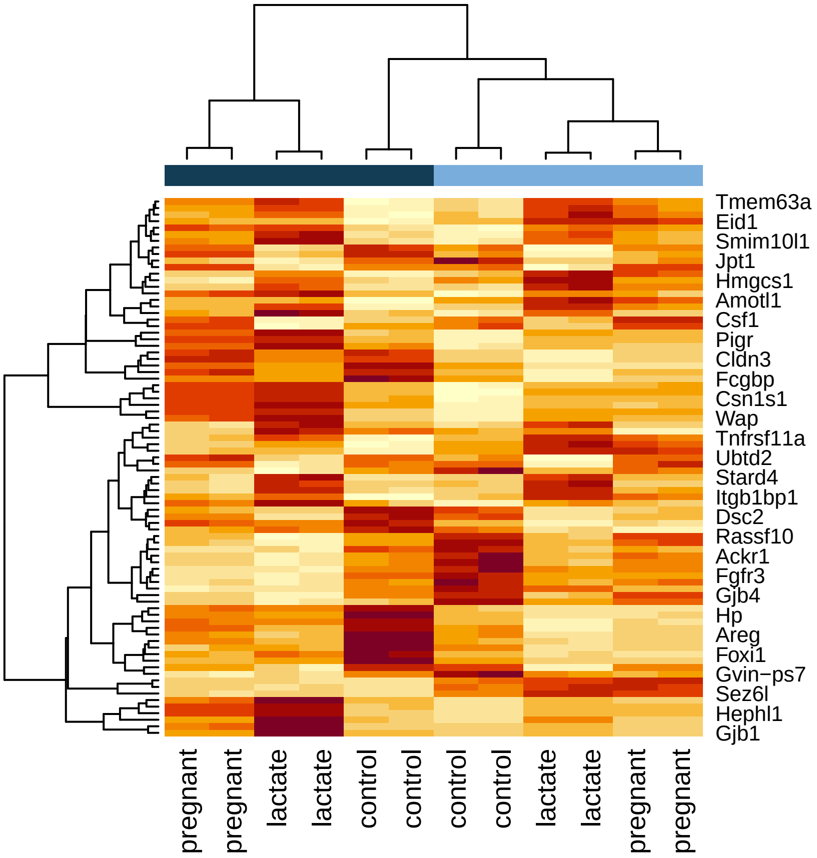
```
resVST <- assay(exprObjvst) %>%
  as.data.frame() %>%
  rownames_to_column(var = "GeneName")  %>%
  as_tibble() %>%
  filter(GeneName %in% top100)
```

The heatmap function in base R wants gene expression data as a matrix (a dataframe with numeric values only). We extract the GeneNames column and convert the tibble into a matrix:

```
HPnames <- resVST %>%
  pull(GeneName)


HPdat <- resVST %>%
  dplyr::select(-GeneName) %>%
  as.matrix()
```

We use the `heatmap` function to generate a heatmap. We can modify the look of the heatmap as desired, e.g. add column colors, row labels, change color scheme etc.

```
heatmap(HPdat,
        ColSideColors=exprInfo$CellType.colors,
        labCol=exprInfo$Status,
        labRow = HPnames,
        cexCol=1.2,
        cexRow = 1.0)
```


Tmem63a
Eid1
Smim10l1
Jpt1
Hmgcs1
Amotl1
Csf1
Pigr
Cldn3
Fcgbp
Csn1s1
Wap
Tnfrsf11a
Ubtd2
Stard4
Itgb1bp1
Dsc2
Rassf10
Ackr1
Fgfr3
Gjb4
Hp
Areg
Foxi1
Gvin−ps7
Sez6l
Hephl1
Gjb1
pregnant
pregnant
lactate
lactate
control
control
control
control
lactate
lactate
pregnant
pregnant