VI. Bioinformatics in R (presentation)

Center for Health Data Science, University of Copenhagen

14 March, 2022

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(DESeq2)
library(dplyr)
```

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 13
##   GeneName MCL1.DG MCL1.DH MCL1.DI MCL1.DJ MCL1.DK MCL1.DL MCL1.LA MCL1.LB
##   <chr>      <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 Xkr4         438     300      65     237     354     287       0       0
## 2 Rp1            1       1       0       0       0       0      10       3
## 3 Sox17        106     182      82     105      43      82      16      25
## 4 Mrpl15       309     234     337     300     290     270     560     464
## 5 Lypla1       652     515     948     935     928     791     826     862
## # ... with 4 more variables: MCL1.LC <dbl>, MCL1.LD <dbl>, MCL1.LE <dbl>,
## #   MCL1.LF <dbl>
```

```
dim(exprDat)
```

```
## [1] 23151    13
```

```
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)

expr16
```

```
## # A tibble: 16 x 13
##    GeneName   MCL1.DG MCL1.DH MCL1.DI MCL1.DJ MCL1.DK MCL1.DL MCL1.LA MCL1.LB
##    <chr>        <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
##  1 Itgad            0       1       3       2       1       0       0       0
##  2 Zfp60          102      51      41      43      67      87     168     182
##  3 Kbtbd8          57      78      36      15      23      13     113     117
##  4 Dnm3os          13       8       9       8       6       9       0       2
##  5 Cyp3a25          0       1       0       3       0       0      13      50
##  6 Rida           299     326     391     397     344     332     312     319
##  7 Coa4            22      23      34      22      25      20      44      51
##  8 Atp5md         123     169     202     184     145     125     246     241
##  9 Vmn1r100         0       0       0       0       0       0       0       0
## 10 Pla2g3          15      19      16      15       8       7       4      17
## 11 Parn           875     819     920     845     864     791     861     871
## 12 Dcdc2b         132     101     108     117     120     122     205     208
```

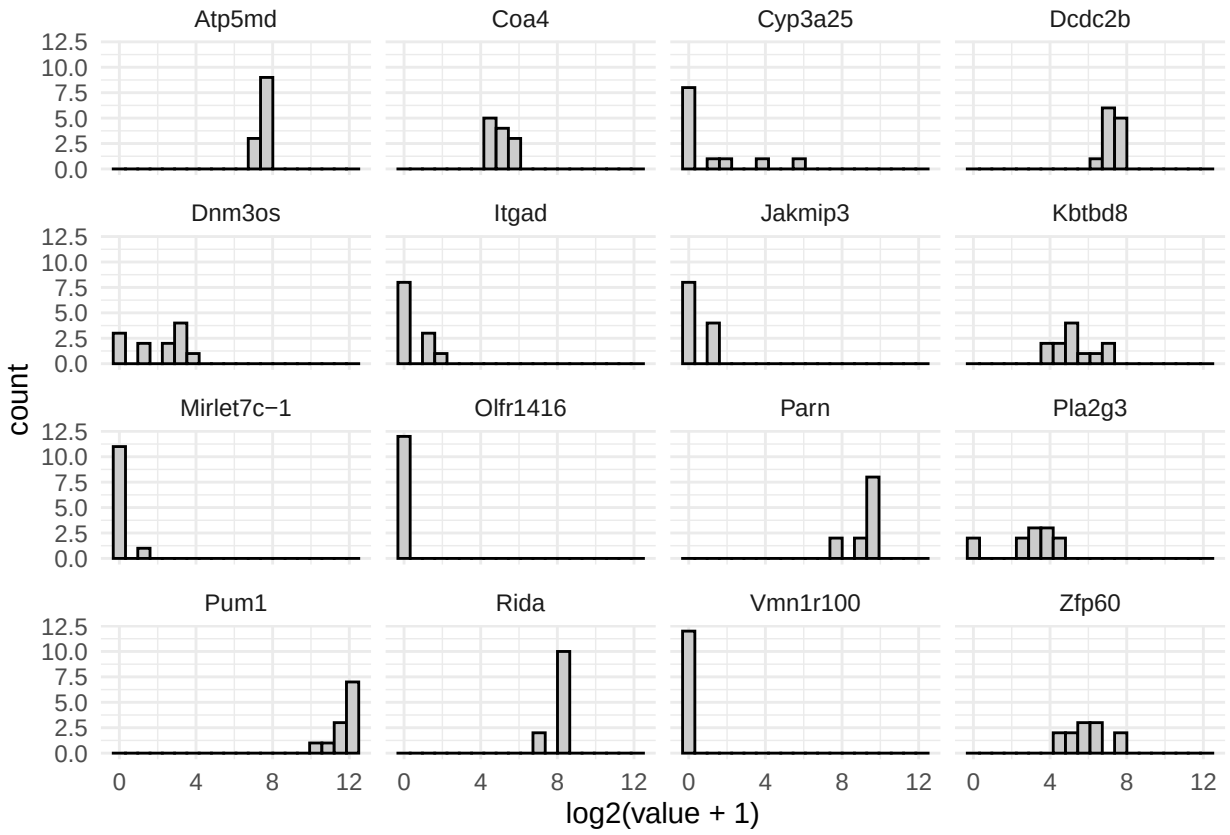
```
## 13 Mirlet7c-1       0       0       0       1       0       0       0       0
## 14 Jakmip3          0       0       1       0       1       0       0       0
## 15 Olfr1416         0       0       0       0       0       0       0       0
## 16 Pum1          4000    3686    4614    3784    2472    2403    4116    4064
## # ... with 4 more variables: MCL1.LC <dbl>, MCL1.LD <dbl>, MCL1.LE <dbl>,
## #   MCL1.LF <dbl>
```

```
# Gather counts
# Gather counts
expr16 <- expr16 %>%
  column_to_rownames(var = "GeneName") %>%
  t() %>%
  as_tibble() %>%
  gather()

# Give it a look:
expr16
```

```
## # A tibble: 192 x 2
##    key   value
##    <chr> <dbl>
##  1 Itgad     0
##  2 Itgad     1
##  3 Itgad     3
##  4 Itgad     2
##  5 Itgad     1
##  6 Itgad     0
##  7 Itgad     0
##  8 Itgad     0
##  9 Itgad     0
## 10 Itgad     0
## # ... 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 low expressed genes. There are many strategies for doing so, but here we will filter out genes that have less than 3 counts in at least n samples. We select n as the smallest number of biologically meaningful groups. In this case, it is 3.

```
table(exprInfo$CellType, exprInfo$Status)
```

```
##
##           control lactate pregnant
##   basal         2       2        2
##   luminal       2       2        2
```

```
# 2 samples in each group
```

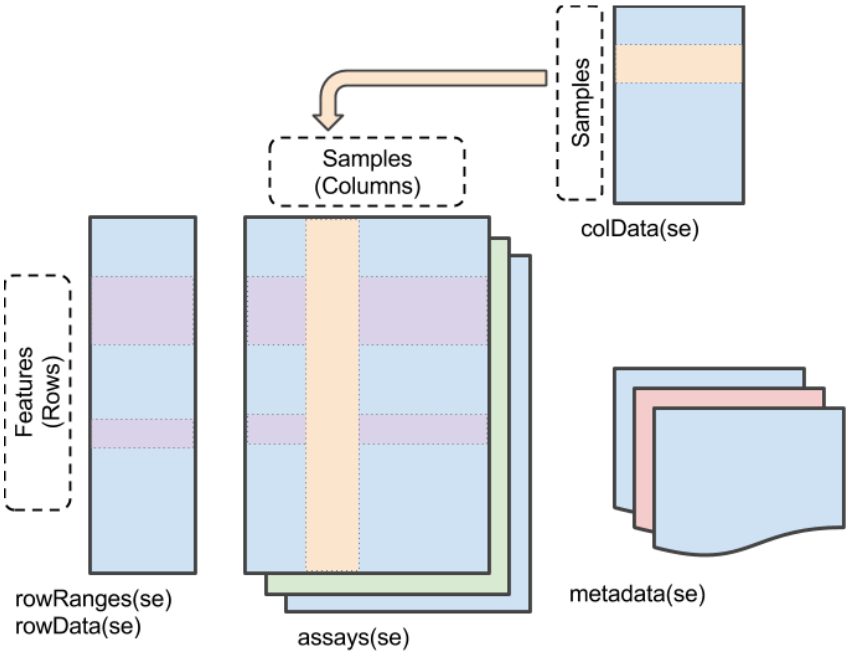
```
# Count number of samples with min. count size of 5 for a given gene.
# Filter for genes were min. 4 samples have a count equal to or greater than 5.

exprDat <- exprDat %>%
  mutate(lcount = rowSums(dplyr::select(.,-GeneName) >= 5)) %>%
  filter(lcount >= 4) %>%
  dplyr::select(-lcount)

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

```
## [1] 15123    13
```

Differential Expression Analysis- DESeq2

We will now make a DESeq2 object. For this we use the function `DESeqDataSetFromMatrix` from the DEseq2 package.

DESeq object is a type of SummarizedExperiment container used to store the input values, intermediate calculations and results of an analysis of differential expression. The rows typically represent Genes (genomic ranges) of interest and the columns represent samples.



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

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

exprDat <- exprDat %>%
  column_to_rownames(., var = "GeneName")
```

Make a DESeq2 object: 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).

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

```
## class: DESeqDataSet
## dim: 15123 12
## metadata(1): version
## assays(1): counts
## rownames(15123): Xkr4 Sox17 ... G530011O06Rik Gm47283,
## rowData names(0):
## colnames(12): MCL1.DG MCL1.DH ... MCL1.LE MCL1.LF
```

```
## colData names(4): SampleName CellType Status CellType.colors
```

Preliminary analysis:

There are multiple biases in RNAseq experiment: library size, genes length, genes GC composition, etc. Library size is the most well-known bias. For the purpose of DEA - genes length and GC composition are not so important because it is supposed to be about the same for the gene across different samples.

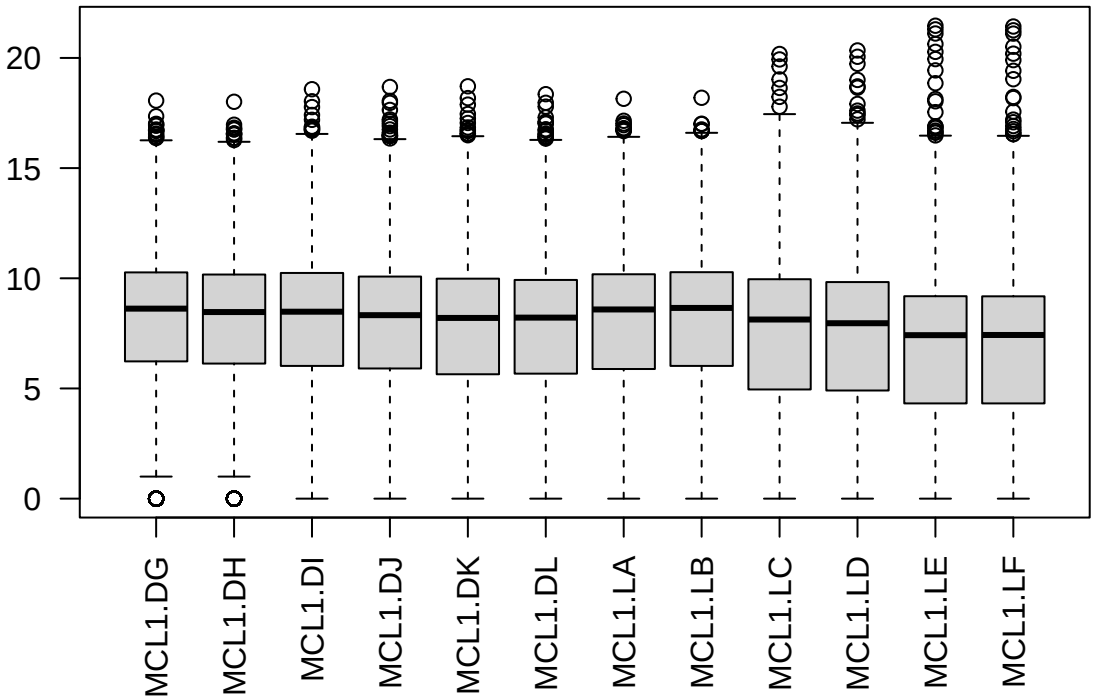
Let's have a look at the library sizes:

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

```
##  MCL1.DG  MCL1.DH  MCL1.DI  MCL1.DJ  MCL1.DK  MCL1.DL  MCL1.LA  MCL1.LB
## 22629791 21149973 23484111 22096278 21053999 19580151 19692421 20935934
##  MCL1.LC  MCL1.LD  MCL1.LE  MCL1.LF
## 21672174 21455126 24416985 24363987
```

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:

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



As you can see we do not have any extreme outliers, but we do see some differences between libraries.

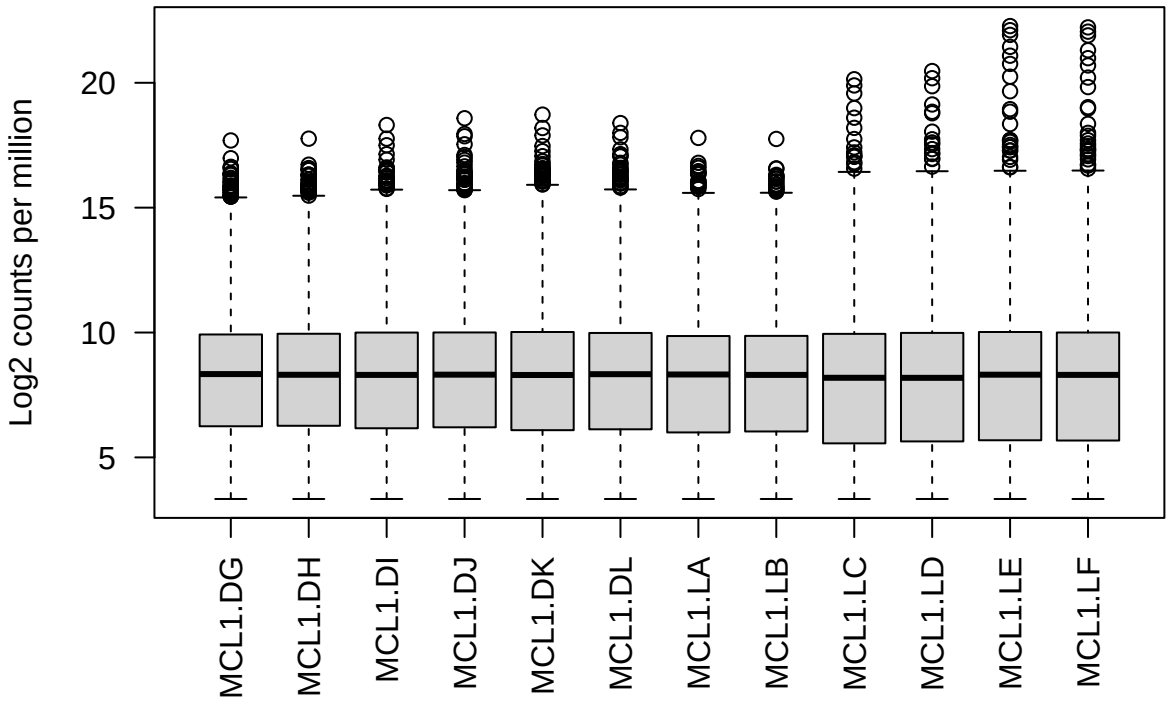
Next, we will apply the "vst" function to do a couple of things

- normalize library size to obtain counts per million mapped reads
- log2 transform the data to get more normally distributed data
- apply variance stabilizing transformation which we will discuss below.

```
exprObjvst <- vst(exprObj,blind=FALSE)
```

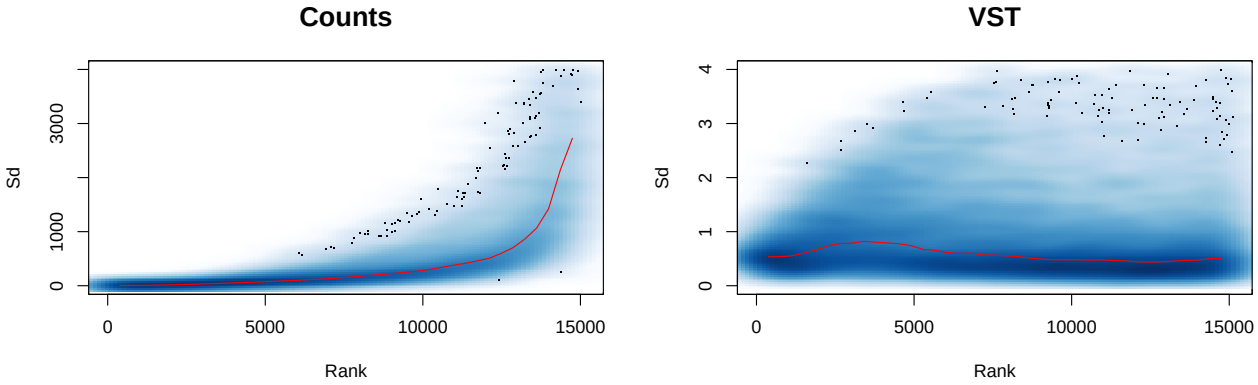
Let's plot normalized data.

```
par(mfrow=c(1,1))
boxplot(assay(exprObjvst), xlab="", ylab="Log2 counts per million",las=2)
```



Variance stabilizing transformation:

In RNA-Seq data, genes with larger average expression have on average larger observed variances (sd) across samples. This is know as data heteroscedasticity. Expression varies from sample to sample more than other genes with lower average expression.

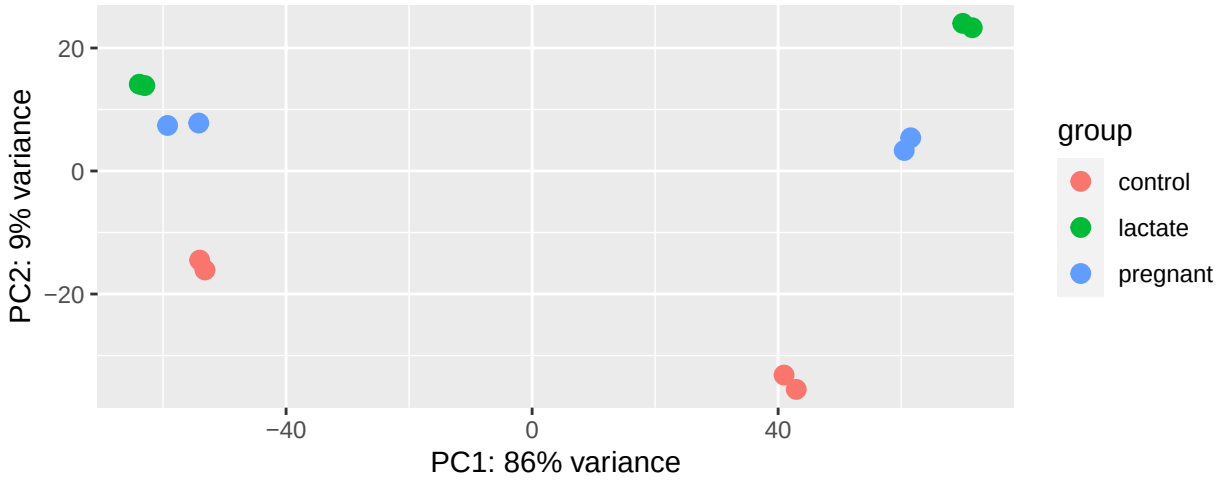


Principal Component Analysis

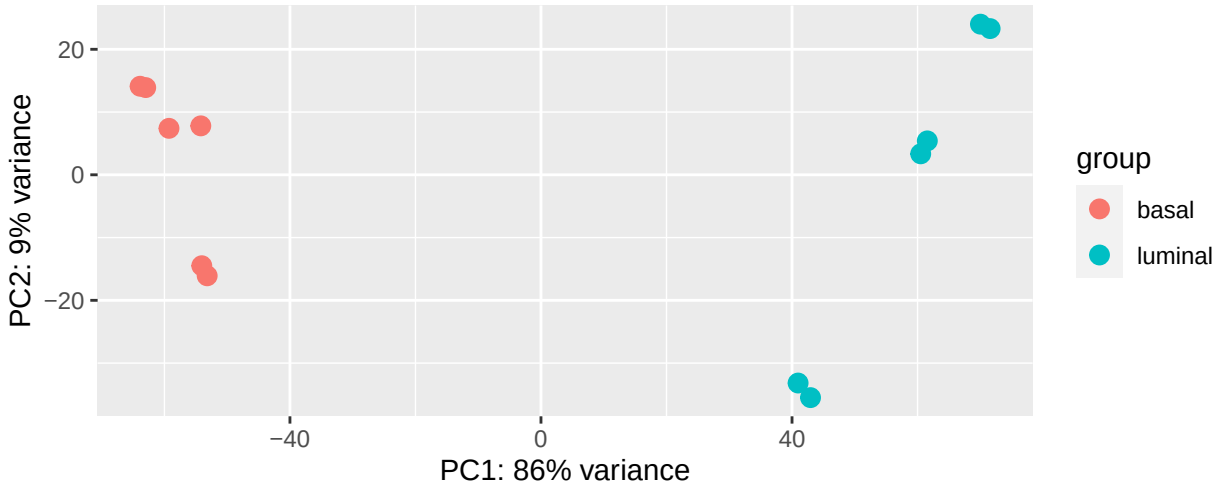
Before performing DEA it is a good idea to explore how samples cluster together based on there gene expression profile. The expectation here is that samples from the same group (treatment vs control, condition A vs condition B, etc.) will cluster together. A principal component analysis (PCA) plot can 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"
```

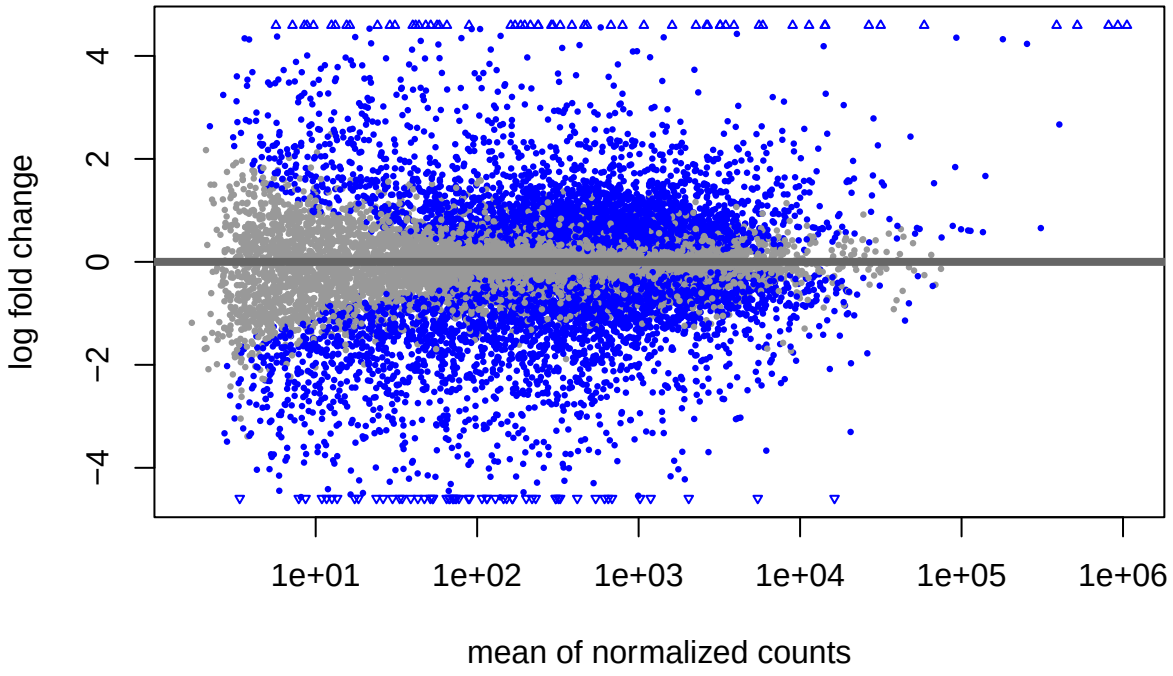
Test for DE genes between the three groups of mice, adjusted for cell type:

(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 15123 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)       : 3566, 24%
## LFC < 0 (down)     : 3497, 23%
## 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:

Convert DEseq2 object to tibble for further analysis:

```
resLC <- resLC %>%
  as.data.frame() %>%
  rownames_to_column(., var = "GeneName")
```

(II) pregnant and control mice:

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

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

resPC <- resPC %>%
  as.data.frame() %>%
  rownames_to_column(., var = "GeneName")
```

(III) lactating and pregnant mice:

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

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

resLP <- resLP %>%
  as.data.frame() %>%
  rownames_to_column(., var = "GeneName")
```

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 indicating in which parwise comparison the gene was DE Lastly. Filter rows (genes) based on logFC and adjusted p-values.

```
resDE <- rbind(resLC, resPC, resLP) %>%
  as_tibble() %>%
  mutate(pair = c(rep("Lactate.Control", nrow(exprDat)),
                  rep("Pregnant.Control", nrow(exprDat)),
                  rep("Lactate.Pregnant", nrow(exprDat)))) %>%
  filter((log2FoldChange >= 1.0 | log2FoldChange <= -1.0) & padj <= 0.05)


# Give it a look:
resDE
```

```
## # A tibble: 5,647 x 8
##    GeneName      baseMean log2FoldChange lfcSE  stat   pvalue     padj pair
##    <chr>            <dbl>          <dbl> <dbl> <dbl>    <dbl>    <dbl> <chr>
##  1 Rgs20            46.0           -1.39 0.567 -2.46 1.38e- 2 3.73e- 2 Lactate.~
##  2 Pcmtd1         1469.             1.33 0.179  7.45 9.43e-14 4.16e-12 Lactate.~
##  3 Adhfe1          118.             2.33 0.541  4.31 1.64e- 5 1.23e- 4 Lactate.~
##  4 2610203C22Rik    17.0            1.70 0.610  2.78 5.36e- 3 1.73e- 2 Lactate.~
##  5 Vxn              21.3            2.36 0.540  4.38 1.20e- 5 9.28e- 5 Lactate.~
##  6 Mybl1           254.            -1.00 0.266 -3.78 1.60e- 4 8.88e- 4 Lactate.~
##  7 A830018L16Rik     3.56          -3.24 1.31  -2.46 1.38e- 2 3.73e- 2 Lactate.~
##  8 Slco5a1          10.7           -1.72 0.703 -2.45 1.42e- 2 3.80e- 2 Lactate.~
##  9 Lactb2          243.             1.13 0.197  5.72 1.04e- 8 1.75e- 7 Lactate.~
## 10 Eya1            197.             1.01 0.309  3.25 1.14e- 3 4.77e- 3 Lactate.~
## # ... with 5,637 more rows
```

Number of DE genes:

```
# Number of DE genes:
dim(resDE)
```

```
## [1] 5647    8
```

```
# Number of unique DE genes:
resDE %>%
  pull(GeneName) %>%
  unique() %>%
  length()
```

```
## [1] 3642
```