VI. Bioinformatics in R (presentation)

Center for Health Data Science, University of Copenhagen

21 October, 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(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 Olfr601        0       0       0       0       0       0       0       0
##  2 Tas2r115       0       0       0       0       0       0       0       0
##  3 Olfr1140       0       0       0       0       0       0       0       0
##  4 Smim3       5047    7044    5552    4584    1895    1274    1816    2014
##  5 Cenpw         41      59      35      45      18      30      75      98
##  6 Rims1          2      10       4       0       5       3       0       0
##  7 Cacng2         0       0       0       0       0       0       0       0
##  8 Kynu           0       0       1       0       0       0       6      17
##  9 Zbtb43       822     817     703     603     425     330    1909    1888
## 10 Agxt2          0       0       1       0       0       0      35      46
## 11 Cep55        168     161     137     119      21      13     231     263
## 12 Pet117         0       0       0       0       0       0       0       0
## 13 Ccdc82       730     580     569     609     986     730     346     335
```

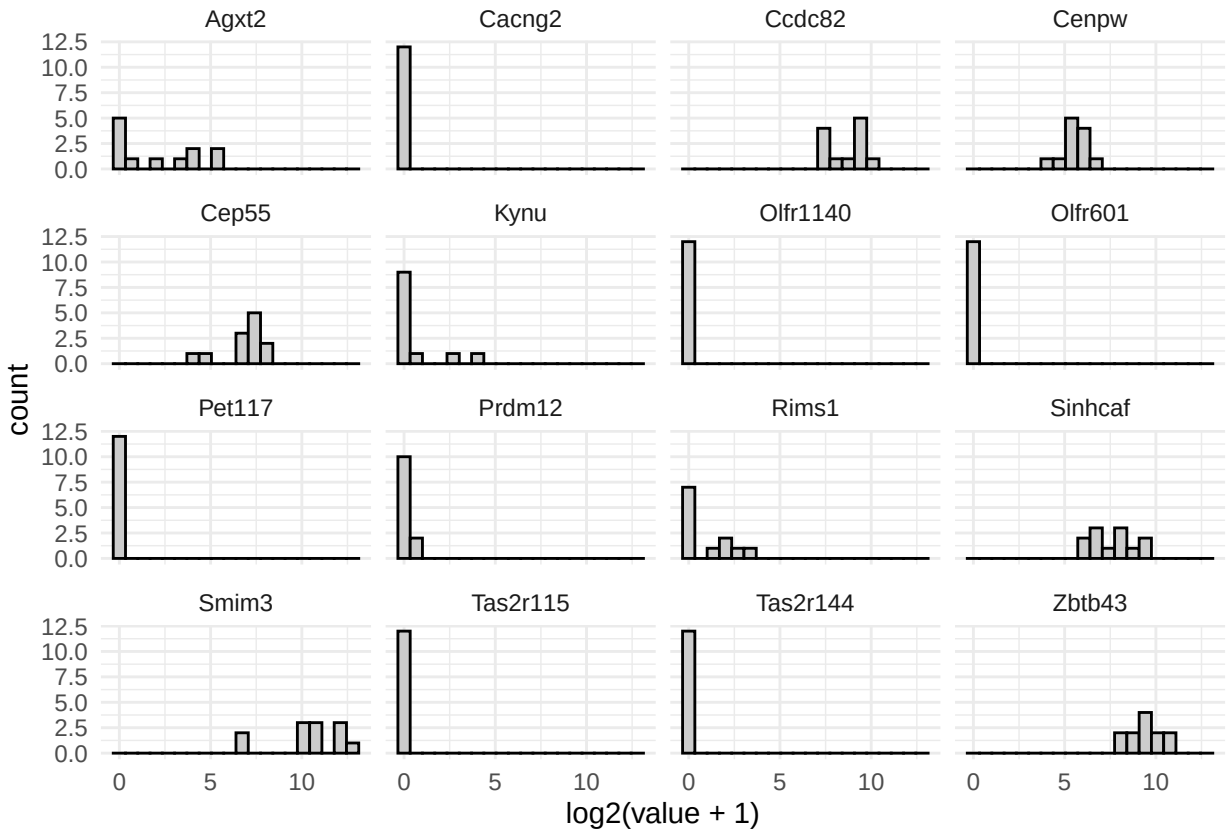
```
## 14 Prdm12         1       0       0       0       0       0       0       0
## 15 Sinhcaf      445     287     196     116      97     106     585     564
## 16 Tas2r144       0       0       0       0       0       0       0       0
## # ... with 4 more variables: MCL1.LC <dbl>, MCL1.LD <dbl>, MCL1.LE <dbl>,
## #   MCL1.LF <dbl>
```

```
# Extract genenames
GeneName <- expr16$GeneName

# Gather counts
expr16 <- expr16 %>%
  dplyr::select(-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 Olfr601     0
##  2 Olfr601     0
##  3 Olfr601     0
##  4 Olfr601     0
##  5 Olfr601     0
##  6 Olfr601     0
##  7 Olfr601     0
##  8 Olfr601     0
##  9 Olfr601     0
## 10 Olfr601     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 2.

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

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

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

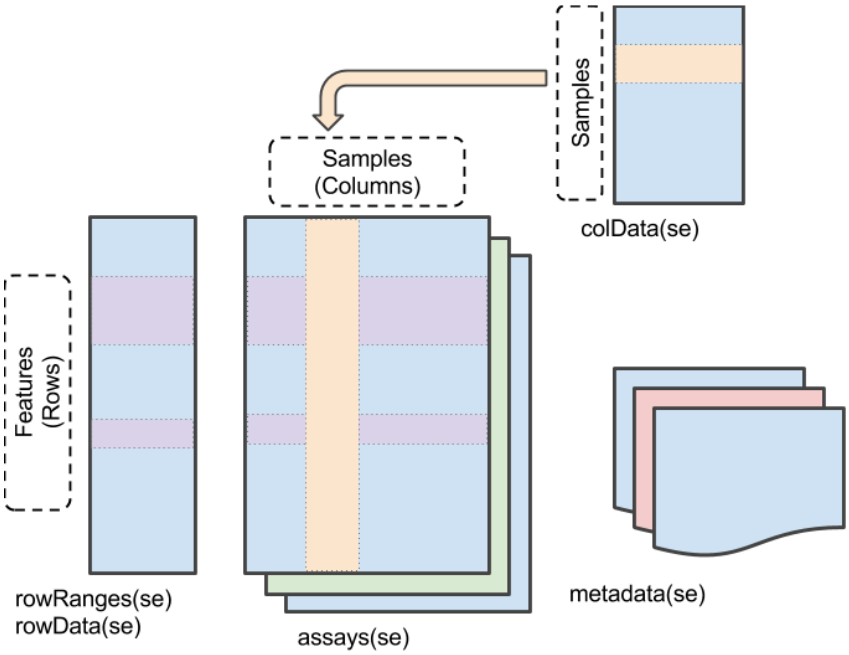
```
# First, we count the number of times a count value in a sample is greater or equal to 3. Then Filter r

exprDat <- exprDat %>%
  mutate(nzeros = rowSums(dplyr::select(.,-GeneName)<=3)) %>% # count number of
  filter(nzeros <= 2) %>%
  dplyr::select(-nzeros)

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

```
## [1] 13165    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: 13165 12
## metadata(1): version
## assays(1): counts
## rownames(13165): Sox17 Mrpl15 ... Mid1 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.

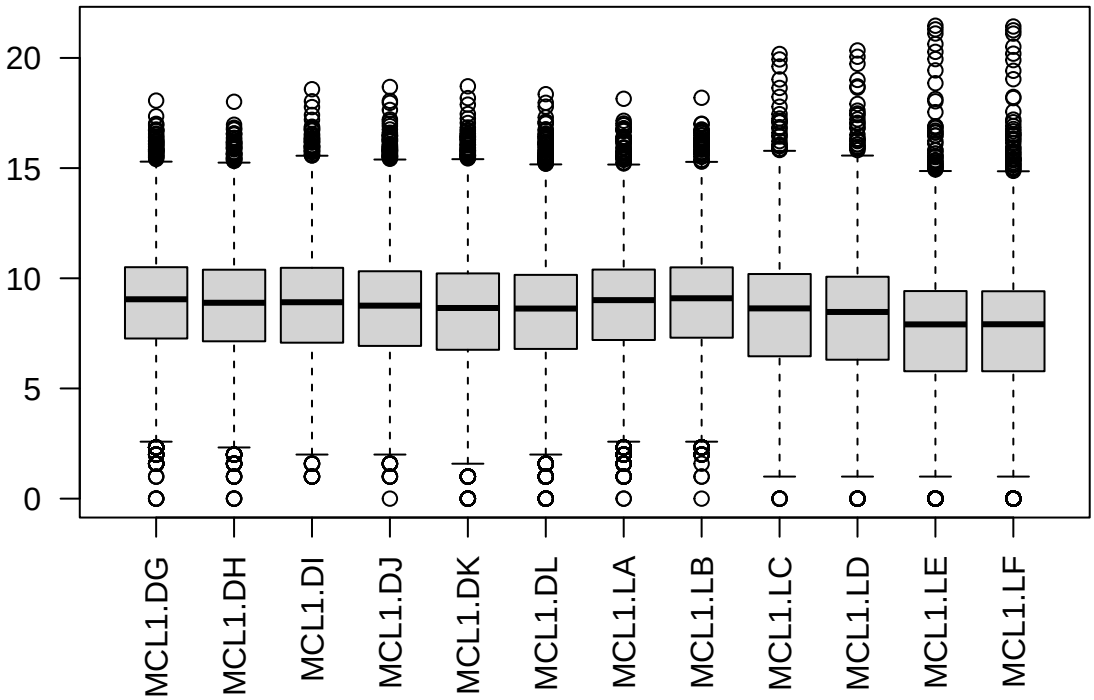
Let's have a look at the library sizes:

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

```
##  MCL1.DG  MCL1.DH  MCL1.DI  MCL1.DJ  MCL1.DK  MCL1.DL  MCL1.LA  MCL1.LB
## 22523486 21043343 23372291 21984207 20937716 19465788 19646462 20876895
##  MCL1.LC  MCL1.LD  MCL1.LE  MCL1.LF
## 21652828 21436656 24394414 24341858
```

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.

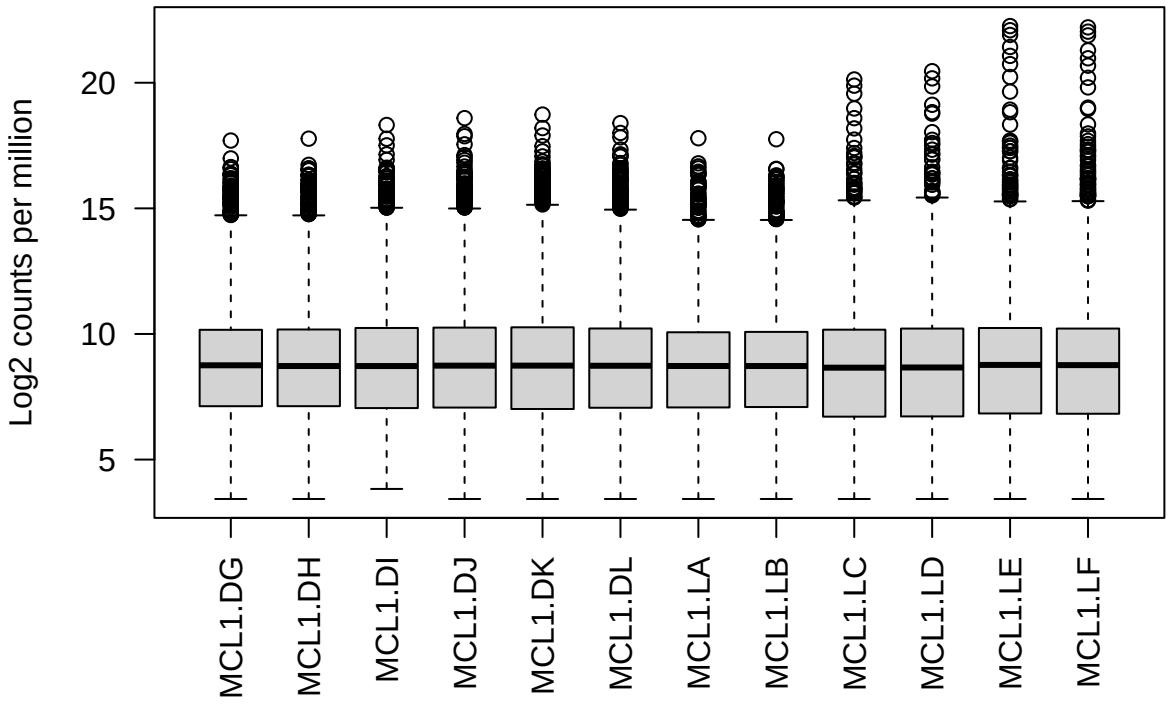
Next, we will apply the "vst" function to do a couple of things

- 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.

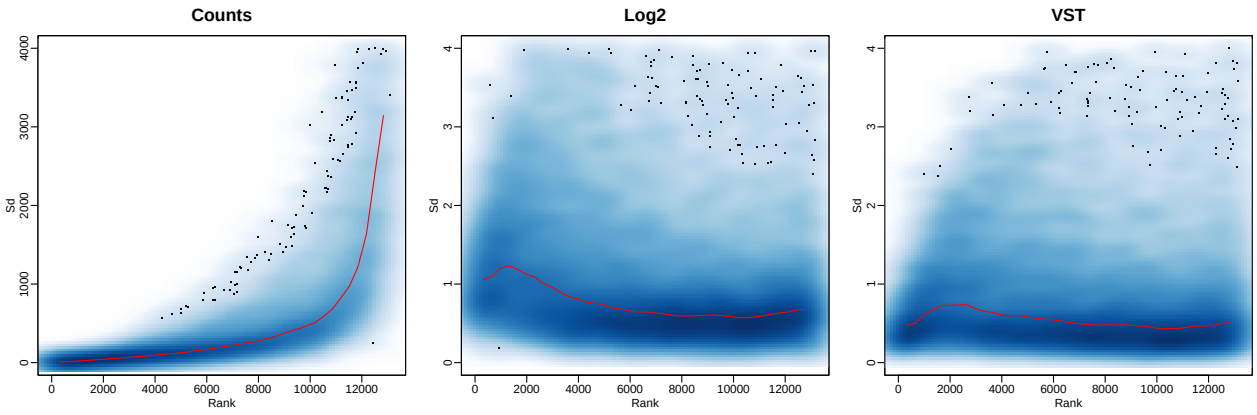
```
par(mfrow=c(1,1))
boxplot(assay(exprObjvst), xlab="", ylab="Log2 counts per million",las=2)
```



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.

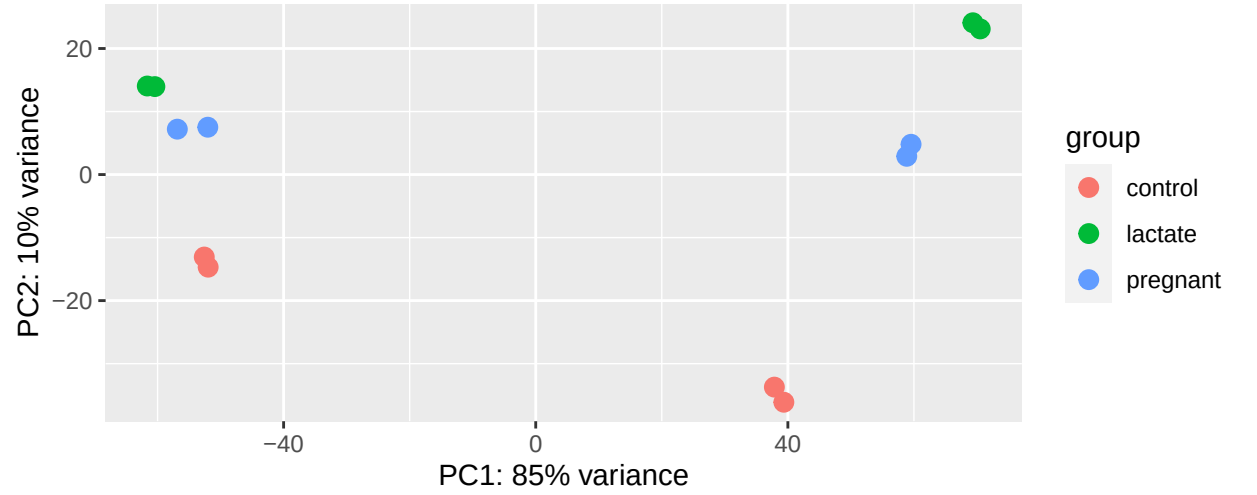
```
# BiocManager::install("vsn")
library(vsn)
rawSd <- vsn::meanSdPlot(as.matrix(assay(exprObj)), plot=FALSE)
logSd <- vsn::meanSdPlot(log2(as.matrix(assay(exprObj))+1), plot=FALSE)
vstSd <- vsn::meanSdPlot(as.matrix(assay(exprObjvst)), plot=FALSE)

par(mfrow = c(1,3),  mar = c(3, 3, 3, 1), mgp = c(1.5, 0.5, 0))
smoothScatter(x = rawSd$px, y = rawSd$py, ylab = "Sd",xlab = "Rank", main = "Counts", cex.main=1.5, yli
lines(x = rawSd$rank, y = rawSd$sd, add = TRUE, col = "red")
smoothScatter(x = logSd$px, y = logSd$py, ylab = "Sd",xlab = "Rank", main = "Log2", cex.main=1.5, ylim =
lines(x = logSd$rank, y = logSd$sd, add = TRUE, col = "red")
smoothScatter(x = vstSd$px, y = vstSd$py, ylab = "Sd",xlab = "Rank", main = "VST", cex.main=1.5, ylim =
lines(x = vstSd$rank, y = vstSd$sd, add = TRUE, col = "red")
```



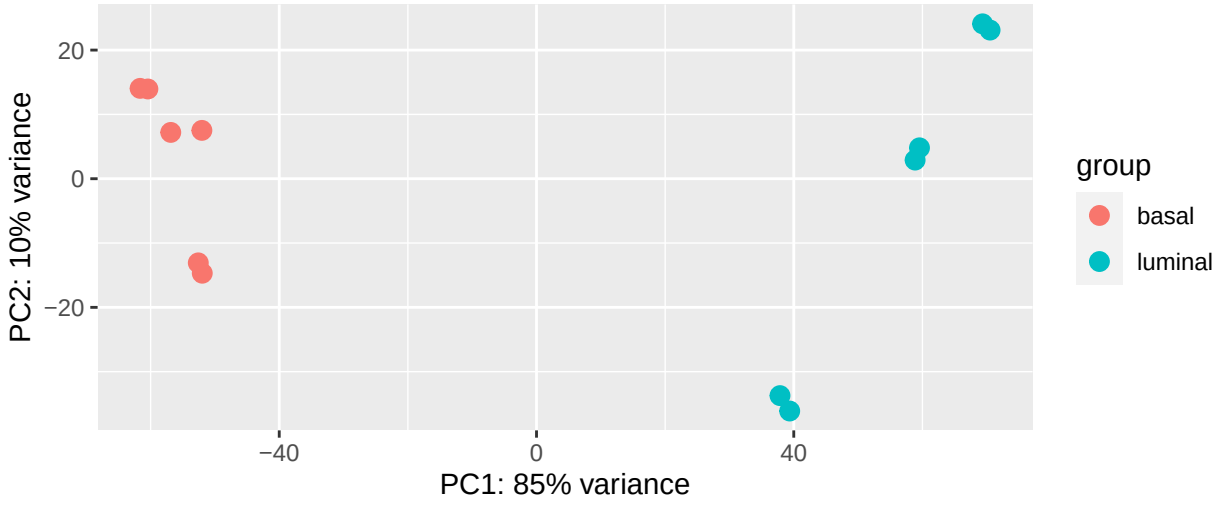
Principal Component Analysis

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"
```

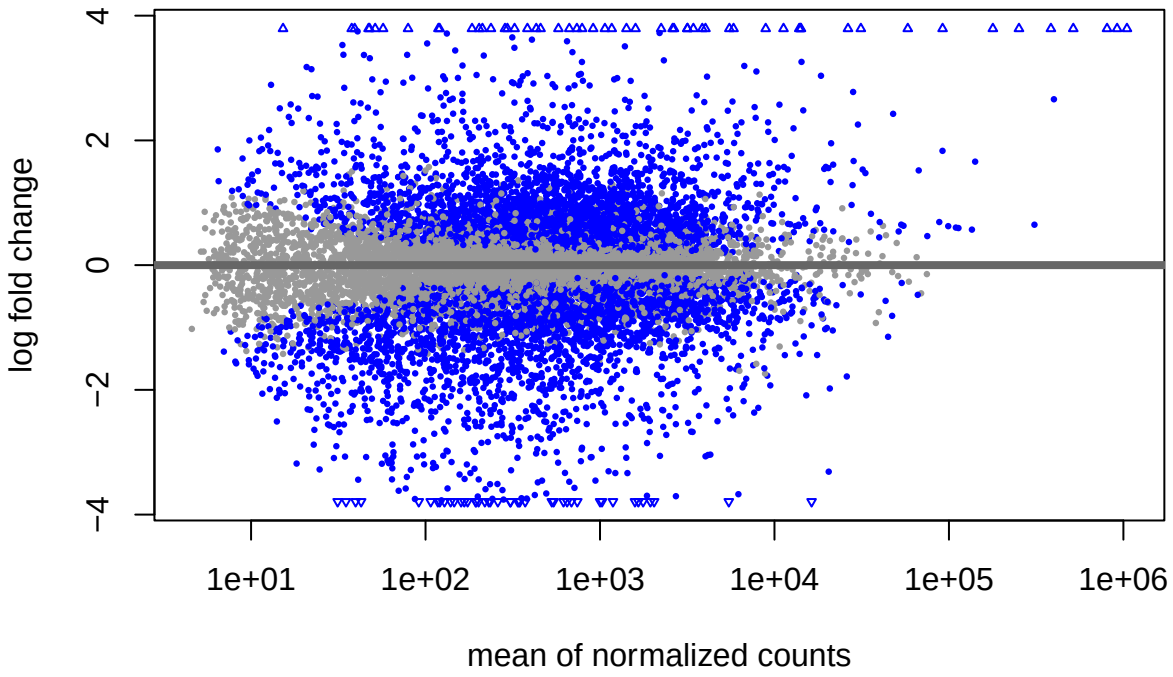
Test for DE genes between the three groups of mice, adjusted for cell type:

(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 13165 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)       : 3252, 25%
## LFC < 0 (down)     : 3142, 24%
## 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] 3406    7
```

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

```
## [1] 2310
```

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] 83
```

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
## Sox17    6.683047  7.454593  6.470362  6.897732  5.984424  6.709795  4.918217
## Mrpl15   8.028254  7.779295  8.239870  8.243236  8.299471  8.221390  8.842444
## Lypla1   9.043343  8.840751  9.665722  9.813424  9.909541  9.703443  9.381580
## Tcea1   10.307130 10.333759 10.508867 10.297672 10.223823 10.124343 10.055472
## Atp6v1h  9.272616  9.365774  9.825477  9.889705 10.007955  9.889907 10.104441
##           MCL1.LB   MCL1.LC   MCL1.LD   MCL1.LE   MCL1.LF
## Sox17    5.202961  5.158798  4.686950  4.414251  5.148489
```

```
## Mrpl15   8.503880  8.948598  8.561658  9.112769  9.249380
## Lypla1   9.358207  9.381944  9.499241  9.913801  9.993021
## Tcea1    9.888775 10.041581 10.005815  9.817283  9.781296
## Atp6v1h 10.069744 10.752296 10.755124 11.184498 11.133651
```

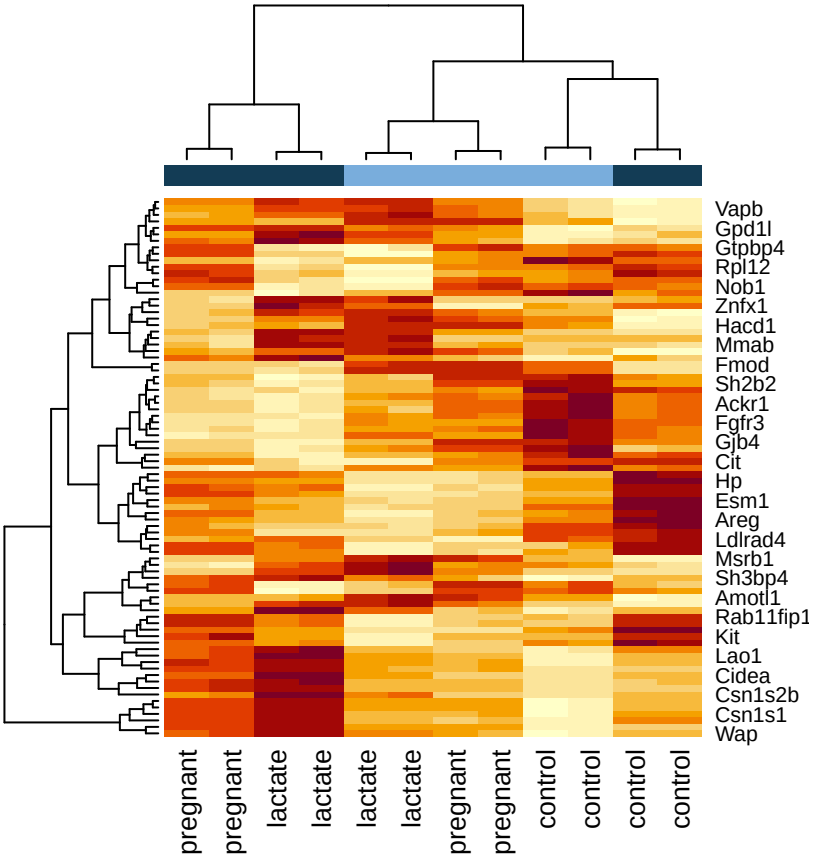
```
resVST <- assay(exprObjvst) %>%
  as.data.frame() %>%
  rownames_to_column(var = "GeneName")  %>%
  as_tibble() %>%
  filter(GeneName %in% top100)
```

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)
```


Vapb
Gpd1l
Gtpbp4
Rpl12
Nob1
Znfx1
Hacd1
Mmab
Fmod
Sh2b2
Ackr1
Fgfr3
Gjb4
Cit
Hp
Esm1
Areg
Ldlrad4
Msrb1
Sh3bp4
Amotl1
Rab11fip1
Kit
Lao1
Cidea
Csn1s2b
Csn1s1
Wap
pregnant
pregnant
lactate
lactate
lactate
lactate
pregnant
pregnant
control
control
control
control