Exercise 3. Statistical Analysis of RNA-Seq Data

The DESeq2 developers provide a well-written vignette how to use the software (https://bioconductor.org/packages/release/bioc/vignettes/DESeq2/inst/doc/DESeq2.html). Many of the examples used in this exercise are from this manual. We highly recommend that you read this vignette.

Part 1. Basic Analysis: Identify differentially expressed (DE) genes

1. Start R. There are several different versions of R installed on BioHPC computers. The default version of R (with the parallel BLAS library) does not work with DESeq2. You need to run R located in the directory "/programs/R-3.5.0s". We will start R from the directory "/workdir/$USER", so this will be your working directory within R.

```
cd /workdir/$USER
/programs/R-3.5.0s/bin/R
```

2. Load the matrix and sample files into R, and examine their contents.

In the exercise from the first week of this workshop, you created a read count matrix file named "gene_count.txt". This file contains read counts for 6 samples (wt1, wt2, wt3, mu1, mu2, mu3) . If you did not keep this file, you can use the "gene_count.txt" file located inside "/shared_data/RNAseq/exercise3/". Also, there is tab-delimited text file "samples.txt" to describe the 6 samples.

Sample	**Genotype**
wt1	wt
wt2	wt
wt3	wt
mu1	mu
mu2	mu
mu3	mu

Two new R objects will be created: "cts" (from the gene_count.txt file) and "coldata" (from the samples.txt file). As the "gene_count.txt" file does not have the sample names, the R command "colnames(cts) <- rownames(coldata)" would take the row names of "coldata" and use them as the column names for the "cts" matrix.

```
matrixFile <- "/shared_data/RNAseq/exercise3/gene_count.txt"
sampleFile <- "/shared_data/RNAseq/exercise3/samples.txt"
cts <- as.matrix(read.csv(matrixFile, sep="\t", row.names=1, header=FALSE))
coldata <- read.csv(sampleFile, sep="\t", row.names=1, header=TRUE)
head(coldata)
head(cts)
colnames(cts) <- rownames(coldata)
head(cts)
```

4. Load the DESeq2 library. Create a DESeq2 object named dds from the gene read count and sample information.

```
library(DESeq2)
dds <- DESeqDataSetFromMatrix(countData = cts,
  colData = coldata,
  design = ~ Genotype)
```

5. Create a PCA plot from the DESeq2 object (Skip this step if you have done it in Exercise 1)

```
library(ggplot2)
vsd <- vst(dds, blind=FALSE)
pcaData <- plotPCA(vsd, intgroup=c("Genotype"), returnData=TRUE)
percentVar <- round(100 * attr(pcaData, "percentVar"))
ggplot(pcaData, aes(PC1, PC2, color=Genotype)) +
geom_point(size=3) +
xlim(-2.5, 2.5) +
ylim(-1, 1) +
xlab(paste0("PC1: ",percentVar[1],"% variance")) +
ylab(paste0("PC2: ",percentVar[2],"% variance")) +
geom_text(aes(label=name),vjust=2)
ggsave("myplot.png")
```

Use FileZilla to download the "myplot.png" file to your laptop, and double click the file to view the plot.

6. Get a list of differentially expressed genes.

```
dds<-DESeq(dds)
resultsNames(dds)  # lists the coefficients

# unfiltered results
res <- results(dds, name="Genotype_wt_vs_mu")
summary(res) # print the summary on screen

# filter the results for FDR < 0.05 and absolute-value-of-Fold-Change > 2
res05 <- res[which(res$padj<0.05 & abs(res$log2FoldChange)>log2(2)), ]

# sort by padj and write to file
res05Ordered <- res05[order(res05$padj),]
write.csv(as.data.frame(res05Ordered), file="wt_vs_mu_fdr05_fc2_results.csv")
```

You can download the file "wt_vs_mu_fdr05_results.csv" with FileZilla and open it in Excel.

7. Shrinkage of log fold change of genes with very low expression.

Genes with very low expression could have imprecise fold changes. It is desirable to shrink the fold change of genes with low read counts, but not shrink the fold change of highly expressed genes. DEseq2 has implemented several different algorithms for shrinkage. The DESeq2 developers recommend to use "apeglm" for shrinkage.

Here you will compare the MA plot with or without shrinkage. The commands provided here write the plots to pdf files, and you can download and view on your laptop.

```
## no shrinkage
res <- results(dds, name="Genotype_wt_vs_mu")
pdf("res_no_shrink.pdf")
plotMA(res, main = "No shrinkage", alpha=0.05, ylim=c(-4,4))
dev.off()

## shrunk by apeglm
res_shrink <- lfcShrink(dds, coef="Genotype_wt_vs_mu", type="apeglm")
pdf("res_shrink.pdf")
plotMA(res_shrink, main = "Shrinkage by apeglm", alpha=0.05, ylim=c(-4,4))
dev.off()
```

Part 2. Interactions

1. The data files.
   In this experiment, there are 12 samples, with two variables: strains (wt vs mut), and sample collection time (0 vs 120 minutes). You are provided with two files: fission_gene_count.csv (count matrix) and fission_sample.csv (sample annotation)

	strain	**minute**
GSM1368273	wt	0
GSM1368274	wt	0
GSM1368275	wt	0
GSM1368285	wt	120
GSM1368286	wt	120
GSM1368287	wt	120
GSM1368291	mut	0
GSM1368292	mut	0
GSM1368293	mut	0
GSM1368303	mut	120
GSM1368304	mut	120
GSM1368305	mut	120

Reference:
 Leong, S. H, Dawson, K., Wirth, C., Li, Y., Connolly, Y., Smith, L. D, Wilkinson, R. C, Miller, J. C (2014). "A global non-coding RNA system modulates fission yeast protein levels in response to stress." *Nat Commun*, **5**, 3947. http://www.ncbi.nlm.nih.gov/pubmed/24853205.

2. Load data files into R. As the values of one of the variable "minute" are numeric, the minute variable needs to be converted to R factor object. The command to convert it to a factor is: `coldata$minute<-as.factor(coldata$minute)`.

```
matrixFile <- "/shared_data/RNAseq/exercise3/fission_gene_count.csv"
sampleFile <- "/shared_data/RNAseq/exercise3/fission_sample.csv"
cts <- as.matrix(read.csv(matrixFile, row.names=1))
coldata <- read.csv(sampleFile, row.names=1)
coldata$minute<-as.factor(coldata$minute)
```

3. Test of interactions of the strain and minute variable.

```
library(DESeq2)
dds <- DESeqDataSetFromMatrix(countData = cts,
     colData = coldata,
     design = ~ strain + minute + strain:minute)
ddsLRT <- DESeq(dds, test="LRT", reduced= ~ strain + minute)
resLRT <- results(ddsLRT)
summary(resLRT)
```

Part 3. Batch effects

1. The data files.

   In this experiment, there are 7 samples from two conditions (untreated and treated), and the data are from two different batches (single- and paired-end sequencing). We will correct for the batch effect in this analysis.

   Reference for this data set: Conservation of an RNA regulatory map between Drosophila and mammals" by Brooks AN, Yang L, Duff MO, Hansen KD, Park JW, Dudoit S, Brenner SE, Graveley BR, Genome Res. 2011 Feb;21(2):193-202, Epub 2010 Oct 4, PMID: 20921232

file	condition	type
treated1fb	treated	single-read
treated2fb	treated	paired-end
treated3fb	treated	paired-end
untreated1fb	untreated	single-read

file	condition	type
untreated2fb	untreated	single-read
untreated3fb	untreated	paired-end
untreated4fb	untreated	paired-end

Run the following commands to load the data set into R. There are two files: read count file "pasilla_gene_counts.tsv" and sample annotation file "pasilla_sample_annotation.csv".

```
matrixFile <- "/programs/R-
3.5.0s/library/pasilla/extdata/pasilla_gene_counts.tsv"
sampleFile <- "/programs/R-
3.5.0s/library/pasilla/extdata/pasilla_sample_annotation.csv"
cts <- as.matrix(read.csv(matrixFile,sep="\t",row.names="gene_id"))
coldata <- read.csv(sampleFile, row.names=1)
coldata <- coldata[,c("condition","type")]
```

We need to remove the two extra characters ("fb") in sample names of the sample annotation file (and convert the "-" in the type to "_" to remove some warning messages).   We also need to fix the order of samples in the read count matrix, to be consistent with sample annotation file.

```
rownames(coldata) <- sub("fb", "", rownames(coldata))
coldata$type <- sub("-", "_", coldata$type)
coldata$type <- as.factor(coldata$type)
cts <- cts[, rownames(coldata)]
```

2. DE test accounting for the batch effect

We include the batch effect variable "type" in the model by using the design formula "type+condition". Then we retrieve the results for the factor "condition", with batch effect "type" corrected.

```
library(DESeq2)
dds <- DESeqDataSetFromMatrix(countData = cts,
  colData = coldata,
  design = ~ type+condition)
dds <- DESeq(dds)

resultsNames(dds)

res <- results(dds, name="condition_untreated_vs_treated")
summary(res)
```

3. Plot PCA before and after removing batch effect.

   a. PCA plot before removing batch effect

```
library(ggplot2)
vsd <- vst(dds, blind=FALSE)
pcaData <- plotPCA(vsd, intgroup=c("condition", "type"), returnData=TRUE)
percentVar <- round(100 * attr(pcaData, "percentVar"))
ggplot(pcaData, aes(PC1, PC2, color=condition, shape = type)) +
geom_point(size=3) +
xlim(-12, 12) +
ylim(-10, 10) +
xlab(paste0("PC1: ",percentVar[1],"% variance")) +
ylab(paste0("PC2: ",percentVar[2],"% variance")) +
geom_text(aes(label=name),vjust=2)
ggsave("myPCAWithBatchEffect.png")
```

b. PCA plot after removing the batch effect with limma::removeBatchEffect

```
assay(vsd) <- limma::removeBatchEffect(assay(vsd), vsd$type)

pcaData <- plotPCA(vsd, intgroup=c("condition", "type"), returnData=TRUE)
percentVar <- round(100 * attr(pcaData, "percentVar"))
ggplot(pcaData, aes(PC1, PC2, color=condition, shape = type)) +
geom_point(size=3) +
xlim(-12, 12) +
ylim(-10, 10) +
xlab(paste0("PC1: ",percentVar[1],"% variance")) +
ylab(paste0("PC2: ",percentVar[2],"% variance")) +
geom_text(aes(label=name),vjust=2)
ggsave("myPCABatchEffectRemoved.png")
```

4. Heatmap of the samples

```
library("pheatmap")
library("RColorBrewer")
sampleDists <- dist(t(assay(vsd)))
sampleDistMatrix <- as.matrix(sampleDists)
rownames(sampleDistMatrix) <- paste(vsd$condition, vsd$type, sep="-")
colnames(sampleDistMatrix) <- NULL
colors <- colorRampPalette( rev(brewer.pal(9, "Blues")) )(255)
pdf("heatmap.pdf", height = 4, width = 5)
pheatmap(sampleDistMatrix,
         clustering_distance_rows=sampleDists,
         clustering_distance_cols=sampleDists,
         col=colors)
dev.off()
```