Exercises RNAseq

Work locally on your laptop

The following instructions have been tested on Ubuntu. Mac and Windows users can use Ubuntu in the Virtual Machine. All packages are already installed within the Virtual Machine, thus you **DON'T** need to do the following steps:

1. Copy the data to your computer:
   `wget https://www.dropbox.com/s/5vm0wfncbculs68/NGScourse.zip?dl=0`

2. Install the Python package HTSeq:
   - Make sure that the Python numpy module is installed:
     `sudo apt-get install python-numpy`
   - Install HTSeq:
     `sudo apt-get install python-htseq`

3. All RNAseq analysis in this exercises are done within R. Therefore make sure that the newest R is installed: add `deb http://stat.ethz.ch/CRAN/bin/linux/ubuntu precise/` in your /etc/apt/sources.list file:
   ```
   sudo add-apt-repository "deb
   http://stat.ethz.ch/CRAN/bin/linux/ubuntu precise/"
   sudo apt-get update
   sudo apt-get install r-base
   ```

4. Make sure that following R packages are installed: DESeq, biomaRt, goseq, GenomicFeatures, GO.db. These are all bioconductor packages.
   First the XML and curl package need to be installed:
   ```
   sudo apt-get install libxml2-dev
   sudo apt-get install libcurl4-gnutls-dev
   ```

   Now open R as administrator (`sudo R`) and download packages:
   ```
   source("http://bioconductor.org/biocLite.R")
   biocLite("DESeq2")
   biocLite("biomaRt")
   biocLite("goseq")
   biocLite("GenomicFeatures")
   biocLite("GO.db")
   biocLite("org.Mm.eg.db")
   ```

Exercise 1: Obtain read counts from HTSeq

If you have an alignment file (SAM or BAM) and an annotation file (GFF or GTF), you can count how many reads map to each feature by using htseq-count script.

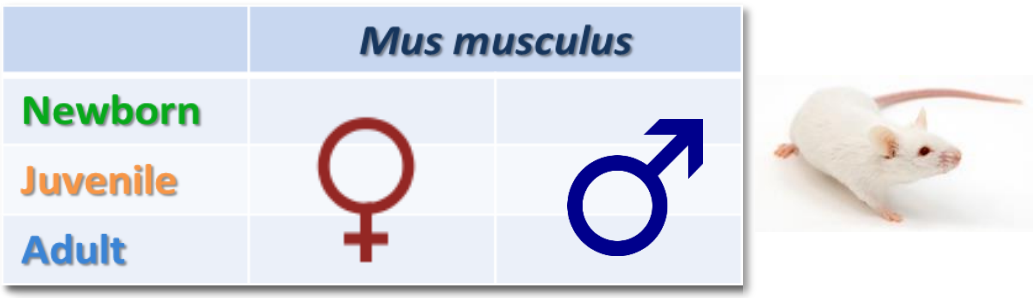
1. Install the Python package HTSeq:
    a. Make sure that the Python numpy module is installed:
    ```
    sudo apt-get install python-numpy
    ```
    b. Install HTSeq:
    ```
    sudo apt-get install python-htseq
    ```

2. Use the yeast_RNASeq_excerpt.sam and Saccharomyces_cerevisiae.SGD1.01.56.gtf file to obtain read counts per gene (-s no: no strand-specific assay):
    ```
    htseq-count –s no yeast_RNASeq_excerpt.sam
    Saccharomyces_cerevisiae.SGD1.01.56.gtf > count.txt
    ```

Exercise 2: Differential gene expression analysis

We will work on an RNAseq data set from a laboratory strain of mouse (Mus musculus). The normalized cDNA of three different life stages (newborn, juvenile and adult) and both sexes were obtained by paired-end Illumina sequencing. Three individuals and several tissues (kidney, liver, heart, muscle and brain) were pooled for each "life stage-sex- category". The resulting reads were mapped against the reference cDNA sequences of mouse (ENSEMBL) and read counts were obtained per gene.

The provided read-count table contains the raw read counts of the 6 different "life stage-sex-categories" (N: newborn, J: juvenile, A: adult, f: female, m: male) for 31,656 genes (83 % of all ENSEMBLE annotated genes).



In this first exercise we aim to find the genes showing differential expression between newborns and adults. For both life-stages we sequenced males and females, which we will use as biological replicates.

1. Start R and read in read-count table with genes in rows and counts for each sample in columns (first row contains column names, gene names are given in the first column and used as row names):
```
countTable <- read.delim("RNAseq_Mus.txt", header=TRUE,
row.names=1)
```

   Have a look at the first few rows of the table:
```
head(countTable)
```

2. Create a dataframe with the conditions of the samples. Each row should correspond to a sample with sample names as row names:
```
group <- data.frame(condition=c("N","N","J","J","A","A"),sex=
c("f","m","f","m","f","m"),row.names=names(countTable))
group
```

3. Load the DESeq2 library and set up the count data set, with the two conditions as factor levels:
```
library(DESeq2)
ddsMF <- DESeqDataSetFromMatrix(countData=countTable,
colData=group, design=~sex+condition)
```

   If you obtained the count tables from HTSeq (like in exercise 1), DESeq2 provides a function to load these data (you will have one count table per sample):
   a. Get list of count table files:
```
sampleFiles <- list.files(path="/path/toFolder/HTseq_Res",
pattern="*.txt")
```
   b. Create vector of conditions (let's assume that we have 3 "healthy" samples and 3 "disease" samples):
```
conditions <- factor(c(rep("healthy",3), rep("disease",3)))
```
   c. Load data:
```
sampleTable <- data.frame(sampleName=sampleFiles,
fileName=sampleFiles, condition=conditions)
directory <- "/path/toFolder/HTseq_Res"
ddsHTSeq <-
DESeqDataSetFromHTSeqCount(sampleTable=sampleTable,
directory=directory, design=~condition)
```

4. The standard differential expression analysis steps are wrapped into a single function: DESeq. To get differentially expressed genes between newborns and adults you just need to execute following commands:

```
ddsMF <- DESeq(ddsMF)
res <- results(ddsMF , contrast=c("condition","N","A"))
head(res)
```

To get a description of the columns in res, execute following command:

```
mcols(res)$description
```

The DESeq function performs following steps, which can also be run manually:

a. Normalize the count data set by estimating the size factor:

```
ddsMF <- estimateSizeFactors(ddsMF)
```

If each column of the count table is divided by the size factor for this column, the count values are brought to a common scale. To view the size Factors you can execute following command:

```
sizeFactors(ddsMF)
```

b. Estimate variance (dispersion) of gene expression between biological replicates. If the gene expression differs between replicates by 20%, then the gene's dispersion is 0.2^2=0.04.

```
ddsMF <- estimateDispersions(ddsMF)
```

To plot the extimated variance (red line= fitted dispersion):

```
plotDispEsts(ddsMF)
```

c. Get differentially expressed genes:

```
ddsMF <- nbinomWaldTest(ddsMF)
```

5. Get a summary of the differential expressed genes at a FDR < 5%:

```
summary(res, alpha=0.05)
```

6. Plot results with all genes differentially expressed (red: FDR < 5% in red):

```
plotMA(res, alpha=0.05)
```

histogram of p-value/FDR distributions:

```
hist(res$pval, breaks=100, col="skyblue", main="")
hist(res$padj, breaks=100, col="skyblue", main="")
```

7. Order results according strength of differential expression:
```
resOrdered <- res[order(res$padj), ]
```

8. Get list of over-expressed genes in newborns/adulsts at FDR 5%:
```
overN <- resOrdered[resOrdered$padj<0.05 &
!is.na(resOrdered$padj) & resOrdered$log2FoldChange>0, ]
nrow(overN)
overA <- resOrdered[resOrdered$padj<0.05 &
!is.na(resOrdered$padj) & resOrdered$log2FoldChange<0, ]
nrow(overA)
```

9. Write differential expression table in a text file:
```
write.table(resOrdered, "diffExp_N-A.txt", sep="\t",
row.names=FALSE)
```

 How many genes are differentially expressed at a FDR of 1%?

Exercise 2: Get additional information for differentially expressed genes

In this exercise we will use the biomaRt library to get additional information about the genes overexpressed in newborns.

1. Load biomaRt libray and setup database to use
```
library(biomaRt)
mart <- useDataset("mmusculus_gene_ensembl", useMart("ensembl"))
```

2. Get vector of gene names overexpressed in newborns:
```
genesOverN <- rownames(overN)
```

3. Get additional gene information from biomart: gene name, chromosomal location, gene description
```
ensembl_translation <- getBM(filters= "ensembl_gene_id",
attributes= c("ensembl_gene_id", "external_gene_name",
"chromosome_name", "description"), values=genesOverN, mart=mart)
head(ensemble_translation)
```

Exercise3: search for enriched GO terms:

In this exercise we will use goSeq to test if overexpressed genes of newborns are enriched for any GO terms.

1. Prepare goSeq database (get genes and their length). This will take some time..:
```
library(GenomicFeatures)
txdb <- makeTxDbFromBiomart(dataset="mmusculus_gene_ensembl")
txsByGene <- transcriptsBy(txdb,"gene")
lengthData <- median(width(txsByGene))
```

2. Load goSeq library and convert differential expressed genes to a goSeq vector (genes overexpressed in newborn =1, genes not overexpressed in newborns = 0)
```
library(goseq)
genesN <- as.integer(resOrdered$padj<=0.05 &
!is.na(resOrdered$padj) & resOrdered$log2FoldChange>0)
names(genesN)<-rownames(resOrdered)
head(genesN)
```

3. Quantify the length bias present in the dataset with a probability weighting function (function which gives the probability that a gene will be differentially expressed based on its length alone)
```
lengthDataN  <- lengthData[names(genesN)]
pwfN <- nullp(genesN, bias.data=lengthDataN)
```

4. Get enriched GO terms. Here we limit to the GO term category Biological processes (to get all categories remove the argument "test.cats"):
```
GO_N_BP <- goseq(pwfN,"mm10","ensGene", test.cats=c("GO:BP"))
```

5. Correct for multiple testing by the Benjamini and Hochberg correction (FDR):
```
GO_N_BP$FDR_over_represented <-
p.adjust(GO_N_BP$over_represented_pvalue, method="BH")
```

6. Filter for GO terms enriched at a FDR of 5% and get a description for those GO terms using the GO.db library
```
enriched_GO_N_BP <- GO_N_BP[GO_N_BP$FDR_over_represented <=
0.05,]
nrow(enriched_GO_N_BP)
head(enriched_GO_N_BP)

library(GO.db)
enriched_GO_N_BP$Description <- NA
index <- 1
for(go in enriched_GO_N_BP$category){
```

```
        enriched_GO_N_BP$Description[index]  <- Term(GOTERM[[go]])
        index <- index + 1
    }
    head(enriched_GO_N_BP)
```

What GO terms are enriched in genes overexpressed in newborns?

And what GO terms are enriched in genes overexpressed in adults?

Extra Exercise: differential expression between females and males

What genes are overall differentially expressed between females and males (hint: samples from different life stages can be used as biological replicates)?

Are there also some GO terms enriched in genes overexpressed in femals/males?