RNAseq Analysis

Vidal Adrien

October 23, 2017

1) Download 4 sequence datasets deposited to the EBI ENA:
I create a script (**./scripts/analysis.sh**) for the retrieval and treatment of files and sequence alignment procedures.

```
  mkdir sources
  wget -P ./sources ftp://ftp.sra.ebi.ac.uk/vol1/fastq/ERR990/ERR990557/ERR990557.fastq.gz
  wget -P ./sources ftp://ftp.sra.ebi.ac.uk/vol1/fastq/ERR990/ERR990558/ERR990558.fastq.gz
  wget -P ./sources ftp://ftp.sra.ebi.ac.uk/vol1/fastq/ERR990/ERR990559/ERR990559.fastq.gz
  wget -P ./sources ftp://ftp.sra.ebi.ac.uk/vol1/fastq/ERR990/ERR990560/ERR990560.fastq.gz
```

Some quality control of the library can be performed using fastqc:

```
mkdir fastqc_samp_57 & fastqc -o fastqc_samp_57 ./sources/ERR990557.fastq.gz
mkdir fastqc_samp_58 & fastqc -o fastqc_samp_58 ./sources/ERR990558.fastq.gz
mkdir fastqc_samp_59 & fastqc -o fastqc_samp_59 ./sources/ERR990559.fastq.gz
mkdir fastqc_samp_60 & fastqc -o fastqc_samp_60 ./sources/ERR990560.fastq.gz
```

The reports generated for this analysis show overall high quality scores for all downloaded libraries. For further operations, I also download a set of reference genome and genes annotation files for *Drosophilia melanogaster* from flybase:

```
#Reference sequence:
wget -P ./sources ftp://ftp.flybase.net/genomes/Drosophila_melanogaster/dmel_r6.01_FB2017_04/fasta/dmel-all-
gunzip -c ./sources/dmel-all-chromosome-r6.01.fasta.gz > ./sources/dmel-all-chromosome-r6.01.fasta
rm ./sources/dmel-all-chromosome-r6.01.fasta.gz
ref="./sources/dmel-all-chromosome-r6.01.fasta"

#Genome annotations:
wget -P ./sources ftp://ftp.flybase.net/genomes/Drosophila_melanogaster/dmel_r6.01_FB2017_04/gff/dmel-all-r6
gunzip -c ./sources/dmel-all-r6.01.gff.gz > ./sources/dmel-all-r6.01.gff
rm dmel-all-r6.01.gff.gz
```

2) Extract fastq files:
The **fastq** files are extracted using gunzip:

```
gunzip -c ./sources/ERR990557.fastq.gz > ./sources/ERR990557.fastq
gunzip -c ./sources/ERR990558.fastq.gz > ./sources/ERR990558.fastq
gunzip -c ./sources/ERR990559.fastq.gz > ./sources/ERR990559.fastq
gunzip -c ./sources/ERR990560.fastq.gz > ./sources/ERR990560.fastq
```

3) For each file, select 8,000,000 (8 millions) of sequence reads and generate the following sample files:
I select 8 000 000 sequences from each library randomly using the **seqtk** utility. The unzipped original **fastq** files are no longer needed after this point:

```
mkdir fastq
samp_size=8000000
seqtk sample -s100 ./sources/ERR990557.fastq $samp_size > ./fastq/ERR990557s.fastq
seqtk sample -s100 ./sources/ERR990558.fastq $samp_size > ./fastq/ERR990558s.fastq
seqtk sample -s100 ./sources/ERR990559.fastq $samp_size > ./fastq/ERR990559s.fastq
seqtk sample -s100 ./sources/ERR990560.fastq $samp_size > ./fastq/ERR990560s.fastq
rm ./sources/*.fastq #remove unzipped fastq files (large files).
```

4) Align these read datasets to the reference genome by any appropriate mean, and generate a sorted bam alignment file:
Alignment to the sample libraries are performed using **bwa** and then filtered for uniquely mapped reads using a **grep** command:

```
mkdir bwa

#~ Index the fasta file
bwa index $ref

#~ Run bwa:
bwa aln -t 4 $ref ./fastq/ERR990557s.fastq > bwa/ERR990557s.sai
bwa samse $ref bwa/ERR990557s.sai ./fastq/ERR990557s.fastq > bwa/ERR990557s.sam
#~  For ERR990557s

#~ Run bwa:
bwa aln -t 4 $ref ./fastq/ERR990558s.fastq > bwa/ERR990558s.sai
bwa samse $ref bwa/ERR990558s.sai ./fastq/ERR990558s.fastq > bwa/ERR990558s.sam
#~  For ERR990558s

#~ Run bwa:
bwa aln -t 4 $ref ./fastq/ERR990559s.fastq > bwa/ERR990559s.sai
bwa samse $ref bwa/ERR990559s.sai ./fastq/ERR990559s.fastq > bwa/ERR990559s.sam
#~  For ERR990559s

#~ Run bwa:
bwa aln -t 4 $ref ./fastq/ERR990560s.fastq > bwa/ERR990560s.sai
bwa samse $ref bwa/ERR990560s.sai ./fastq/ERR990560s.fastq > bwa/ERR990560s.sam
#~  For ERR990560s

#~  remove sai files:
rm bwa/*.sai

cd ./bwa
#~ Filter uniquely mapped reads:
grep XT:A:U ERR990557s.sam > ERR990557s_filt.sam
grep XT:A:U ERR990558s.sam > ERR990558s_filt.sam
grep XT:A:U ERR990559s.sam > ERR990559s_filt.sam
grep XT:A:U ERR990560s.sam > ERR990560s_filt.sam
#~ The tag XT:A:U means that the read maps to a unique position in the genome.
cd ..
```

Additional filters (PCR duplicates, low quality reads, etc.) and a **bam** conversion are applied before sorting the reads using **samtools** utilites:

```
#~ Additional filters + sam to sorted bam conversion:
samtools faidx $ref
samtools view -bS -F 1548 -q 30 -t $ref.fai ./bwa/ERR990557s_filt.sam \
| samtools sort - ./bwa/ERR990557s_filt_sorted #.bam
samtools view -bS -F 1548 -q 30 -t $ref.fai ./bwa/ERR990558s_filt.sam \
 | samtools sort - ./bwa/ERR990558s_filt_sorted #.bam
samtools view -bS -F 1548 -q 30 -t $ref.fai ./bwa/ERR990559s_filt.sam \
 | samtools sort - ./bwa/ERR990559s_filt_sorted #.bam
samtools view -bS -F 1548 -q 30 -t $ref.fai ./bwa/ERR990560s_filt.sam \
 | samtools sort - ./bwa/ERR990560s_filt_sorted #.bam
```

5) Count reads aligning to genome's genes by any appropriate mean:
After indexing the sorted **bam** files, the read's coverage of the genes is measured against the annotation file:

```
samtools index ./bwa/ERR990557s_filt_sorted.bam
samtools index ./bwa/ERR990558s_filt_sorted.bam
samtools index ./bwa/ERR990559s_filt_sorted.bam
samtools index ./bwa/ERR990560s_filt_sorted.bam

bedtools multicov -s -bams ./bwa/ERR990557s_filt_sorted.bam \
                           ./bwa/ERR990558s_filt_sorted.bam \
                           ./bwa/ERR990559s_filt_sorted.bam \
                           ./bwa/ERR990560s_filt_sorted.bam \
                           -bed ./sources/dmel-all-r6.01.bed > coverage.txt
```

At this point, this tool has been running continuously for 20 hours with no indication of its overall progress. I have also not been able to find the correspondance between the library IDs and the experiment design presented in the publication. Instead of continuing the analysis, I will atempt to describe what my next steps would have been were I able to proceed.

7) Perform a statistical differential expression analysis and report using any appropriate figure(s)/graph(s):
I would write an **R** script to parse the output **coverage.txt** table and perform the differential expression analysis. The parsed file would result in a table with the 4 libraries as columns and the genes' identifiers as rows, each cell containing the number of reads of a library aligning to a gene. Assuming an experimental design with two control replicates and two Rpp30 mutant replicates, the column headers would be changed to reflect said design.

The **HTSFilter** library could be used first to filter out genes with zero alignment hits. The data could then be normalized using the cpm formula (# of reads / library size). This can be done with the **edgeR** library by creating a **DGEList** object from our counts table and calculating its normalization factors.

A descriptive analysis of the data may be done by performing a PCA (using **MixOmics** for instance) on the cpm values. The axes of the PCA should reflect a good proportion of the statistical information and the main axis should allow us to separate the mutant libraries from the control libraries. **5) Select a list**

of genes likely to be differentially expressed with a p-adj value ¡ 0.01:

A simple model matrix can then be created for the experimental design from which we can estimate the dispersions (common, trended and pairwise biological coefficients of variation). These dispersions can be visualized using **plotBCV**. The dispersions may then be fitted to the model matrix and a contrast object (**makeContrasts**) created (*Ctl.vs.Mut = Mut-Ctl*) to start performing the differential analysis. This analysis is performed using the **glmLRT** function and outputs the *log(Fold Change)* as well as the p-values. These *p-values* can be adjusted either by the Bonferroni or the Benjamini-Hochberg method (Adjusted *p-values* added to the output table). A group of genes likely differentially expressed can then be selected by filtering the table using the prefered adjusted *p-value* column.

8) Code a simple script that parse the table of differential expressions (from 6.) and return the genes with a p-adj value ¡ 0.01 for rejection of H0 (non differential expression)
I don't see the need to use differential expressions with the type of tabulated output given by edgeR. A script to filter a table file outputed by edgeR (and then written to a file) would look like this:

```
#!/usr/bin/R
#Script name: filter-DE-Genes.R
#Call:  Rscript --vanilla filter-DE-Genes.R filename p-value p-col out
#                                          arg 1    arg 2   arg 3 arg 4
#
#Example: Rscript --vanilla filter-DE-Genes.R Ctl-vsMut.tab 0.1 BH DE-genes.txt
#
#Arguments:
#    (1) filename: Name of a table file such as:
#        - The first column  corresponds to the gene's ids.
#        - At least one column contains (adjusted) p-values for the differential expression of these genes.
#    (2) p-value:  (adjusted) p-value threshold for filtering.
#    (3) p-col:    Name of the column containing (adjusted) p-values.
#    (4) out:      Name of the file to output the filtered gene ids list.

args <- commandArgs(trailingOnly=TRUE)
```

```
table <- read.table(args[1], sep="\t", header=T)

write.lines(table[1][ table[as.character(args[3])] < as.numeric(args[2]) ], file(as.character(args[4])) )
```