1. Downloading and extraction of Fastq files

Files were downloaded from https://www.ebi.ac.uk/ena using "Run numbers" (ERR990557, ERR990558, ERR990559 and ERR990560) and selecting "FASTQ files (FTP)" option. Then, files were unzipped to start the pipeline.

2. Selection of sequence reads

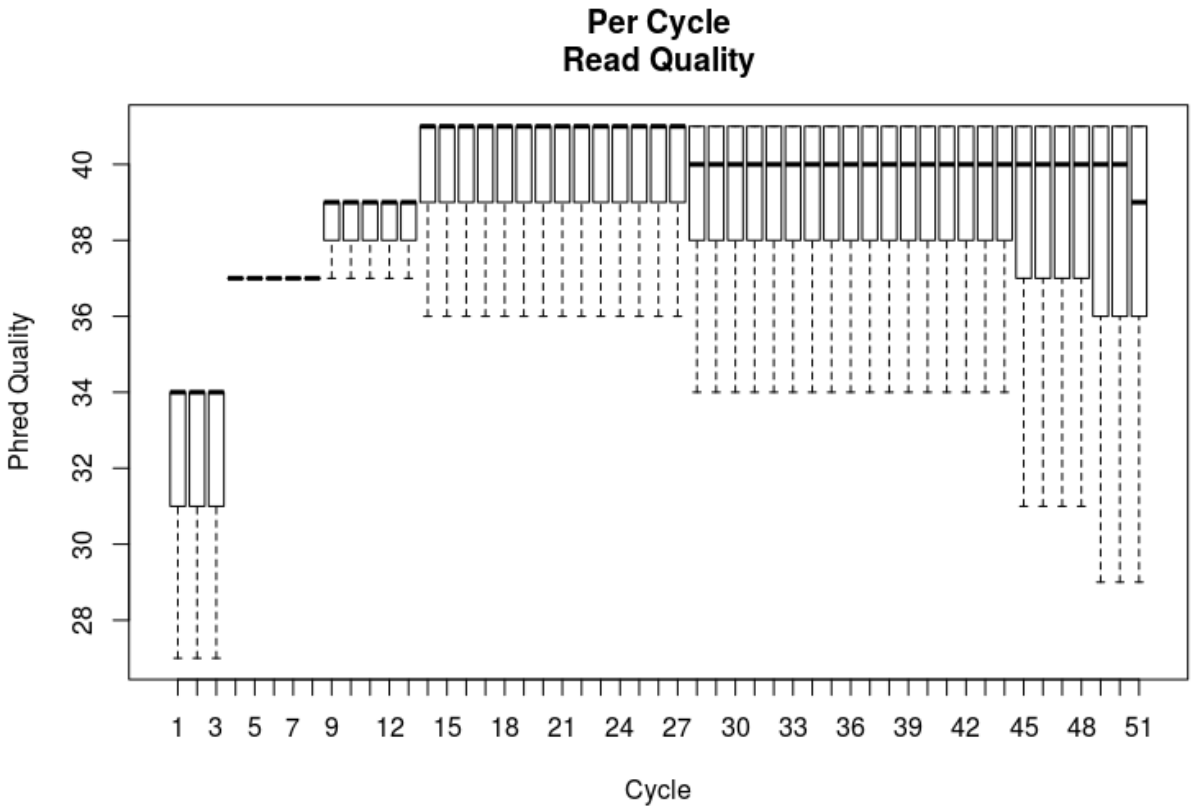
Eight million of sequence reads were selected randomly using the following command in the terminal:

```
cat ERR990557.fastq | awk '{ printf("%s",$0); n++; if(n%4==0) { printf("\n");} else { printf("X#&X");} }' | shuf | head -8000000 | sed 's/X#&X/\n/g' > ERR990557s.fastq

cat ERR990558.fastq | awk '{ printf("%s",$0); n++; if(n%4==0) { printf("\n");} else { printf("X#&X");} }' | shuf | head -8000000 | sed 's/X#&X/\n/g' > ERR990558s.fastq

cat ERR990559.fastq | awk '{ printf("%s",$0); n++; if(n%4==0) { printf("\n");} else { printf("X#&X");} }' | shuf | head -8000000 | sed 's/X#&X/\n/g' > ERR990559s.fastq

cat ERR990560.fastq | awk '{ printf("%s",$0); n++; if(n%4==0) { printf("\n");} else { printf("X#&X");} }' | shuf | head -8000000 | sed 's/X#&X/\n/g' > ERR990560s.fastq
```

In addition, all new files were charged in *Rstudio* to verify the quality of reads. The commands used were:

```
library(ShortRead)

myFiles <- list.files(getwd(), "fastq", full=TRUE)

myFQ <- lapply(myFiles, readFastq)

myQual <- FastqQuality(quality(quality(myFQ[[1]])))

readM <- as(myQual, "matrix")

boxplot(readM, outline = FALSE, main="Per Cycle Read Quality", xlab="Cycle", ylab="Phred Quality")
```

Those commands generated a plot that indicate that reads had a good quality, the same results were obtained using FASTQC tool (file:///E:/Test/alignments.together.sorted_fastqc.html).



3. Aligning dataset to the *Drosophila melanogaster* genome and generation of a "sorted bam file"

To align fastq files to the reference genome (*Drosophila melanogaster*), the first fasta of the site ftp://ftp.ensemblgenomes.org/pub/metazoa/release-37/fasta/drosophila_melanogaster/dna_index/ was downloaded. To facilitate commands in the terminal, the name of this file was changed to *D_m.fa*.

The following commands were executed in order to obtain

```
bwa index -a bwtsw D_m.fa

# Mapping
bwa aln D_m.fa ERR990557s.fastq > out57.sai
bwa aln D_m.fa ERR990558s.fastq > out58.sai
bwa aln D_m.fa ERR990559s.fastq > out59.sai
bwa aln D_m.fa ERR990560s.fastq > out60.sai

# SAM files (samse for sigle reads et sampe for paired reads)

bwa samse D_m.fa out57.sai ERR990557s.fastq > align57.sam
bwa samse D_m.fa out58.sai ERR990558s.fastq > align58.sam
bwa samse D_m.fa out59.sai ERR990559s.fastq > align59.sam
bwa samse D_m.fa out60.sai ERR990560s.fastq > align60.sam
```

```
# SAM to BAM (1870 sequences) (remove not mapping sequences)
samtools view -F 4 -Sbh align57.sam > BAMalign57.bam
samtools view -F 4 -Sbh align58.sam > BAMalign58.bam
samtools view -F 4 -Sbh align59.sam > BAMalign59.bam
samtools view -F 4 -Sbh align60.sam > BAMalign60.bam

# Generate un "BAMalign57.sorted.bam"
samtools sort BAMalign57.bam BAMalign57.sorted
samtools sort BAMalign58.bam BAMalign58.sorted
samtools sort BAMalign59.bam BAMalign59.sorted
samtools sort BAMalign60.bam BAMalign60.sorted

# Generate a ".bai" index used to localise reads easily (IGV)
samtools index BAMalign57.sorted.bam
samtools index BAMalign58.sorted.bam
samtools index BAMalign59.sorted.bam
samtools index BAMalign60.sorted.bam

# Check the number of reads.
samtools idxstats BAMalign57.sorted.bam
```

4. Differential expression treatment starting from "sorted bam files"

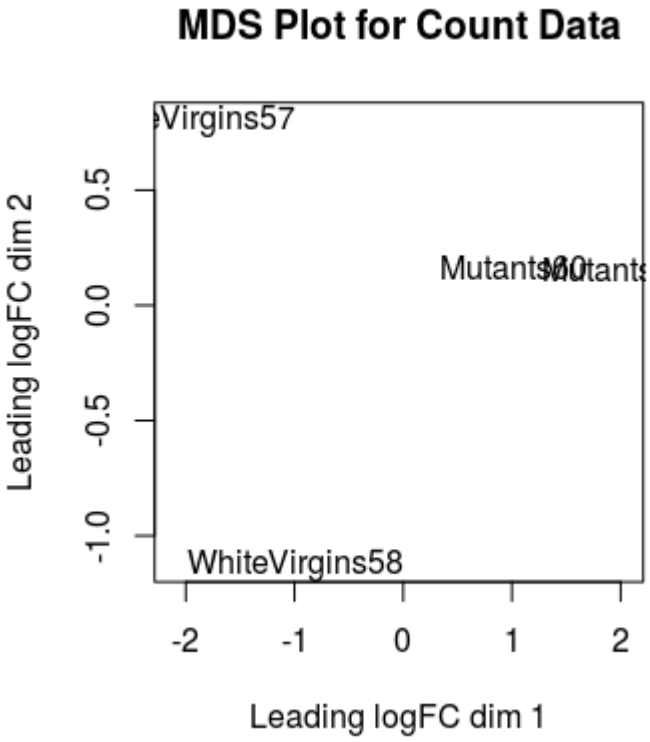
This part of the analysis was done using *Rstudio* and included the creation of table to count reads, the annotation of genes, the differential expression analysis and plotting some information. Here is the script:

```
source("https://bioconductor.org/biocLite.R")

biocLite("GenomicRanges")
biocLite("GenomicFeatures")
biocLite("Rsamtools")
biocLite("DESeq")
biocLite("edgeR")
biocLite("org.Dm.eg.db")

# load library for genomic annotations
library(GenomicFeatures)
library(GenomicRanges)

# load the transcript annotation file from UCSC.  Make sure to enter the
correct genome version
txdb <- makeTxDbFromBiomart(host="ensembl.org",
                            biomart ="ENSEMBL_MART_ENSEMBL",
                            dataset = "dmelanogaster_gene_ensembl")

ex_by_gene=transcriptsBy(txdb,'gene')
```

```
# load the samtools library for R
library(Rsamtools)

# read the sequencing read alignment into R (combine with next step to save
memory)

biocLite("GenomicAlignments")    #    necessaire    pour    la    fonction
"readGAlignments"
library(GenomicAlignments)
reads1r57=readGAlignments("BAMalign57.sorted.bam")
reads1r58=readGAlignments("BAMalign58.sorted.bam")
reads2r59=readGAlignments("BAMalign59.sorted.bam")
reads2r60=readGAlignments("BAMalign60.sorted.bam")
#repeat as necessary for more samples)

# count reads overlapping the exons
counts1r57 = countOverlaps(ex_by_gene,reads1r57)
counts1r58 = countOverlaps(ex_by_gene,reads1r58)
counts2r59 = countOverlaps(ex_by_gene,reads2r59)
counts2r60 = countOverlaps(ex_by_gene,reads2r60)

# create count table
countTable                                                                =
data.frame(WhiteVirgins57=counts1r57,WhiteVirgins58=counts1r58,Mutants59=c
ounts2r59,
                        Mutants60=counts2r60,stringsAsFactors=FALSE)

# set the gene IDs to the table row names
rownames(countTable)=names(ex_by_gene)

#removing rows that are zero for all genes (edgeR and DESeq have trouble
with these)
x <- rowSums(countTable==0)!=ncol(countTable)
newCountTable <- countTable[x,]

# # Adding Annotation
# # Lets say you have a table named "dataTable" (must be data table, i.e.
dataTable <- as.data.table(x)).

library(org.Dm.eg.db)

# # Use this command to see which types of IDs you can convert:
keytypes(org.Dm.eg.db)
# # [1] "ACCNUM"       "ALIAS"        "ENSEMBL"      "ENSEMBLPROT"
# # [5] "ENSEMBLTRANS" "ENTREZID"     "ENZYME"       "EVIDENCE"
# # [9] "EVIDENCEALL"  "FLYBASE"      "FLYBASECG"    "FLYBASEPROT"
# # [13] "GENENAME"     "GO"           "GOALL"        "MAP"
# # [17] "ONTOLOGY"     "ONTOLOGYALL"  "PATH"         "PMID"
# # [21] "REFSEQ"       "SYMBOL"       "UNIGENE"      "UNIPROT"
#
```

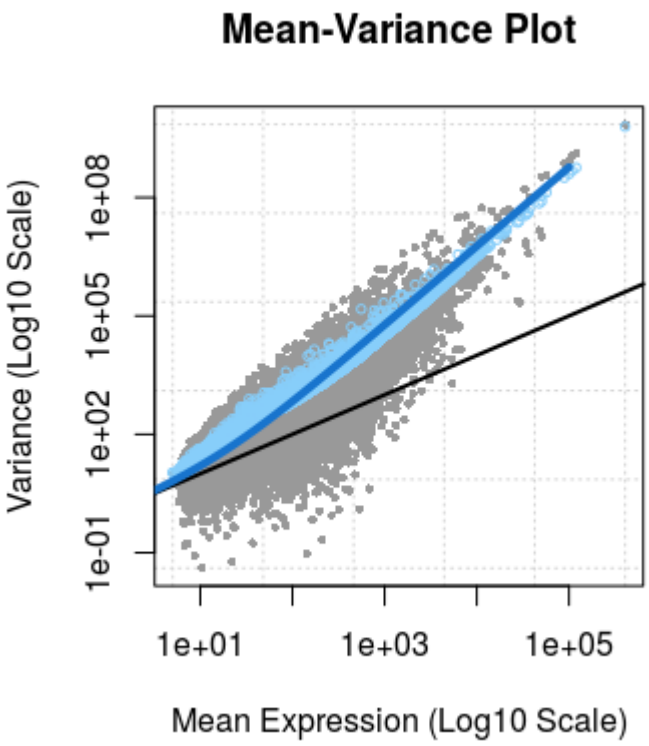
```
install.packages("data.table")
library(data.table)

dataTable = copy(newCountTable)
fbids = rownames(newCountTable)
annots <- select(org.Dm.eg.db, keys = fbids, columns = "SYMBOL", keytype =
"ENSEMBL")
dataTable$ENSEMBL = rownames(dataTable)
newTable = merge(annots, dataTable, by.x = "ENSEMBL", by.y = "ENSEMBL")

# Il y a des codes ENSEMBL qui on plusieurs symbols, donc la commande
suivante est pour garde qu'un
newTable = newTable[!duplicated(newTable[,"ENSEMBL"]),]

# To convert a column in rownames
rownames(newTable) = newTable$ENSEMBL

# To supprime the 1st column
newTable = newTable[-1]

### DE analysis
########
# edgR #
########


biocLite("edgeR")
biocLite("goseq")
library(edgeR)
library(goseq)

# Building edgeR Object
myTreat <- factor(rep(c("WhiteVirgin","Mutant"),times = c(2,2)))

cds = DGEList(newCountTable, group = myTreat)
names(cds)
# [1] "counts"  "samples"
head(cds$counts)
#             WhiteVirgins57 WhiteVirgins58 Mutants59 Mutants60
# FBgn0000003          30423          29615     31789     45589
# FBgn0000008            470            429       300       443
# FBgn0000014            245            176       749       409
# FBgn0000015             32             25       101        42
# FBgn0000017            501            447       468       531
# FBgn0000018            111             84        49        73

# The method used in the edgeR vignette is to keep only those genes that
have at least 1 read per million in at least 3 samples
```

```
cds <- cds[rowSums(1e+06 * cds$counts/expandAsMatrix(cds$samples$lib.size,
dim(cds)) > 1) >= 3, ]
dim( cds )
# [1] 10064     4

cds <- calcNormFactors( cds )
cds$samples
#                       group lib.size norm.factors
# WhiteVirgins57 WhiteVirgin  5457105    1.0668849
# WhiteVirgins58 WhiteVirgin  6256661    0.9923420
# Mutants59           Mutant  5683632    0.9166546
# Mutants60           Mutant  6509157    1.0304225

# effective library sizes
cds$samples$lib.size * cds$samples$norm.factors
# [1] 5822103 6208748 5209927 6707182

# Plot similarity between samples
plotMDS( cds , main = "MDS Plot for Count Data", labels = colnames( cds$counts
) )
plotMDS(myDG)
```




```
# Estimating dispersions
cds <- estimateCommonDisp( cds )
names( cds )
```

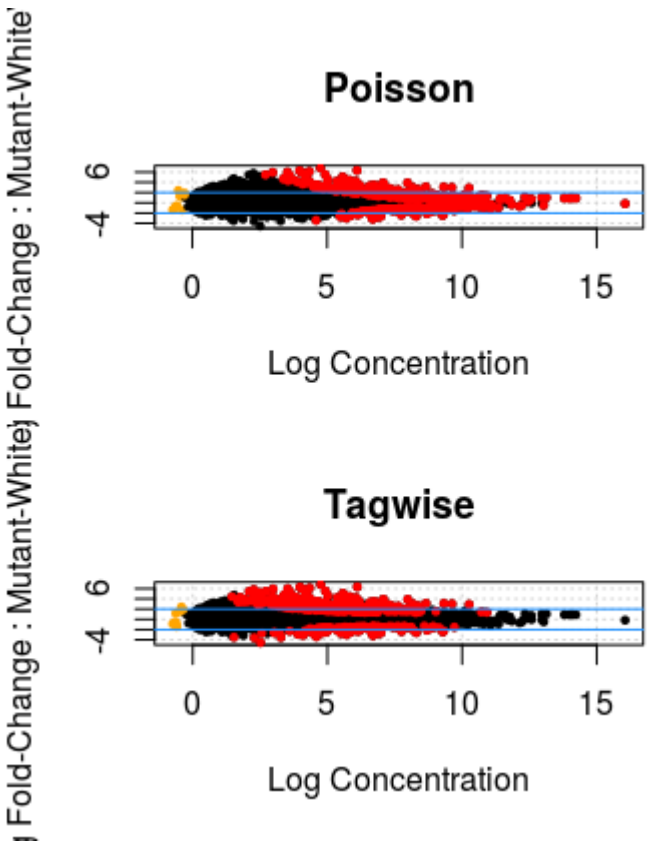
```
# [1] "counts"            "samples"           "common.dispersion"
# [4] "pseudo.counts"     "pseudo.lib.size"   "AveLogCPM"

# The estimate
cds$common.dispersion
# [1] 0.05840181

# More shrinkage/sqeezing toward the common
# The recommended value is the nearest integer to 50/(#samples - #groups) =
50/(4-2) = 25
cds <- estimateTagwiseDisp( cds , prior.n = 25 )
summary( cds$tagwise.dispersion ) # Min. 1st Qu.  Median    Mean 3rd Qu.
Max.
# 0.02885 0.03660 0.05068 0.07614 0.09980 0.65524

# Mean-variance plot
meanVarPlot <- plotMeanVar( cds , show.raw.vars=TRUE ,
                            show.tagwise.vars=TRUE ,
                            show.binned.common.disp.vars=FALSE ,
                            show.ave.raw.vars=FALSE ,
                            dispersion.method = "qcml" , NBline = TRUE ,
                            nbins = 100 ,
                            pch = 16 ,
                            xlab ="Mean Expression (Log10 Scale)" ,
                            ylab = "Variance (Log10 Scale)" ,
                            main = "Mean-Variance Plot" )
```




```
# Testing
de.cmn <- exactTest( cds , dispersion = "common" , pair = c( "WhiteVirgin"
, "Mutant" ) )
de.tgw <- exactTest( cds , dispersion = "tagwise" , pair = c( "WhiteVirgin"
, "Mutant" ) )
de.poi <- exactTest( cds , dispersion = 1e-06 , pair = c( "WhiteVirgin" ,
"Mutant" ) )
names( de.tgw )
# [1] "table"      "comparison" "genes"

de.tgw$comparison # which groups have been compared
# [1] "WhiteVirgin" "Mutant"

head( de.tgw$table ) # results table in order of your count matrix.
#                    logFC    logCPM       PValue
# FBgn0000003  0.36790690 12.482765 0.1493271642
# FBgn0000008 -0.27598545  6.102401 0.3199365793
# FBgn0000014  1.53789000  6.108545 0.0001165959
# FBgn0000015  1.41918363  3.172038 0.0230004687
# FBgn0000017  0.09620594  6.358934 0.7230523361
# FBgn0000018 -0.67927760  3.761421 0.1096707431

head( cds$counts )
#             WhiteVirgins57 WhiteVirgins58 Mutants59 Mutants60
```

```
# FBgn0000003          30423          29615     31789     45589
# FBgn0000008            470            429       300       443
# FBgn0000014            245            176       749       409
# FBgn0000015             32             25       101        42
# FBgn0000017            501            447       468       531
# FBgn0000018            111             84        49        73

# Significants genes
de.tgw_0.01<- topTags(de.tgw, p.value = 0.01, n = Inf)
dim(de.tgw_0.01)
# [1] 1827    4

# Top tags for tagwise analysis
options( digits = 3 ) # print only 3 digits
topTags( de.tgw , n = 20 , sort.by = "p.value" ) # top 20 DE genes

# Back to count matrix for tagwise analysis
cds$counts[ rownames( topTags( de.tgw , n = 15 )$table ) , ]

# Sort tagwise results by Fold-Change instead of p-value
resultsByFC.tgw <- topTags( de.tgw , n = nrow( de.tgw$table ) , sort.by =
"logFC" )$table
head( resultsByFC.tgw )

# Store full topTags results table
resultsTbl.cmn <- topTags( de.cmn , n = nrow( de.cmn$table ) )$table
resultsTbl.tgw <- topTags( de.tgw , n = nrow( de.tgw$table ) )$table
resultsTbl.poi <- topTags( de.poi , n = nrow( de.poi$table ) )$table
head( resultsTbl.tgw )

# Names/IDs of DE genes
de.genes.cmn <- rownames( resultsTbl.cmn[ resultsTbl.cmn$adj.P.Val <= 0.01
])
de.genes.tgw <- rownames( resultsTbl.tgw[ resultsTbl.tgw$adj.P.Val <= 0.01
])
de.genes.poi <- rownames( resultsTbl.poi[ resultsTbl.poi$adj.P.Val <= 0.01
])

# Amount significant
length( de.genes.cmn )
length( de.genes.tgw )
length( de.genes.poi )

# Percentage of total genes
length( de.genes.cmn ) / nrow( resultsTbl.cmn ) * 100
length( de.genes.tgw ) / nrow( resultsTbl.tgw ) * 100
length( de.genes.poi ) / nrow( resultsTbl.poi ) * 100

# Up/Down regulated summary for tagwise results
```

```
summary( decideTestsDGE( de.tgw , p.value = 0.01 ) ) # the adjusted p-values
are used here
#    WhiteVirgin+Mutant
# -1                646
# 0                8237
# 1                1181

# Visualizing
par( mfrow = c(2,1) )
plotSmear( cds , de.tags=de.genes.poi[1:500] , main="Poisson" ,
           pair=c("WhiteVirgin" , "Mutant") ,
           cex=.5 ,
           xlab="Log Concentration" , ylab="Log Fold-Change" )
abline( h=c(-2,2) , col="dodgerblue" )
plotSmear( cds , de.tags=de.genes.tgw[1:500] , main="Tagwise" ,
           pair=c("WhiteVirgin" , "Mutant") ,
           cex = .5 ,
           xlab="Log Concentration" , ylab="Log Fold-Change" )
abline( h=c(-2,2) , col="dodgerblue" )
par( mfrow=c(1,1) )
```



5. Graphs explanation

- **MDS plot:** It was observed that mutant sample were near in two dimensions but it was not the case for white virgins, because it showed their similarity in one dimension.
- **Mean variance plot:** this plot shows the importance to adjust variance. Grey dots show the raw variance of each count and blue dots, the variability is adjusted taking in a count common dispersion and values are binged.
- **MA plot:** the last one plot show DE genes in a negative binomial model and in a poisson model. Only the top 500 genes DE genes have a red colour and orange are genes in which count were zero in all samples.

Remarks: it was found 1827 genes DE in mutants flies whose 646 were down regulated and 1181 were upregulated.