RNAseq_Analysis

Tzu L. Phang

April 29, 2015

Contents

1 RNAseq Analysis

1.1 Background

This tutorial outline an RNAseq analysis routine conducted in my R + Biocondcutor data analysis course (BIOS6660) using one of my collaborator's data set, Dr. Eric Schmidt. Since RNAseq dataset is typically very large, we extracted only Chromosome 19 to ease perform the analysis on local computer (mostly laptops)

1.2 Experimental Design

Read in the descriptive data file and run summary statistics

```
## Read in descriptive data file
targets = read.table('./data/targets.txt', header = T)
## printing file table
knitr::kable(targets)
```

FileName	SampleName	Factor	Factor_long
CLP_lung_48h_rep1_chr19_read1.fastq	CLP_lung_48h_rep1	CLP	CLP_lung_48h_rep1
CLP_lung_48h_rep2_chr19_read1.fastq	CLP_lung_48h_rep2	CLP	CLP_lung_48h_rep2
CLP_lung_48h_rep3_chr19_read1.fastq	CLP_lung_48h_rep3	CLP	CLP_lung_48h_rep3
Sham_lung_48h_rep1_chr19_read1.fastq	Sham_lung_48h_rep1	Sham	Sham_lung_48h_rep1
Sham_lung_48h_rep2_chr19_read1.fastq	Sham_lung_48h_rep2	Sham	Sham_lung_48h_rep2
Sham_lung_48h_rep3_chr19_read1.fastq	Sham_lung_48h_rep3	Sham	Sham_lung_48h_rep3

It seems like we have a total of 6 FASTQ files and 2 experimental grouping (CLP vs. Sham) in mouse sample

1.3 Mapping FASTQ to genome

We will be using QuasR, which is an extremely versatile NGS mapping and postprocessing pipeline for RNA-seq It uses Rbowtie for upgapped alignments adn SpliceMap for spliced alignments

Note: QuasR is trying to be clever: if it finds BAM files already exists then it will not generate the file again. Therefore, to do a fresh run, need to delete everything in the "result" folder …

```
library(QuasR)
targets = read.table("./data/targets.txt", header = T)
write.table(targets[,1:2], 'data/QuasR_samples.txt', row.names=F, quote=F, sep='\t')
sampleFile = "./data/QuasR_samples.txt"
genomeFile = "./data/Mouse.chromosome.19.fa"
results = "./result"
cl = makeCluster(10)


## Single command to index reference, align all samples and generate BAM files
proj <- qAlign(sampleFile, genome=genomeFile, maxHits=1, splicedAlignment=T, alignmentsDir=results, clO
# Note: splicedAlignment should be set to TRUE when the reads are >=50nt long
alignstats <- alignmentStats(proj) # Alignment summary report
#knitr::kable(alignstats)
```

1.4 Alignment Summary

The following enumerates the number of reads in each FASTQ file and how many of them aligned to the reference. Note: the percentage of aligned reads is 100% in this particular example because only alignable reads were selected when generating the sample FASTQ files for this exercise. For QuasR this step can be omitted because the qAlign function generats this information automatically.

```
library(ShortRead); library(Rsamtools)
## Extract bam file names:
bam.filenames = proj@alignments$FileName

Nreads <- countLines(dirPath="./data/", pattern=".fastq$")/4
bfl <- BamFileList(bam.filenames, yieldSize=50000, index=character())
```

```
Nalign <- countBam(bfl, param=ScanBamParam(flag=scanBamFlag(isUnmappedQuery=F)))
(read_statsDF <- data.frame(FileName=names(Nreads), Nreads=Nreads, Nalign=Nalign$records, Perc_Aligned=
```

```
##                                                                    FileName
## CLP_lung_48h_rep1_chr19_read1.fastq   CLP_lung_48h_rep1_chr19_read1.fastq
## CLP_lung_48h_rep2_chr19_read1.fastq   CLP_lung_48h_rep2_chr19_read1.fastq
## CLP_lung_48h_rep3_chr19_read1.fastq   CLP_lung_48h_rep3_chr19_read1.fastq
## Sham_lung_48h_rep1_chr19_read1.fastq Sham_lung_48h_rep1_chr19_read1.fastq
## Sham_lung_48h_rep2_chr19_read1.fastq Sham_lung_48h_rep2_chr19_read1.fastq
## Sham_lung_48h_rep3_chr19_read1.fastq Sham_lung_48h_rep3_chr19_read1.fastq
##                                       Nreads  Nalign Perc_Aligned
## CLP_lung_48h_rep1_chr19_read1.fastq  1290664 1168495     90.53441
## CLP_lung_48h_rep2_chr19_read1.fastq  1324913 1200681     90.62338
## CLP_lung_48h_rep3_chr19_read1.fastq  1467134 1323209     90.19006
## Sham_lung_48h_rep1_chr19_read1.fastq 2094335 1905972     91.00607
## Sham_lung_48h_rep2_chr19_read1.fastq 1666375 1507205     90.44813
## Sham_lung_48h_rep3_chr19_read1.fastq 1955572 1783449     91.19833
```

```
write.table(read_statsDF, "./result/read_statsDF.txt", row.names=FALSE, quote=FALSE, sep="\t")

knitr::kable(read_statsDF)
```

	FileName	Nreads	Nalign	Perc_Align
CLP_lung_48h_rep1_chr19_read1.fastq	CLP_lung_48h_rep1_chr19_read1.fastq	1290664	1168495	90.534
CLP_lung_48h_rep2_chr19_read1.fastq	CLP_lung_48h_rep2_chr19_read1.fastq	1324913	1200681	90.623
CLP_lung_48h_rep3_chr19_read1.fastq	CLP_lung_48h_rep3_chr19_read1.fastq	1467134	1323209	90.190
Sham_lung_48h_rep1_chr19_read1.fastq	Sham_lung_48h_rep1_chr19_read1.fastq	2094335	1905972	91.006
Sham_lung_48h_rep2_chr19_read1.fastq	Sham_lung_48h_rep2_chr19_read1.fastq	1666375	1507205	90.448
Sham_lung_48h_rep3_chr19_read1.fastq	Sham_lung_48h_rep3_chr19_read1.fastq	1955572	1783449	91.198

1.5 Quality Report

The following shows how to create read quality reports with QuasR's qQCReport function or with the custom seeFastq function

```
qQCReport(proj, pdfFilename="qc_report.pdf")
source("http://faculty.ucr.edu/~tgirke/Documents/R_BioCond/My_R_Scripts/fastqQuality.R")
myfiles <- paste0("data/", targets$FileName); names(myfiles) <- targets$SampleName
fqlist <- seeFastq(fastq=myfiles, batchsize=50000, klength=8)
pdf("RNAseq_Results/fastqReport.pdf", height=18, width=4*length(myfiles)); seeFastqPlot(fqlist); dev.of
```

1.6 Store Annotations in TranscriptDb

Storing annotation ranges in TranscriptDb databases makes many operations more robust and convenient

```
library(GenomicFeatures)
txdb <- makeTranscriptDbFromGFF(file="./data/genes.gtf",
                                format="gtf",
                                dataSource="mm9",
                                species="Mus musculus")
saveDb(txdb, file="./data/mouse_annotation.sqlite")
txdb <- loadDb("./data/mouse_annotation.sqlite")
eByg <- exonsBy(txdb, by="gene")
```

1.7 Read Counting with countOverlaps

The gene signal was obtained by overlapping sequened reads onto the pre-defined gene region ranges. The resulting count table was further filtered to remove genes with zero counts.

```
samples <- as.character(targets$Factor_long)
samplespath <- bam.filenames
names(samplespath) <- samples
countDF <- data.frame(row.names=names(eByg))
for(i in samplespath) {
  aligns <- readGAlignmentsFromBam(i) # Substitute next two lines with this one.
  seqnames(aligns) = rep('chr19', length(aligns))
  counts <- countOverlaps(eByg, aligns, ignore.strand=TRUE)
  countDF <- cbind(countDF, counts)
}
colnames(countDF) <- samples

## Remove row with all zeros
row.sum = rowSums(countDF)
chr19.countDF = countDF[row.sum != 0,]

chr19.countDF[1:4,]
```

```
##                CLP_lung_48h_rep1 CLP_lung_48h_rep2 CLP_lung_48h_rep3
## 1110059E24Rik                  2                 0                 1
## 1500015L24Rik                  1                 4                 0
## 1500017E21Rik                  0                 1                 0
## 1700001K23Rik                 40                21                33
##                Sham_lung_48h_rep1 Sham_lung_48h_rep2 Sham_lung_48h_rep3
## 1110059E24Rik                   1                  1                  3
## 1500015L24Rik                   2                  2                  5
## 1500017E21Rik                   2                  0                  1
## 1700001K23Rik                  34                 28                 29
```

```
write.table(chr19.countDF, "./result/chr19_countDF", quote=FALSE, sep="\t", col.names = NA)
countDF <- read.table("./result/chr19_countDF")
```

1.8 Simple RPKM Normalization

RPKM: reads per kilobase of exon model per million mapped reads

```
returnRPKM <- function(counts, gffsub) {
  geneLengthsInKB <- sum(width(reduce(gffsub)))/1000 # Length of exon union per gene in kbp
  millionsMapped <- sum(counts)/1e+06 # Factor for converting to million of mapped reads.
  rpm <- counts/millionsMapped # RPK: reads per kilobase of exon model.
  rpkm <- rpm/geneLengthsInKB # RPKM: reads per kilobase of exon model per million mapped reads.
  return(rpkm)
}
countDFrpkm <- apply(countDF, 2, function(x) returnRPKM(counts=x, gffsub=eByg[rownames(countDF)]))
countDFrpkm[1:4,]
```

```
##                CLP_lung_48h_rep1 CLP_lung_48h_rep2 CLP_lung_48h_rep3
## 1110059E24Rik           1.779027          0.000000           0.84371
## 1500015L24Rik           1.632680          6.627559           0.00000
## 1500017E21Rik           0.000000          0.484245           0.00000
## 1700001K23Rik         104.116816         55.471850          81.47334
##                Sham_lung_48h_rep1 Sham_lung_48h_rep2 Sham_lung_48h_rep3
## 1110059E24Rik           0.5254325          0.6985178          1.6985821
## 1500015L24Rik           1.9288369          2.5642245          5.1961767
## 1500017E21Rik           0.5637247          0.0000000          0.3037284
## 1700001K23Rik          52.2762243         57.2326512         48.0475975
```

1.9 Reproducibility Check by Sample-Wise Clustering

QC check of the sample reproducibility by computing a correlating matrix and plotting it as a tree. Note: the plotMDS function from edgeR is a more robust method for this task.

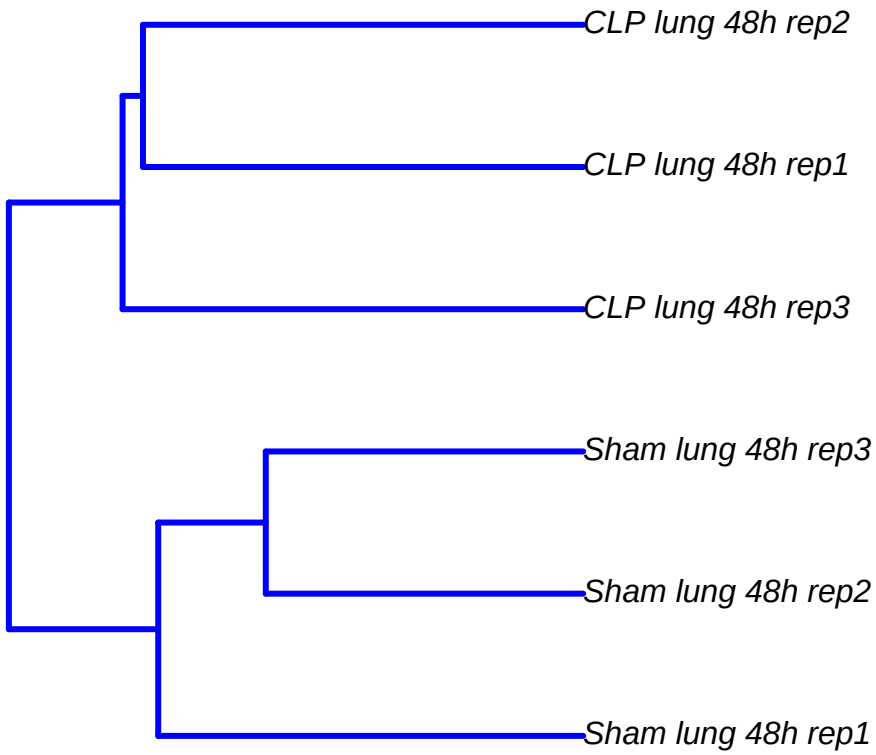
```
library(ape)
d <- cor(countDFrpkm, method="spearman")
hc <- hclust(dist(1-d))
plot.phylo(as.phylo(hc), type="p", edge.col=4, edge.width=3, show.node.label=TRUE, no.margin=TRUE)
```

As expected, CLP and Sham samples clustered together respectively

---

1.10 Identify DEGs with Simple Fold Change Method

```
## Compute mean values for replicates
source("http://faculty.ucr.edu/~tgirke/Documents/R_BioCond/My_R_Scripts/colAg.R")
countDFrpkm_mean <- colAg(myMA=countDFrpkm, group=c(1,1,1,2,2,2), myfct=mean)
countDFrpkm_mean[1:4,]
```




```
##                CLP_lung_48h_rep1_CLP_lung_48h_rep2_CLP_lung_48h_rep3
## 1110059E24Rik                                                0.8742456
## 1500015L24Rik                                                2.7534129
## 1500017E21Rik                                                0.1614150
## 1700001K23Rik                                               80.3540010
##                Sham_lung_48h_rep1_Sham_lung_48h_rep2_Sham_lung_48h_rep3
## 1110059E24Rik                                                 0.9741774
## 1500015L24Rik                                                 3.2297461
## 1500017E21Rik                                                 0.2891510
## 1700001K23Rik                                                52.5188243
```

```
## Log2 fold change
countDFrpkm_mean <- cbind(countDFrpkm_mean, log2ratio=log2(countDFrpkm_mean[,2]/countDFrpkm_mean[,1]))
countDFrpkm_mean <- countDFrpkm_mean[is.finite(countDFrpkm_mean[,3]), ]
degs2fold <- countDFrpkm_mean[countDFrpkm_mean[,3] >= 1 | countDFrpkm_mean[,3] <= -1,]
degs2fold[1:4,]
```

```
##                CLP_lung_48h_rep1_CLP_lung_48h_rep2_CLP_lung_48h_rep3
## 1700019N19Rik                                                0.4351094
## 4930524O05Rik                                                3.5604664
## 5730408K05Rik                                                3.2674506
## 5830416P10Rik                                                0.6330198
##                Sham_lung_48h_rep1_Sham_lung_48h_rep2_Sham_lung_48h_rep3
## 1700019N19Rik                                                 1.6436593
## 4930524O05Rik                                                17.6887380
## 5730408K05Rik                                                 7.6869962
## 5830416P10Rik                                                 0.1256593
##                log2ratio
```

```
## 1700019N19Rik  1.917461
## 4930524O05Rik  2.312693
## 5730408K05Rik  1.234255
## 5830416P10Rik -2.332733
```

```
write.table(degs2fold, "./result/degs2fold.xls", quote=FALSE, sep="\t", col.names = NA)
degs2fold <- read.table("./result/degs2fold.xls")
```

1.11 Identify DEGs with DESeq Library

```
library(DESeq)
countDF <- read.table("./result/chr19_countDF")
conds <- targets$Factor
cds <- newCountDataSet(countDF, conds) # Creates object of class CountDataSet derived from eSet class
counts(cds)[1:4, ] # CountDataSet has similar accessor methods as eSet class.
```

```
##                CLP_lung_48h_rep1 CLP_lung_48h_rep2 CLP_lung_48h_rep3
## 1110059E24Rik                  2                 0                 1
## 1500015L24Rik                  1                 4                 0
## 1500017E21Rik                  0                 1                 0
## 1700001K23Rik                 40                21                33
##                Sham_lung_48h_rep1 Sham_lung_48h_rep2 Sham_lung_48h_rep3
## 1110059E24Rik                   1                  1                  3
## 1500015L24Rik                   2                  2                  5
## 1500017E21Rik                   2                  0                  1
## 1700001K23Rik                  34                 28                 29
```

```
cds <- estimateSizeFactors(cds) # Estimates library size factors from count data. Alternatively, one ca
cds <- estimateDispersions(cds) # Estimates the variance within replicates
res <- nbinomTest(cds, "CLP", "Sham") # Calls DEGs with nbinomTest
res <- na.omit(res)
res2fold <- res[res$log2FoldChange >= 1 | res$log2FoldChange <= -1,]
res2foldpadj <- res2fold[res2fold$padj <= 0.05, ]
res2foldpadj[1:4,1:8]
```

```
##         id   baseMean   baseMeanA  baseMeanB foldChange log2FoldChange
## 43   Acta2  287.77458  384.673260  190.87591  0.4962027      -1.010999
## 101 Carns1  119.19376   75.208883  163.17865  2.1696725       1.117477
## 111    Cd5   11.23337    4.651678   17.81507  3.8298153       1.937275
## 134  Cpt1a 3664.99583 5109.789409 2220.20224  0.4344998      -1.202573
##              pval         padj
## 43  1.524263e-13 1.295624e-11
## 101 3.173251e-09 1.244891e-07
## 111 7.125290e-04 8.863166e-03
## 134 8.638185e-18 1.468491e-15
```

1.12 Identify DEGs with edgeR's Exact Method

```
library(edgeR)
countDF <- read.table("./result/chr19_countDF")
y <- DGEList(counts=countDF, group=conds) # Constructs DGEList object
y <- estimateCommonDisp(y) # Estimates common dispersion
y <- estimateTagwiseDisp(y) # Estimates tagwise dispersion
et <- exactTest(y, pair=c("CLP", "Sham")) # Computes exact test for the negative binomial distribution.
topTags(et, n=4)
```

```
## Comparison of groups:  Sham-CLP
##              logFC    logCPM       PValue          FDR
## Slc22a12 -3.420277  5.888933 1.705319e-36 8.697127e-34
## Ms4a18   -2.037469  7.191865 6.677911e-32 1.702867e-29
## Ms4a15   -1.789720  7.243905 2.055271e-24 3.493961e-22
## Cpt1a    -1.465574 11.952793 8.843267e-24 1.127517e-21
```

```
edge <- as.data.frame(topTags(et, n=50000))
edge2fold <- edge[edge$logFC >= 1 | edge$logFC <= -1,]
edge2foldpadj <- edge2fold[edge2fold$FDR <= 0.01, ]
```

1.13 Idenfity DEGs with edgeR's GLM Approach

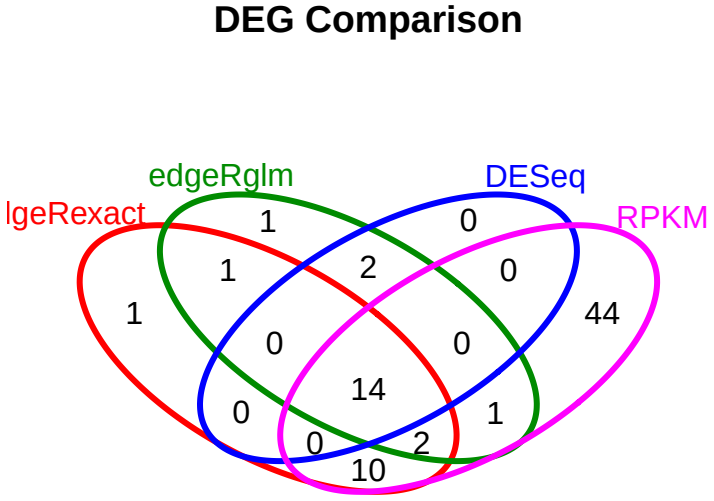
```
library(edgeR)
countDF <- read.table("./result/chr19_countDF")
y <- DGEList(counts=countDF, group=conds) # Constructs DGEList object
## Filtering and normalization
keep <- rowSums(cpm(y)>1) >= 2; y <- y[keep, ]
y <- calcNormFactors(y)
design <- model.matrix(~0+group, data=y$samples); colnames(design) <- levels(y$samples$group) # Design
## Estimate dispersion
y <- estimateGLMCommonDisp(y, design, verbose=TRUE) # Estimates common dispersions
```

```
## Disp = 0.00727 , BCV = 0.0853
```

```
y <- estimateGLMTrendedDisp(y, design) # Estimates trended dispersions
y <- estimateGLMTagwiseDisp(y, design) # Estimates tagwise dispersions
## Fit the negative binomial GLM for each tag
fit <- glmFit(y, design) # Returns an object of class DGEGLM
contrasts <- makeContrasts(contrasts="CLP-Sham", levels=design) # Contrast matrix is optional
lrt <- glmLRT(fit, contrast=contrasts[,1]) # Takes DGEGLM object and carries out the likelihood ratio t
edgeglm <- as.data.frame(topTags(lrt, n=length(rownames(y))))
## Filter on fold change and FDR
edgeglm2fold <- edgeglm[edgeglm$logFC >= 1 | edgeglm$logFC <= -1,]
edgeglm2foldpadj <- edgeglm2fold[edgeglm2fold$FDR <= 0.01, ]
```

1.14 Comparison Among DEG Results

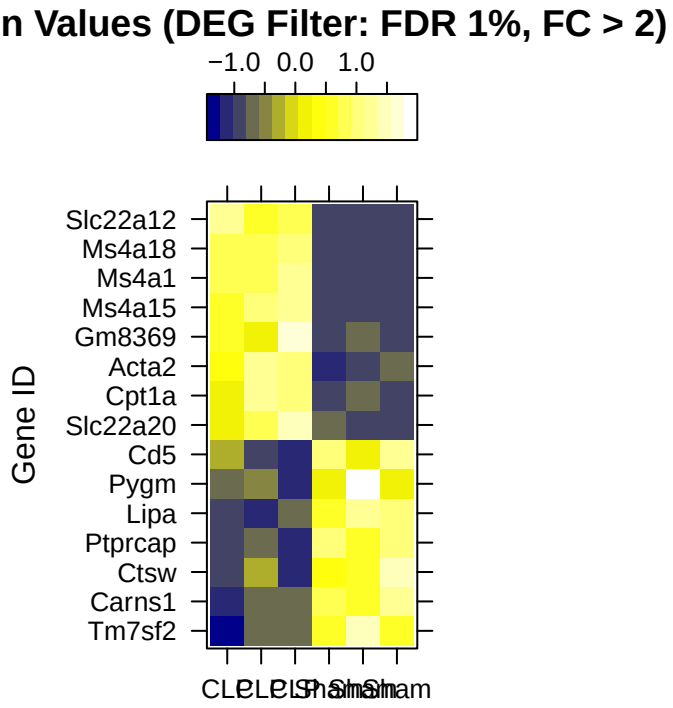
```
source("http://faculty.ucr.edu/~tgirke/Documents/R_BioCond/My_R_Scripts/overLapper.R")
setlist <- list(edgeRexact=rownames(edge2foldpadj), edgeRglm=rownames(edgeglm2foldpadj), DESeq=as.chara
OLlist <- overLapper(setlist=setlist, sep="_", type="vennsets")
counts <- sapply(OLlist$Venn_List, length)
vennPlot(counts=counts, mymain="DEG Comparison")
```



Number of common genes among all 4 methods: 14

1.15 Heatmap of Top Ranking DEGs

```
library(lattice); library(gplots)
y <- countDFrpkm[rownames(edgeglm2foldpadj)[1:15],]
colnames(y) <- targets$Factor
y <- t(scale(t(as.matrix(y))))
y <- y[order(y[,1]),]
levelplot(t(y), height=0.2, col.regions=colorpanel(40, "darkblue", "yellow", "white"), main="Expression
```


Expression Values (DEG Filter: FDR 1%, FC > 2)
−1.0
0.0
1.0
Gene ID
Slc22a12
Ms4a18
Ms4a1
Ms4a15
Gm8369
Acta2
Cpt1a
Slc22a20
Cd5
Pygm
Lipa
Ptprcap
Ctsw
Carns1
Tm7sf2