Created using following resources:

<http://combine-australia.github.io/RNAseq-R/06-rnaseq-day1.html>

<https://bioinformatics-core-shared-training.github.io/cruk-bioinf-sschool/Day3/rnaSeq_DE.pdf>

<https://f1000research.com/articles/5-1438/v1>

<https://github.com/mistrm82/msu_ngs2015/blob/master/hands-on.Rmd>

Data file downloaded from:

<https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE98877>

Data file:

GSE98877\_gf\_vs\_spf\_rna\_seq.txt

Packages used:

edgeR

limma

gplots

RColorBrewer

Overview:

Reading in table of counts

Create DGEList

Filtering lowly expressed genes

Quality control

Normalization for composition bias

Design matrix

Estimate dispersions

Differential expression

#Reading in data

setwd("/users/aswaniunnikrishnan/project2/GSE98877")

seqdata <- read.delim("GSE98877\_gf\_vs\_spf\_rna\_seq.txt",stringsAsFactors = FALSE)

seqdata

dim(seqdata)

[1] 15448 17

library(edgeR)

#remove 1st column from seqdata

countdata <- seqdata[,-(1)]

dim(countdata)

[1] 15448 16

# store id's as rownames

rownames(countdata) <- seqdata[,1]

colnames(countdata) <- c("SPF01","SPF02","SPF03","SPF04","GF01","GF02","GF03","GF04","GF05","GF06","GF07","GF08","GF09","SPF05","SPF06","SPF07")

colnames(countdata)

#creating a DGEList object to hold read counts

dgList <- DGEList(counts = countdata, genes = rownames(countdata))

dgList

dgList$samples

group lib.size norm.factors

SPF01 1 37263137 1

SPF02 1 35400221 1

SPF03 1 35215097 1

SPF04 1 36216286 1

GF01 1 34618094 1

GF02 1 34773750 1

GF03 1 36459556 1

GF04 1 33407698 1

GF05 1 34864851 1

GF06 1 35199431 1

GF07 1 33009700 1

GF08 1 34656811 1

GF09 1 31263149 1

SPF05 1 38279693 1

SPF06 1 35304420 1

SPF07 1 37245693 1

dim(dgList)

head(dgList$counts)

head(dgList$genes)

#filtering

CountsPerMillion <- cpm(dgList)

summary(CountsPerMillion)

countcheck <- CountsPerMillion >1

head(countcheck)

keep <- which(rowSums(countcheck) >= 2)

dgList <- dgList[keep,]

#quality control

dgList$sample$lib.size

barplot(y$samples$lib.size,names=colnames(y),las=2)

title("Barplot of library sizes")

logcounts <-cpm(dgList, log=TRUE)

boxplot(logcounts,xlab="",ylab="Log2 counts per million",las=2)

abline(h=median(logcounts),col ="blue")

title("Boxplot of logCPMs- unnormalised")

#Normalization- TMM normalization is performed to eliminate composition biases between libraries. The calcNormFactors function calculates the normalization factors between libraries.

dgList <- calcNormFactors(dgList, method = "TMM")

dim(dgList)

[1] 15412 16

dgList

An object of class "DGEList"

$counts

SPF01 SPF02 SPF03

ENSMUSG00000000001.4 8445.4718 9320.6756 9607.3103

ENSMUSG00000000028.14 391.7259 371.8243 568.2848

ENSMUSG00000000031.15 45452.8017 29863.0494 30186.7658

ENSMUSG00000000037.16 215.0893 230.1412 247.0418

ENSMUSG00000000056.7 2554.3402 2583.9224 2447.5681

SPF04 GF01 GF02

ENSMUSG00000000001.4 8850.9185 9244.0350 10230.7916

ENSMUSG00000000028.14 294.6428 628.3952 315.8086

ENSMUSG00000000031.15 33749.1133 10869.4530 17495.4825

ENSMUSG00000000037.16 281.7297 148.3031 167.6438

ENSMUSG00000000056.7 2993.8385 1935.6839 2276.2490

logcounts2 <-cpm(dgList, log=TRUE)

boxplot(logcounts2,xlab="",ylab="Log2 counts per million",las=2)

abline(h=median(logcounts2),col ="blue")

title("Boxplot of logCPMs- normalised")

#Data Exploration – multidimensional scaling plots -  An MDSplot is a visualisation of a principle components analysis, which determines the greatest sources of variation in the data. It is also an incredibly useful tool for quality control and checking for outliers.  - examine inter-sample relationships

plotMDS(dgList)

#setting up the model

sampleType <- rep("GF", ncol(dgList))

sampleType[grep("SPF", colnames(dgList))] <- "SPF"

designMat <- model.matrix(~0 + sampleType)

designMat # The design matrix records which treatment conditions were applied to each samples,

sampleTypeGF sampleTypeSPF

1 0 1

2 0 1

3 0 1

4 0 1

5 1 0

6 1 0

7 1 0

8 1 0

9 1 0

10 1 0

11 1 0

12 1 0

13 1 0

14 0 1

15 0 1

16 0 1

attr(,"assign")

[1] 1 1

attr(,"contrasts")

attr(,"contrasts")$sampleType

[1] "contr.treatment"

#Estimating Dispersions

estimate the dispersion parameter for negative binomial model , plot the estimates and see how they differ The biological coefficient of variation (BCV) is the square root of the dispersion parameter in the negative binomial model.

dgList <- estimateGLMCommonDisp(dgList, design= designMat)

dgList <- estimateGLMTrendedDisp(dgList, design = designMat)

dgList <- estimateGLMTagwiseDisp(dgList, design = designMat)

plotBCV(dgList)

#Differential expression

# edgeR

fit <- glmFit(dgList, designMat)

lrt <- glmLRT(fit, coef = 2)

lrt

edgeR\_result <- topTags(lrt)

edgeR\_result

save(edgeR\_result)$table, file=”edgeR\_result.RData”)

deGenes <- decideTestsDGE(lrt, p=0.1)

deGenes <- rownames(lrt)[as.logical(deGenes)]

plotSmear(lrt,de.tags = deGenes) # plot the log-fold changes of all the genes, and

highlight those that are differentially expressed.

#voom limma – using lmFit -  limma package offers the voom function, which transforms the read counts into logCPMs while taking into account the mean-variance relationship in the data.

v <- voom(dgList,designMat,plot=T)

fit <- lmFit(v) #fit the linear model, lmFit estimates group means according to the design matrix, as well as gene-wise variances.

names(fit)

[1] "coefficients" "stdev.unscaled" "sigma"

[4] "df.residual" "cov.coefficients" "pivot"

[7] "rank" "genes" "Amean"

[10] "method" "design"

#makecontrasts for comparisons

cont.matrix <- makeContrasts(gfvsspf= sampleTypeGF - sampleTypeSPF, levels = designMat)

cont.matrix

Contrasts

Levels gfvsspf

sampleTypeGF 1

sampleTypeSPF -1

fit.cont <- contrasts.fit(fit, cont.matrix)

fit.cont <-eBayes(fit.cont)

dim(fit.cont)

[1] 15412 1

summa.fit <- decideTests(fit.cont)

summary(summa.fit)

gfvsspf

Down 808

NotSig 14206

Up 398

plotMD(fit.cont,coef = 1,status = summa.fit[,"gfvsspf"],values =c(1,-1))

#using glmQLFit (Quasi logFit)

fit <- glmQLFit(dgList,designMat)

head(fit$coefficients)

summary(fit$df.prior)

gfvsspf <- makeContrasts(sampleTypeGF - sampleTypeSPF, levels = designMat)

res <- glmQLFTest(fit, contrast = gfvsspf)

topTags(res)

is.de <- decideTestsDGE(res)

summary(is.de)

1\*sampleTypeGF -1\*sampleTypeSPF

Down 746

NotSig 13626

Up 1040