TABLE OF CONTENTS

[1 Introduction 2](#_Toc97919889)

[2 Differential expression for replicated samples (DESeq2) 2](#_Toc97919890)

[2.1 Installation 2](#_Toc97919891)

[2.2 Preparing count matrix 3](#_Toc97919892)

[2.3 Run DESeq2 4](#_Toc97919893)

[2.4 Output file of DESeq2 5](#_Toc97919894)

[3 Differential expression for non-replicated samples (edgeR) 6](#_Toc97919895)

[3.1 Installation 6](#_Toc97919896)

[3.2 Preparing count matrix 6](#_Toc97919897)

[3.3 Run edgeR 6](#_Toc97919898)

[3.4 Output file of edgeR 7](#_Toc97919899)

[4 Time series analysis (Mfuzz) 8](#_Toc97919900)

[4.1 Installation 8](#_Toc97919901)

[4.2 Preparing count matrix 8](#_Toc97919902)

[4.3 Run Mfuzz 9](#_Toc97919903)

[4.4 Output files for Mfuzz 9](#_Toc97919904)

[4.4.1 output cluster figure 9](#_Toc97919905)

[4.4.2 output cluster information 11](#_Toc97919906)

[4.4.3 output expression 11](#_Toc97919907)

[5 Volcano plot 11](#_Toc97919908)

[5.1 Installation 11](#_Toc97919909)

[5.2 Run EnhancedVolcano 11](#_Toc97919910)

[5.2.1 R script for result from DESeq2: 12](#_Toc97919911)

[5.2.2 R script for result from edgeR: 12](#_Toc97919912)

[5.3 Output files 12](#_Toc97919913)

[6 Heatmap 13](#_Toc97919914)

[6.1 Installation 13](#_Toc97919915)

[6.2 Preparing count matrix 14](#_Toc97919916)

[6.3 Run the pheatmap 14](#_Toc97919917)

[6.4 Output files 14](#_Toc97919918)

# 1 Introduction

Transcriptome analysis can measure the whole gene expression at the same time and space, so detecting gene expression differences in different tissues or different time points is the most basic and cost-effective method of molecular biology. MGI MegaBOLT RNA-seq pipeline can quickly analyze and detect all gene expression in a sample (normally stored as FPKM file), and this App-note introduces how to use MegaBOLT expression results to do the analysis.

Usually, we recommend differential expression analysis on replicates (e.g., triplicated samples), so that we can calculate the statistics p-value to prove that the difference between the two samples is valid and not caused by sampling error. However, in some special cases (such as sparse samples), we can only perform a single sample sequencing without repetition. This situation is also included in this app-note.

Time series analysis is a very important field in data science. It interprets the development process of individuals or tissues by counting gene expression at different times at the same spatial location, usually time series data will include parts with patterns and irregular parts, and what we need to do is to eliminate the irregular parts and find the pattern, then make a prediction.

The Scripts and test data are available at GitHub:

<https://github.com/MGI-APAC-FBS/App-Note_RNA-seq>

# 2 Differential expression for replicated samples (DESeq2)

Normally people use DESeq2 for he replicated samples analysis. In this case, an example with a batch of triplicated samples would be used for the analysis.

## 2.1 Installation

The easiest way is to use Bioconductor for the installation. We just need to find an appropriate Bioconductor version for your R system, i.e., Bioconductor 3.14 is only for R4.1, and 3.10 is for R3.6.

In this case we will use R3.6, which is widely used in most laboratories. Please refer to the following linkage for the installation:

<https://bioconductor.org/packages/release/bioc/html/DESeq2.html>

In some cases, there maybe some software missed in your operating system, i.e., you cannot install XML package on R-Studio Pro, which gives an error as:

Text

Description automatically generated

It means the libxml2 is missing in your operating system, so what you should do is to install it separately, if the system you are using is CentOS, you should install it in command as:

yum install libxml2

## 2.2 Preparing count matrix

The input file for DESeq2 is in the form of a matrix of integer values. The first column is genes id, and the following ones are the expression for different samples, which **MUST BE** raw counts of sequencing reads. For example:

A picture containing scatter chart

Description automatically generated

To use MegaBOLT data in DESeq2, we need to extract the reads counts for different samples and store them in a single file. For example, the format of FPKM result in MegaBOLT is from RSEM:

Text

Description automatically generated

The data we need for each sample should be from the 1st and 4th column (need to change to integer format):

Graphical user interface

Description automatically generated with medium confidence

By doing so for all samples, the Metrix would be:

Chart

Description automatically generated with medium confidence

There is a script for the generating of expression matrix available in GitHub, you can go to the folder ‘01\_extract\_Exp\_From\_RSEM’ in the following website:

<https://github.com/MGI-APAC-FBS/App-Note_RNA-seq>

## 

## 2.3 Run DESeq2

In this step, you can use the input file generated in last step to do the analysis. The analysis need to be done on R system, so you need to prepare a R script first:

library("DESeq2")

countdata <- read.table("./all\_exp.txt",sep="\t",header=T) **## input file**

**## set gene names as the row names**

len <- length(countdata)

rownames(countdata) <- countdata[,1]

countdata <- countdata[,2:len]

**## 2 controls + 3 treatments**

type <- c("Control","Control","Treat","Treat","Treat")

coldata <- data.frame(type)

**## execute DESeq2**

dds <- DESeqDataSetFromMatrix(countData=countdata, colData=coldata, design = ~ type)

dds <- DESeq(dds)

result <- results(dds)

write.csv(result, file="./DiffGeneExp\_DESeq2.csv") **## output csv file, can be opened by Excel**

**## extract differentially expressed genes, by setting 'padj < 0.05 && log2FoldChange > 1 or < -1'**

filter\_deseq2 <- subset(result, padj < 0.05 & (log2FoldChange > 1 | log2FoldChange < -1))

write.csv(filter\_deseq2, file="./DiffGeneExp\_DESeq2\_Filter.csv") **## output csv file, can be opened by Excel**

## 

## 2.4 Output file of DESeq2

The output file contails the information for differentially expressed genes. The format for the output file is:

Diagram

Description automatically generated with low confidence

You can get information on the meaning of the columns by checking the ‘result’, which is a DataFram object:

Text

Description automatically generated

The DESeq2 also provide many types of analysis, such as Summary, plot Counts, volcano plot and PCA. You can find the instruction in:

<https://lashlock.github.io/compbio/R_presentation.html>

<https://bioc.ism.ac.jp/packages/2.14/bioc/vignettes/DESeq2/inst/doc/beginner.pdf>

# 

# 3 Differential expression for non-replicated samples (edgeR)

For some samples which is not suitable to do replicated sequencing, the log2foldchange should be considered, which can be done directly by comparing the FPKM result from 2 samples. There is also a software ‘edgeR’ can be used for this type of comparison. In edgeR, the p-value can be generated, but the p-value is of less reference.

## 3.1 Installation

Like DESeq2, the edgeR can be installed in using of Bioconductor. The way of installation is:

A picture containing rectangle

Description automatically generated

## 3.2 Preparing count matrix

Since there is no replicated samples being sequenced, the matrix should contains 3 columns (this can be done by using the same scripts in 2.2). In this case, we use C1 and T1 data prepared for the 2.2, which is:

Chart

Description automatically generated

## 3.3 Run edgeR

Like DESeq2, to run the edgeR, you need to prepare a R script:

library(edgeR)

**## input count matrix**

countdata <- read.table("./all\_exp.txt",sep="\t",header=T) **## input file**

**## set gene names as the row names**

len <- length(countdata)

rownames(countdata) <- countdata[,1]

countdata <- countdata[,2:len]

**## make DGEList**

type <- c("Control","Treat") **## one control, one treatment**

y<-DGEList(counts=countdata,group=type)

**## filter genes with low expression**

keep = filterByExpr(y)

y = y[keep,,keep.lib.sizes = FALSE]

**## standardization**

y = calcNormFactors(y)

**## differential expressed genes calculation, use a loose BCV value**

y\_bcv <- y

bcv <- 0.2

diffExp <- exactTest(y\_bcv,dispersion = bcv ^ 2)

**## extract result**

result = topTags(diffExp, n =90000)

write.csv(result, file="./DiffGeneExp\_edgeR.csv") **## output csv file, can be opened by Excel**

**## extract differentially expressed genes, by setting 'FDR < 0.05 && logFC > 1 or < -1'**

result<- as.data.frame(result)

result\_DEG = subset(result, FDR < 0.05 & (logFC > 1 | logFC < -1))

write.csv(result\_DEG, file="./DiffGeneExp\_edgeR\_Filter.csv") **## output csv file, can be opened by Excel**

## 3.4 Output file of edgeR

The output file of edgeR includes the log2foldchange and other information.

Apart from the widely used value (logFC, pValue, FDR), there is also a value named logCPM, which is the log2 counts-per-million, which can be understood as measuring expression level.

Text

Description automatically generated

For more information, please refer to the instruction in Bioconductor:

<https://bioconductor.org/packages/release/bioc/vignettes/edgeR/inst/doc/edgeRUsersGuide.pdf>

# 

# 4 Time series analysis (Mfuzz)

The common way for time series analysis is to use gene expression of time series sample to build the expression pattern, normally heatmap. The most classical way is Mfuzz, which is a soft clustering function based on fuzzy c-means. It groups genes based on the Euclidean distance and the c-means objective function which is a weighted square error function. Each gene is assigned a membership value between 0 and 1 for each cluster. Hence, genes can be assigned to different clusters in a gradual manner.

## 4.1 Installation

A picture containing text

Description automatically generated

## 4.2 Preparing count matrix

In this case, we prepared samples from 5 time point (this can be done by using the same scripts in 2.2). The first part of the file should be like:

Chart

Description automatically generated with medium confidence

## 

## 4.3 Run Mfuzz

library("Mfuzz")

**## input matrix**

data<-table2eset("./all\_exp.txt") **## input file**

**## data triming, get rid of odd number**

data.r <- filter.NA(data, thres=0.25)

data.m <- fill.NA(data.r,mode="mean")

data.f <- filter.std(data.m,min.std=0.05,visu=F)

**## standardization**

data.s <- standardise(data.f)

**## cluster based on fuzzy c-means**

cl <- mfuzz(data.s,c=12,m=1.25)

**## plot**

pdf("TimeSeries.mfuzz.plot.pdf",width=7,height=9) **## output cluster figure**

mfuzz.plot2(data.s,cl=cl,mfrow=c(4,3),min.mem=0.75,time.labels=c("day1","day2","day3","day4","day5"),x11 = FALSE)

dev.off()

pdf("TimeSeries.mfuzz.plot.split.pdf",width=7,height=9) **## another cluster figure, one figure per page**

mfuzz.plot2(data.s,cl=cl,mfrow=c(1,1),min.mem=0.75,time.labels=c("day1","day2","day3","day4","day5"),x11 = FALSE)

**## output**

cluster<-cl$cluster

expStandard<-exprs(data.s)

write.table(cluster,file="TimeSeries.mfuzz.cluster") **## output cluster information**

write.table(expStandard,file="TimeSeries.mfuzz.expStandard") **## out up/down regulation across different samples**

dev.off()

The Mfuzz is run on R system, so you need to prepare a R script.

## 4.4 Output files for Mfuzz

### 4.4.1 output cluster figure

In this case, we have 2 figures for the cluster, the first one is for the summary and another one gives detailed information for each cluster. Here is an example for the cluster result:

Logo, company name

Description automatically generated

From this figure we can see that 12 clusters was generated and each cluster represent one expression pattern.

The second figure gives amplified clusters information, for example:

Chart

Description automatically generated with low confidence

### 4.4.2 output cluster information

If you want to have more information for a single gene, you can go to the file timeSeries.cluster which contains the linkage between gene and cluster:

A picture containing text

Description automatically generated

### 4.4.3 output expression

This file gives more information about the expression pattern：Text

Description automatically generated

# 5 Volcano plot

The volcano plot would be easier for the interpretation of DEG result. There are many software and methods to generate this figure, in this case, we will use the R packages ‘EnhancedVolcano’ for the plotting.

## 5.1 Installation



## 5.2 Run EnhancedVolcano

The inputs can be directly from the output of DESeq2 or edgeR, to make a volcano plot, you need to prepare a R script first:

### 5.2.1 R script for result from DESeq2:

library(EnhancedVolcano)

**## input the result from DESeq2**

inputFileName="./DiffGeneExp\_DESeq2.csv"

exp\_data <- read.csv(inputFileName, header=TRUE)

**## set gene names as the row names**

len <- length(exp\_data)

rownames(exp\_data) <-exp\_data[,1]

exp\_data <- exp\_data[,2:len]

**## change values to numeric**

exp\_data$log2FoldChange=as.numeric(exp\_data$log2FoldChange)

exp\_data$padj=as.numeric(exp\_data$padj)

**## plot the figure, for log2FlodChange and padj.**

pdf("Volcano\_For\_DESeq2.pdf")

EnhancedVolcano(exp\_data,

lab = rownames(exp\_data),

x = 'log2FoldChange',

y = 'padj')

library(EnhancedVolcano)

inputFileName="./DEG\_edgeR.csv"

# Import edgeR data

volcanodata <- read.csv(inputFileName, header=TRUE)

len <- length(volcanodata)

rownames(volcanodata) <-volcanodata[,1]

volcanodata <- volcanodata[,2:len]

volcanodata$logFC=as.numeric(volcanodata$logFC)

volcanodata$FDR=as.numeric(volcanodata$FDR)

head(volcanodata)

pdf("aaa.pdf")

EnhancedVolcano(volcanodata,

lab = rownames(volcanodata),

x = 'logFC',

y = 'FDR')

### 5.2.2 R script for result from edgeR:

library(EnhancedVolcano)

**## input the result from edgeR**

inputFileName="./DiffGeneExp\_edgeR.csv"

exp\_data <- read.csv(inputFileName, header=TRUE)

**## set gene names as the row names**

len <- length(exp\_data)

rownames(exp\_data) <-exp\_data[,1]

exp\_data <- exp\_data[,2:len]

**## change values to numeric**

exp\_data$logFC=as.numeric(exp\_data$logFC)

exp\_data$FDR=as.numeric(exp\_data$FDR)

**## plot the figure, for log2FlodChange and FDR.**

pdf("Volcano\_For\_edgeR.pdf")

EnhancedVolcano(exp\_data,

lab = rownames(exp\_data),

x = 'logFC',

y = 'FDR')

## 5.3 Output files

This case shows the most basic volcano plot, for more functions, please visit the Bioconductor:

<https://bioconductor.org/packages/release/bioc/vignettes/EnhancedVolcano/inst/doc/EnhancedVolcano.html>

In the following picture, the default cut-off for log2FC is >|2|; the default cut-off for P value is 10e-6.

Chart, scatter chart

Description automatically generated

# 6 Heatmap

After we got the differentially expressed genes, a heatmap would be more intuitive and effective for the display of these genes. In this case, the R package ‘pheatmap’ would be used for the picture generating.

## 6.1 Installation

For a typical installation, you just need to open R and install the packages by:

Graphical user interface, text, application, email

Description automatically generated

## 6.2 Preparing count matrix

The only thing needed for the input is the expression for all samples. Normally we will use the differentially expressed genes for the plot, so the subset of the whole data should be used, the format for it is:

Table

Description automatically generated

## 6.3 Run the pheatmap

To run the pheatmap, a R script should be prepared:

library(pheatmap)

**## input count matrix**

countdata <- read.table("./all\_exp.txt",sep="\t",header=T) ## input file

**## set gene names as the row names**

len <- length(countdata)

rownames(countdata) <- countdata[,1]

countdata <- countdata[,2:len]

**## doing log10 for all values. All number 0 should be changed to 0.001 before that.**

countdata[countdata == 0] <- 0.001

lcountdata <- log(countdata)/log(10)

**# Heatmap**

pdf("./pheatmap\_Diff\_Exp.pdf")

pheatmap(lcountdata)

## 6.4 Output files

The result is a heatmap with upgrade or downgrade information, and clustering for both genes and samples. For more information, you could visit:

<https://r-charts.com/correlation/pheatmap/>

Graphical user interface, bar chart

Description automatically generated

*-End-*