# TABLE OF CONTENTS



## 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/TaylorTianWei/RNA-seq_App_Notes

# 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:

```
checking for pkg-config... /usr/bin/pkg-config
checking for xml2-config... /opt/python/3.6.5/bin/xml2-config
USE_XML2 = yes
SED_EXTENDED_ARG: -E
Minor 9, Patch 8 for 2.9.8
Located parser file -I/opt/python/3.6.5/include/libxml2 -I/opt/python/3.6.5/include/
Checking for 1.8:  -I/opt/python/3.6.5/include/libxml2 -I/opt/python/3.6.5/include
Using libxml2.*
checking for gzopen in -lz... yes
checking for xmlParseFile in -lxml2... no
checking for xmlParseFile in -lxml... no
configure: error: "libxml not found"
ERROR: configuration failed for package 'XML'
* removing '/home/juri.kuusik/R/x86_64-pc-linux-gnu-library/3.6/XML'
Warning in install.packages :
  installation of package 'XML' had non-zero exit status

The downloaded source packages are in
    '/tmp/RtmpCCCTwU/downloaded_packages'
```

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:

```
##          ensgene SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516
## 1 ENSG00000000003        723        486        904        445       1170
## 2 ENSG00000000005          0          0          0          0          0
## 3 ENSG00000000419        467        523        616        371        582
## 4 ENSG00000000457        347        258        364        237        318
## 5 ENSG00000000460         96         81         73         66        118
## 6 ENSG00000000938          0          0          1          0          2
##   SRR1039517 SRR1039520 SRR1039521
## 1       1097        806        604
## 2          0          0          0
## 3        781        417        509
## 4        447        330        324
## 5         94        102         74
## 6          0          0          0
```

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:

```
gene_id transcript_id(s)        length  expected_count  FPKM
1       NM_130786       1766.00 9.00    1.79
100     NM_000022,NM_001322050,NM_001322051     1458.00 3.00    0.73
1000    NM_001308176,NM_001792  3859.00 5.00    0.45
10000   NM_001206729,NM_005465,NM_181690        5803.36 94.77   5.60
10001   NM_001284209,NM_001284210,NM_001284211,NM_005466        2386.00 70.11   10.22
10003   NM_001300930,NM_005467  3097.00 1.00    0.11
100037417       NM_001084393    1641.00 11.70   2.51
10004   NM_005468       2703.00 8.00    1.03
10005   NM_005469       1188.00 31.06   9.32
```

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

```
gene_id expected_count
1       9
100     3
1000    5
10000   94
10001   70
10003   1
100037417       11
10004   8
10005   31
```

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

| gene_id | C1 | C2 | T1 | T2 | T3 | |
|---|---|---|---|---|---|---|
| 1 | 9 | 5 | 40 | 33 | 47 | |
| 10 | 0 | 0 | 0 | 0 | 1 | |
| 100 | 3 | 37 | 701 | 595 | 509 | |
| 1000 | 5 | 8 | 157 | 107 | 109 | |
| 10000 | 94 | 76 | 137 | 119 | 112 | |
| 10001 | 70 | 73 | 92 | 81 | 78 | |
| 10002 | 0 | 0 | 4 | 3 | 1 | |
| 10003 | 1 | 8 | 6 | 10 | 8 | |
| 100037417 | | 11 | 0 | 22 | 13 | 7 |
| 10004 | 8 | 1 | 3 | 3 | 4 | |
| 100049587 | | 0 | 2 | 11 | 4 | 10 |
| 10005 | 31 | 5 | 67 | 70 | 48 | |

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/TaylorTianWei/RNA-seq_App_Notes/

## 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)
len <- length(countdata)
rownames(countdata) <- countdata[,1]
countdata <- countdata[,2:len]

type <- c("Control","Control","Treat","Treat","Treat")
coldata <- data.frame(type)
head(countdata)

dds <- DESeqDataSetFromMatrix(countData=countdata, colData=coldata, design = ~ type)
dds <- DESeq(dds)
result <- results(dds)
write.table(result, file="./Ctrl-VS-Case.deseq2.output", quote=FALSE, sep="\t")
~
~
```

input file

output file

## 2.4 Output file of DESeq2

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

| | baseMean | log2FoldChange | lfcSE | stat | pvalue | padj |
|---|---|---|---|---|---|---|
| 1 | 24.7112919981258 | 2.16899282835183 | 0.659867728978982 | 3.28701152230604 | 0.00101256690575275 | 0.0047223810753624 |
| 10 | 0.19705257512115 | 0.580987595763125 | 4.56207316163209 | 0.127351661224844 | 0.89866207338732 | NA |
| 100 | 329.156994520089 | 4.38880305110266 | 0.799939004984075 | 5.48642211938401 | 4.10156126068942e-08 | 5.34006535075392e-07 |
| 1000 | 68.5923049551391 | 3.84651448920949 | 0.558016435942969 | 6.89319210232477 | 5.45541232292296e-12 | 1.28141930984239e-10 |
| 10000 | 104.480081823531 | 0.151800658824309 | 0.328648838494098 | 0.461893185199969 | 0.644157921271074 | 0.753620784002288 |
| 10001 | 78.0644317924107 | -0.167039386348705 | 0.375319687821052 | -0.445058950460248 | 0.656277153008548 | 0.763176399076437 |
| 10002 | 1.36969920339644 | 3.42768243843498 | 2.49747290770918 | 1.37246030892044 | 0.169920186762899 | NA |
| 10003 | 6.6678642842486 | 0.374247925339799 | 1.19314355245425 | 0.313665463447365 | 0.753775134041687 | 0.837942400002152 |
| 100037417 | 9.43132636287444 | 1.04704668846867 | 1.25131504379923 | 0.836757053035686 | 0.402729142364558 | 0.546184276494247 |
| 10004 | 3.71634623981986 | -0.698175686441309 | 1.35228464341225 | -0.516293437067796 | 0.605649479287338 | 0.722698929183361 |
| 100049587 | 4.91488616535525 | 2.58974083982722 | 1.52561838481595 | 1.6975023804132 | 0.0896017214451461 | 0.180186339090737 |
| 10005 | 40.5129287557805 | 1.48742536561726 | 0.696847962132747 | 2.13450486540119 | 0.0328014758074227 | 0.0831718183867384 |
| 10006 | 252.032553587835 | -2.03735201011255 | 0.352946076590259 | -5.77241721963592 | 7.81423367031242e-09 | 1.14759864463916e-07 |
| 10007 | 124.475952572451 | 1.55797047483934 | 0.440560855997712 | 3.53633431937818 | 0.000405720899898955 | 0.00215407544357138 |
| 10008 | 4.49732853017748 | 2.62388369177243 | 1.52113557124977 | 1.72495058386981 | 0.0845363784308949 | 0.17261179101195 |
| 10009 | 116.385937183786 | -0.184934343723244 | 0.335036117081966 | -0.551983306557962 | 0.580959790538962 | 0.702427996308536 |
| 1001 | 182.586023393159 | -2.30077998653188 | 0.408172346331803 | -5.63678555690685 | 1.73253635964874e-08 | 2.39655007806052e-07 |
| 10010 | 125.222326084193 | -1.80313841886667 | 0.378406171129996 | -4.76508724337696 | 1.88771952696203e-06 | 1.78890307400578e-05 |
| 100101267 | 439.137771582558 | 0.0114415529255182 | 0.412998065234032 | 0.0277036477617266 | 0.977898514328426 | 0.986519692109488 |
| 100101467 | 31.066045740679 | 0.311600068661022 | 0.574365328866105 | 0.542511974523555 | 0.587465860586407 | 0.707772549515985 |

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

```
mcols(result, use.names=TRUE)

## DataFrame with 6 rows and 2 columns
##                         type                                        description
##                  <character>                                        <character>
## baseMean        intermediate     mean of normalized counts for all samples
## log2FoldChange       results log2 fold change (MLE): type Treat vs Control
## lfcSE                results          standard error: type Treat vs Control
## stat                 results          Wald statistic: type Treat vs Control
## pvalue               results       Wald test p-value: type Treat vs Control
## padj                 results                           BH adjusted p-values
```

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:

```
if (!require("BiocManager", quietly = TRUE))
    install.packages("BiocManager")

BiocManager::install("edgeR")
```

## 3.2 Preparing count matrix

Since there is no replicated samples being sequenced, the matrix should contains 3 columns, in this case, we use C1 and T1 data prepared for the 2.2, which is:

| gene_id | C1 | T1 | |
|---|---|---|---|
| 1 | 9 | 40 | |
| 10 | 0 | 0 | |
| 100 | 3 | 701 | |
| 1000 | 5 | 157 | |
| 10000 | 94 | 137 | |
| 10001 | 70 | 92 | |
| 10002 | 0 | 4 | |
| 10003 | 1 | 6 | |
| 100037417 | | 11 | 22 |
| 10004 | 8 | 3 | |
| 100049587 | | 0 | 11 |
| 10005 | 31 | 67 | |

## 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)
len <- length(countdata)
rownames(countdata) <- countdata[,1]
countdata <- countdata[,2:len]

## make DGEList
type <- c("Control","Treat")
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, p-value set to 0.05
y_bcv <- y
bcv <- 0.2
diffExp <- exactTest(y_bcv,dispersion = bcv ^ 2)

## extract result
df = topTags(diffExp, n =9000)
write.table(df,file="./Different_Expressed_genes.txt")
~
```

input

output

## 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.

| | logFC | logCPM | PValue | FDR |
|---|---|---|---|---|
| "3849" | -18.6688619362526 | 13.0634170832686 | 3.09989616095101e-79 | 4.33985462533141e-75 |
| "3852" | -18.0315137089068 | 12.4260333013456 | 1.92284587363421e-74 | 1.34599211154395e-70 |
| "3861" | -17.7952058892431 | 12.1897077910574 | 1.14943597548854e-72 | 5.36403455227984e-69 |
| "3848" | -14.6818442552855 | 12.0669902123252 | 9.60718417601436e-72 | 3.36251446160503e-68 |
| "3891" | -16.7430353709209 | 11.1374121135358 | 9.22869868602019e-65 | 2.58403563208565e-61 |
| "388698" | -12.6909238440408 | 10.9823456320744 | 3.48535929805857e-62 | 8.13250502880334e-59 |
| "7062" | -11.3310991703902 | 11.1417988298055 | 2.01540456679793e-60 | 3.87719749727654e-57 |
| "2312" | -11.0350180485172 | 11.4195426095224 | 2.33864058883421e-60 | 3.87719749727654e-57 |
| "147183" | -16.1523643975625 | 10.5466197735126 | 2.49248410539206e-60 | 3.87719749727654e-57 |
| "117159" | -15.8232342062935 | 10.2173976316427 | 7.3054517449136e-58 | 1.0227632442879e-54 |
| "3881" | -15.8171221843233 | 10.2112837016003 | 8.10960269106514e-58 | 1.03213125159011e-54 |
| "112802" | -15.7789288294985 | 10.1730782440101 | 1.56492682829801e-57 | 1.82574796634768e-54 |
| "1828" | -10.7092563297479 | 10.520122424645 | 9.26221712371498e-56 | 9.97469536400074e-53 |
| "810" | -15.1221941414178 | 9.51607886478937 | 1.28194460926769e-52 | 1.28194460926769e-49 |
| "3854" | -11.2916576360042 | 9.58298169283706 | 1.04453988691553e-51 | 9.74903894454496e-49 |
| "342574" | -14.9605448211249 | 9.35434469600566 | 2.06292268267923e-51 | 1.80505734734432e-48 |
| "653499" | -11.8719894640602 | 9.25667754466055 | 1.09788538482592e-50 | 9.04140905150754e-48 |
| "1823" | -14.7643674737995 | 9.15805152463511 | 5.99735419635534e-50 | 4.66460881938749e-47 |
| "337880" | -14.7609425096737 | 9.15462440372241 | 6.35339176479048e-50 | 4.68144656352983e-47 |
| "1825" | -10.0066523544587 | 9.56140936432377 | 1.89312471293119e-49 | 1.32518729905184e-46 |
| "374897" | -10.5304327070438 | 9.37435794027096 | 5.04447430546983e-49 | 3.36298287031322e-46 |
| "574414" | -14.6281391908835 | 9.02173364488769 | 6.21456576082955e-49 | 3.95472366598244e-46 |
| "3886" | -14.5136063288693 | 8.90711909858271 | 4.39569958772395e-48 | 2.67564322731023e-45 |
| "121391" | -14.5013434357391 | 8.89484709723909 | 5.44825466355886e-48 | 3.17814855374267e-45 |
| "4014" | -14.4168136793581 | 8.81025256963414 | 2.31507240560087e-47 | 1.29644054713649e-44 |
| "5317" | -8.53617035693224 | 10.6918213377524 | 4.74061085002841e-47 | 2.55263661155376e-44 |
| "1308" | -9.15638431009065 | 9.69200640546692 | 8.29081155428936e-47 | 4.29893932444634e-44 |
| "6274" | -14.3296658376615 | 8.72303418502568 | 1.02423318629218e-46 | 5.12116593146089e-44 |
| "100423062" | 10.2723638075661 | 8.98415748100571 | 3.96541310020261e-45 | 1.91433735871185e-42 |

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

```
if (!require("BiocManager", quietly = TRUE))
    install.packages("BiocManager")

BiocManager::install("Mfuzz")
```

## 4.2 Preparing count matrix

In this case, we prepared samples from 5 time point. The first part of the file should be like:

| gene_id | 1day | 2day | 3day | 4day | 5day | |
|---|---|---|---|---|---|---|
| 1 | 9 | 5 | 40 | 33 | 47 | |
| 10 | 0 | 0 | 0 | 0 | 1 | |
| 100 | 3 | 37 | 701 | 595 | 509 | |
| 1000 | 5 | 8 | 157 | 107 | 109 | |
| 10000 | 94 | 76 | 137 | 119 | 112 | |
| 10001 | 70 | 73 | 92 | 81 | 78 | |
| 10002 | 0 | 0 | 4 | 3 | 1 | |
| 10003 | 1 | 8 | 6 | 10 | 8 | |
| 100037417 | | 11 | 0 | 22 | 13 | 7 |
| 10004 | 8 | 1 | 3 | 3 | 4 | |
| 100049587 | | 0 | 2 | 11 | 4 | 10 |
| 10005 | 31 | 5 | 67 | 70 | 48 | |
| 10006 | 352 | 430 | 162 | 113 | 109 | |

## 4.3 Run Mfuzz

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

```
library("Mfuzz")

## input matrix
data<-table2eset("./all exp.txt")

## 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)
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)
mfuzz.plot2(data.s,cl=cl,mfrow=c(1,1),min.mem=0.75,time.labels=c("day1","day2","day3","day4","day5"),x11 = FALSE)
cluster<-cl$cluster
membership<-cl$membership
expStandard<-exprs(data.s)

## output
write.table(cluster,file="timeSeries.cluster")
write.table(expStandard,file="timeSeries.expStandard")
dev.off()
~
```

Input file

output cluster figure

another plot figure, one figure per page

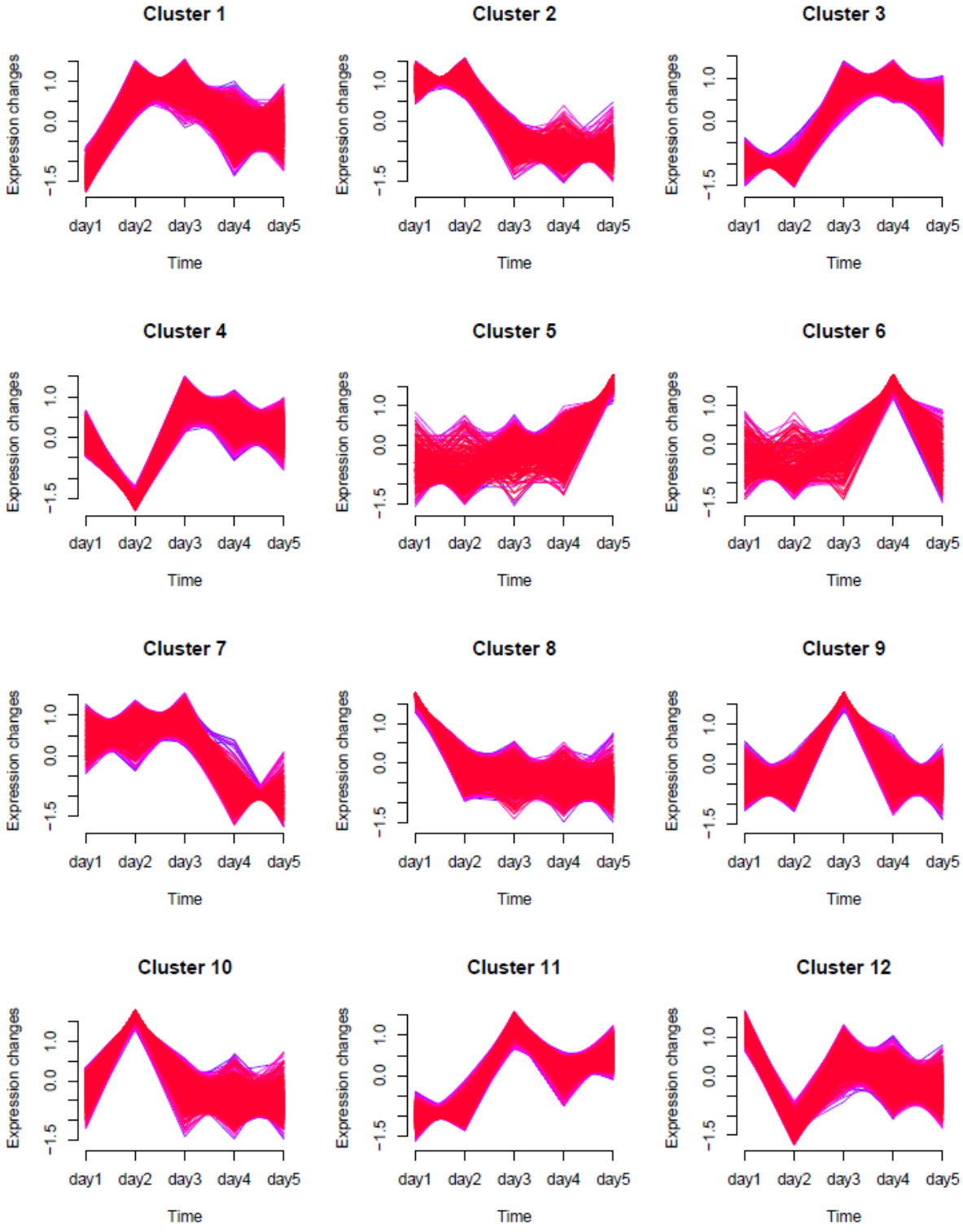
output cluster information

output expression across samples

## 4.4 Output files for Mfuzz

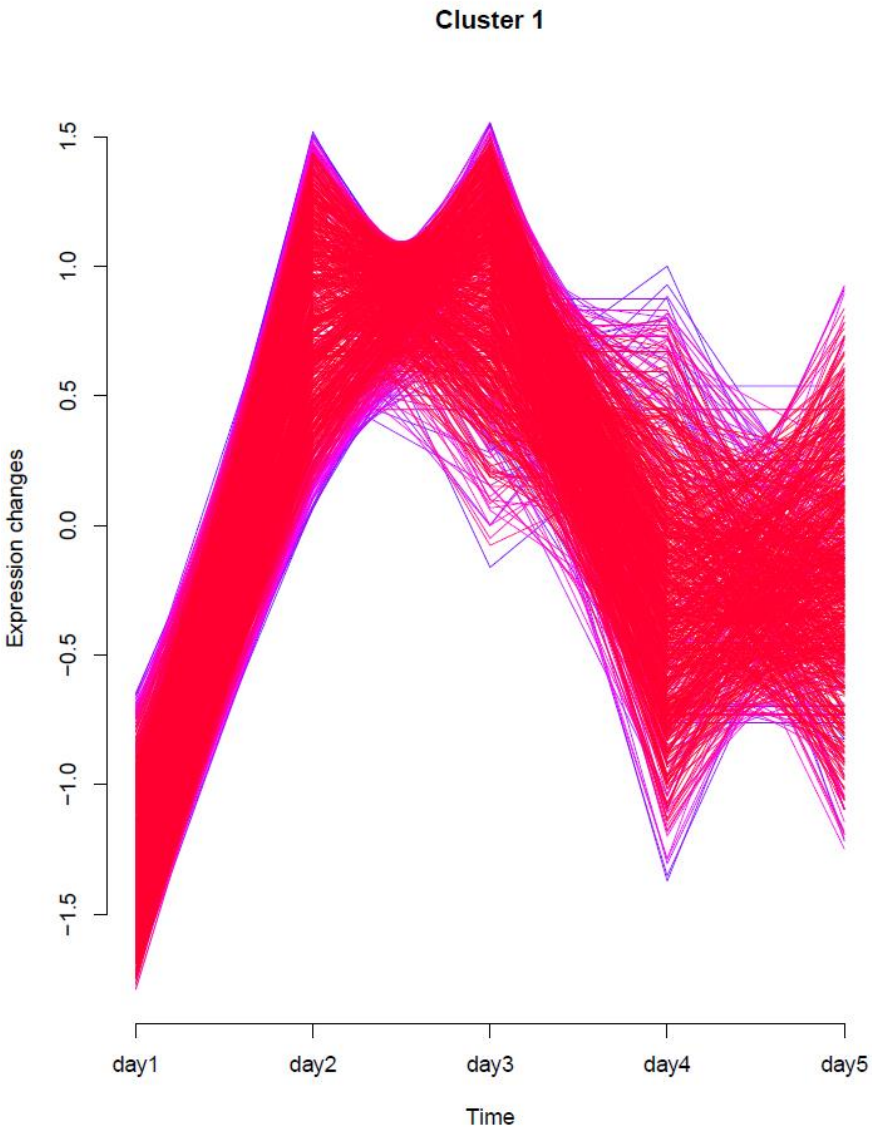
### 4.4.1 output cluster figure

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:



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

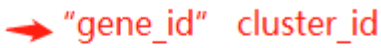
The second figure gives amplified clusters information, for example:




### 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:

```
"x"
"1" 10
"10" 3
"100" 10
"1000" 11
"10000" 5
"10001" 11
"10002" 10
"10003" 1
"100037417" 5
"10004" 8
"100049587" 11
"10005" 5
"10006" 12
"10007" 10
"10008" 10
"10009" 2
"1001" 12
"10010" 12
"100101267" 6
"100101467" 5
"100101629" 2
"10011" 5
```




### 4.4.3 output expression

This file gives more information about the expression pattern:

```
"1day" "2day" "3day" "4day" "5day"
"1" -0.947130135007187 -1.15996836759307 0.702366167533419 0.329899260508121 1.07483307455872
"10" -0.447213595499958 -0.447213595499958 -0.447213595499958 -0.447213595499958 1.78885438199983
"100" -1.12273107983722 -1.0184336571201 1.0184336571201 0.6932711039432 0.429459975894018
"1000" -1.06834662547965 -1.02395549145695 1.18080416500383 0.440951931292156 0.470546020640623
"10000" -0.58027533217633 -1.34828680123324 1.25441873279295 0.486407263736042 0.187736136880578
"10001" -1.03208500124441 -0.680237841729269 1.54812750186661 0.258021250311103 -0.0938259092040369
"10002" -0.880771012101089 -0.880771012101089 1.32115651815163 0.770674635588452 -0.330289129537908
"10003" -1.63022302727083 0.407555756817708 -0.174666752921875 0.98977826655729 0.407555756817708
"100037417" 0.0494997950403355 -1.31174456856889 1.41074415864956 0.296998770242013 -0.445498155363019
"10004" 1.62260155950114 -1.08173437300076 -0.309066963714502 -0.309066963714502 0.0772667409286256
"100049587" -1.10689208328256 -0.69693205243717 1.1478880863671 -0.286972021591776 0.942908070944406
"10005" -0.489325381922889 -1.45314810389222 0.845198386957717 0.956408701031101 0.140866397826286
"10006" 0.801921444167337 1.32843552367114 -0.480612852059885 -0.81137169687638 -0.838372418902216
"10007" -0.778926085845456 -1.32621075400485 0.684742212720352 0.990203422855825 0.430191204274125
"10008" -0.727606875108999 -1.21267812518166 0.727606875108999 1.21267812518166 0
"10009" -0.334312287961949 -0.598243041616119 1.77713374127141 -0.334312287961949 -0.510266123731395
"1001" 0.62447899565441 1.45386516175792 -0.505762936584691 -0.741568807339612 -0.83101241348803
"10010" 0.781766565862976 1.35568249715127 -0.583222676119998 -0.738335089981699 -0.81589129691255
"100101267" -1.57739412513299 0.101469797523175 1.11616777275493 0.488899933520753 -0.129143378665859
"100101467" 0.221464041157669 -1.47642694105112 1.32878424694601 0 -0.0738213470525562
```

*-End-*