Gene differential expression analysis

Chunhui

7/8/2019

Contents

PART 1: Preparation: Packages & options

```
library(BiocManager)
```

```
## Warning: package 'BiocManager' was built under R version 3.5.3
```

```
# if you have install DESeq2, uncomment the followiing line
# BiocManager::install("DESeq2")
library(DESeq2)
```

```
## Warning: package 'DESeq2' was built under R version 3.5.2

## Loading required package: S4Vectors

## Loading required package: stats4

## Loading required package: BiocGenerics

## Loading required package: parallel

##
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB

## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind,
##     colMeans, colnames, colSums, dirname, do.call, duplicated,
##     eval, evalq, Filter, Find, get, grep, grepl, intersect,
##     is.unsorted, lapply, lengths, Map, mapply, match, mget, order,
##     paste, pmax, pmax.int, pmin, pmin.int, Position, rank, rbind,
##     Reduce, rowMeans, rownames, rowSums, sapply, setdiff, sort,
##     table, tapply, union, unique, unsplit, which, which.max,
```

```
##     which.min
##
## Attaching package: 'S4Vectors'
## The following object is masked from 'package:base':
##
##     expand.grid
## Loading required package: IRanges
##
## Attaching package: 'IRanges'
## The following object is masked from 'package:grDevices':
##
##     windows
## Loading required package: GenomicRanges
## Loading required package: GenomeInfoDb
## Warning: package 'GenomeInfoDb' was built under R version 3.5.2
## Loading required package: SummarizedExperiment
## Loading required package: Biobase
## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.
## Loading required package: DelayedArray
## Loading required package: matrixStats
## Warning: package 'matrixStats' was built under R version 3.5.3
##
## Attaching package: 'matrixStats'
## The following objects are masked from 'package:Biobase':
##
##     anyMissing, rowMedians
## Loading required package: BiocParallel
## Warning: package 'BiocParallel' was built under R version 3.5.2
##
## Attaching package: 'DelayedArray'
## The following objects are masked from 'package:matrixStats':
##
##     colMaxs, colMins, colRanges, rowMaxs, rowMins, rowRanges
## The following objects are masked from 'package:base':
##
##     aperm, apply
library(tidyverse)
## Warning: package 'tidyverse' was built under R version 3.5.3
```

```
## -- Attaching packages ------------------------------------------------------------------------ tidy
## v ggplot2 3.1.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.2
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
## Warning: package 'ggplot2' was built under R version 3.5.3
## Warning: package 'tibble' was built under R version 3.5.3
## Warning: package 'tidyr' was built under R version 3.5.3
## Warning: package 'readr' was built under R version 3.5.3
## Warning: package 'purrr' was built under R version 3.5.3
## Warning: package 'dplyr' was built under R version 3.5.3
## Warning: package 'stringr' was built under R version 3.5.3
## Warning: package 'forcats' was built under R version 3.5.3
## -- Conflicts --------------------------------------------------------------------------- tidyverse_
## x dplyr::collapse()   masks IRanges::collapse()
## x dplyr::combine()    masks Biobase::combine(), BiocGenerics::combine()
## x dplyr::count()      masks matrixStats::count()
## x dplyr::desc()       masks IRanges::desc()
## x tidyr::expand()     masks S4Vectors::expand()
## x dplyr::filter()     masks stats::filter()
## x dplyr::first()      masks S4Vectors::first()
## x dplyr::lag()        masks stats::lag()
## x ggplot2::Position() masks BiocGenerics::Position(), base::Position()
## x purrr::reduce()     masks GenomicRanges::reduce(), IRanges::reduce()
## x dplyr::rename()     masks S4Vectors::rename()
## x purrr::simplify()   masks DelayedArray::simplify()
## x dplyr::slice()      masks IRanges::slice()
```

```
library(biomaRt)
library(dplyr)
# run next line if you need to get some help from DESeq2
# browseVignettes("DESeq2")
```

PART2: Data preparation

2.1 reading dataset from csv file

```
head(table <- read.csv('kinetics_All_HD.csv', stringsAsFactors = F))
```

```
##           Geneid1      Geneid HD1_Blood HD2_Blood HD6_Blood HD1_1H HD2_1H
## 1 ENSG00000223972     DDX11L1         0         0         0      0      0
## 2 ENSG00000227232      WASH7P        56        51        57    105     85
## 3 ENSG00000278267   MIR6859-1         8         9        12      1      6
## 4 ENSG00000243485 MIR1302-2HG         0         0         0      0      0
## 5 ENSG00000284332   MIR1302-2         0         0         0      0      0
## 6 ENSG00000237613     FAM138A         0         0         0      0      0
##   HD6_1H HD1_2H HD2_2H HD6_2H HD1_4H HD2_4H HD6_4H HD1_6H HD2_6H HD6_6H
## 1      0      0      0      0      1      0      0      0      0      0
## 2     31     64     30     18     41     45     17     40     48    118
```

```
## 3      1      2      0      3      0      0      6      1      0     28
## 4      0      0      0      0      0      0      0      0      0      0
## 5      0      0      0      0      0      0      0      0      0      0
## 6      0      0      0      0      0      0      0      0      0      0
```

The first column is the `ensemble gene id`, the second column is a mixture of `ensemble gene id` and `external gene name`. The third column to the last column are gene expression level, the prefix indicates which patient we get the sample and the suffix indicates which time point we get the sample. In this example, there are total 3 patients and each patient has 4 time samples, 12 total.

2.2 Get external

For further analysis, we need to get `external gene name` from `ensenble gene id` as possible as we can. By nature, `external gene name` and `ensenble gene id` are not one-to-one relationship, e.g. multiple `external gene names` could correspond to one `external gene name`

To do this, we need library `BiomaRt`.

List available `Mart` by use `listMarts`, then create ensembl objection by using `useMart`. Use `listDatasets` with ensembl objection as parameter to get available gene dataset

Create Mart object by using `useDataset` with corresponding `ensemble` and `gene dataset`. `listFilters` would list all the available filters

For more information about `biomaRt`, run **browseVignettes('biomaRt')**

use `getBM` with correct filter and attributes to get corresponding `external gene name` from `ensemble gene id`

This procedure could take a long time depending how large your dataset and you network speed

```
G_list_2 <- getBM(filters= "ensembl_gene_id",
                  attributes= c("ensembl_gene_id", 'external_gene_name'),
                  values=table$Geneid1,
                  mart= mart)
head(G_list_2)
```

```
##   ensembl_gene_id external_gene_name
## 1 ENSG00000007923            DNAJC11
## 2 ENSG00000008128              CDK11A
## 3 ENSG00000008130                NADK
## 4 ENSG00000009724               MASP2
## 5 ENSG00000011021               CLCN6
## 6 ENSG00000028137           TNFRSF1B
```

rebuild dataset

left join original dataset `table` with `G_list`, for those `ensembl gene name` don't have `external gene name` , using their `ensemble gene name` instead.

```
table_merge <- sqldf::sqldf('select *
             from "table"
             left join G_list_2 on "table".Geneid1 = G_list_2.ensembl_gene_id')
```

```
## Warning: package 'RSQLite' was built under R version 3.5.2
```

```
table_reordered <- table_merge %>%
  as_tibble() %>%
   dplyr::select(Geneid1, external_gene_name, everything())
```

```
#remove useless column
table_reordered <- table_reordered[, c(-3, -19)]
table_reordered$external_gene_name <-
  ifelse(is.na(table_reordered$external_gene_name),table_reordered$Geneid1,
         table_reordered$external_gene_name)

head(table_reordered)
```

```
## # A tibble: 6 x 17
##   Geneid1 external_gene_n~ HD1_Blood HD2_Blood HD6_Blood HD1_1H HD2_1H
##   <chr>   <chr>                <int>     <int>     <int>  <int>  <int>
## 1 ENSG00~ DDX11L1                  0         0         0      0      0
## 2 ENSG00~ WASH7P                  56        51        57    105     85
## 3 ENSG00~ MIR6859-1                8         9        12      1      6
## 4 ENSG00~ MIR1302-2HG              0         0         0      0      0
## 5 ENSG00~ MIR1302-2                0         0         0      0      0
## 6 ENSG00~ FAM138A                  0         0         0      0      0
## # ... with 10 more variables: HD6_1H <int>, HD1_2H <int>, HD2_2H <int>,
## #   HD6_2H <int>, HD1_4H <int>, HD2_4H <int>, HD6_4H <int>, HD1_6H <int>,
## #   HD2_6H <int>, HD6_6H <int>
```

keep only `Geneid` column

```
table_geneid <- table_reordered[,-1]
colnames(table_geneid)[1] = 'Geneid'
# omit any row if has missing value
# Caution! this may not the best method
head(table_geneid)
```

```
## # A tibble: 6 x 16
##   Geneid HD1_Blood HD2_Blood HD6_Blood HD1_1H HD2_1H HD6_1H HD1_2H HD2_2H
##   <chr>      <int>     <int>     <int>  <int>  <int>  <int>  <int>  <int>
## 1 DDX11~         0         0         0      0      0      0      0      0
## 2 WASH7P        56        51        57    105     85     31     64     30
## 3 MIR68~         8         9        12      1      6      1      2      0
## 4 MIR13~         0         0         0      0      0      0      0      0
## 5 MIR13~         0         0         0      0      0      0      0      0
## 6 FAM13~         0         0         0      0      0      0      0      0
## # ... with 7 more variables: HD6_2H <int>, HD1_4H <int>, HD2_4H <int>,
## #   HD6_4H <int>, HD1_6H <int>, HD2_6H <int>, HD6_6H <int>
```

resolve duplicates

Since the transformation from `ensemble gene id` and `external gene name` is not one-to-one, so `Geneid` is not unique, to transform dataset into a matrix with row name, `Geneid` must be `unique` ** for duplicates `Geneid`, take the mean of each column into one row**

```
#remove depublicate by geneid using sqlite
# base rule: average the read from the same gene
# caution! This may not applicable for bio-duplicate
table_nodup <-  sqldf::sqldf('
  select Geneid,
         avg(HD1_Blood) as HD1_Blood, avg(HD2_Blood) as HD2_Blood, avg(HD6_Blood) as HD6_Blood,
         avg(HD1_1H) as HD1_1H, avg(HD2_1H) as HD2_1H, avg(HD6_1H) as HD6_1H,
         avg(HD1_2H) as HD1_2H, avg(HD2_2H) as HD2_2H, avg(HD6_2H) as HD6_2H,
         avg(HD1_4H) as HD1_4H, avg(HD2_4H) as HD2_4H, avg(HD6_4H) as HD6_4H,
```

```
        avg(HD1_6H) as HD1_6H, avg(HD2_6H) as HD2_6H, avg(HD6_6H) as HD6_6H
  from table_geneid
  group by Geneid
  order by Geneid
  '
)
head(table_nodup)
```

```
##     Geneid HD1_Blood HD2_Blood HD6_Blood HD1_1H HD2_1H HD6_1H HD1_2H
## 1     A1BG         4         0         0      0      0      0      0
## 2 A1BG-AS1         2         2         4      2      2      0      3
## 3     A1CF         0         0         0      0      0      0      0
## 4      A2M         9        17         4      4      5      5      6
## 5  A2M-AS1         0         0         0      0      2      0      2
## 6    A2ML1         2         1         0      2      0      0      0
##   HD2_2H HD6_2H HD1_4H HD2_4H HD6_4H HD1_6H HD2_6H HD6_6H
## 1      0      0      0      0      0      0      2      4
## 2     24      2      6     15      0     15     13      8
## 3      1      0      0      2      0      0      0      4
## 4      0      3      0      1      2      0      0      0
## 5      2      0     11      4      0      2      7      8
## 6      2      0      2      8     17      2      0      2
```

Double check whether there is no duplicates

```
#check how many gene left in the processed dataset
length(table_nodup$Geneid)
```

```
## [1] 56682
```

```
#check whether there is duplicate now, if there is no duplicate, a True should be returen
all(!duplicated(table_nodup[,1]))
```

```
## [1] TRUE
```

PART 3: Using DESeq2

DESque2 is the updated version of DESque, You may have different type of orginal file, DESeq2 has different data preparation solutions for them. For more information,m run **browseVignettes('DESeq')**

3.1 Data preparation

make the matrix needed for next step

```
#make the matrix needed for next step
#use the geneid as row.names
table_rownames <- data.frame(table_nodup[,-1], row.names=table_nodup[,1])
count_maxtrix <- as.matrix(table_rownames)
# make the mode of matrix integer otherwise it will be number
mode(count_maxtrix) <- 'integer'
head(count_maxtrix)
```

```
##          HD1_Blood HD2_Blood HD6_Blood HD1_1H HD2_1H HD6_1H HD1_2H HD2_2H
## A1BG             4         0         0      0      0      0      0      0
## A1BG-AS1         2         2         4      2      2      0      3     24
## A1CF             0         0         0      0      0      0      0      1
## A2M              9        17         4      4      5      5      6      0
```

```
## A2M-AS1          0         0         0      0      2      0      2      2
## A2ML1            2         1         0      2      0      0      0      2
##          HD6_2H HD1_4H HD2_4H HD6_4H HD1_6H HD2_6H HD6_6H
## A1BG          0      0      0      0      0      2      4
## A1BG-AS1      2      6     15      0     15     13      8
## A1CF          0      0      2      0      0      0      4
## A2M           3      0      1      2      0      0      0
## A2M-AS1       0     11      4      0      2      7      8
## A2ML1         0      2      8     17      2      0      2
```

```
#create the coldata corresponding to the processed dataset
coldata <- data.frame(condition = c(rep('blood', 3),
                                    rep('1h', 3),
                                    rep('2h', 3),
                                    rep('4h', 3),
                                    rep('6h', 3)),
                      row.names = colnames(count_maxtrix)[1:15])
head(coldata)
```

```
##           condition
## HD1_Blood     blood
## HD2_Blood     blood
## HD6_Blood     blood
## HD1_1H           1h
## HD2_1H           1h
## HD6_1H           1h
```

It is absolutely critical that the columns of the count matrix and the rows of the column data (information about samples) are in the same order. DESeq2 will not make guesses as to which column of the count matrix belongs to which row of the column data, these must be provided to DESeq2 already in consistent order.

```
#check whether the name of row match col, a True should be returned
all(rownames(coldata) == colnames(count_maxtrix)[1:15])
```

```
## [1] TRUE
```

```
#create DESeqDataSet(dds) object, dds is a container for intermediate data
dds <- DESeqDataSetFromMatrix(countData = count_maxtrix[,1:15],
                              colData = coldata,
                              design = ~ condition)
```

3.2 pre-filtering

```
# pre-filtering
# by removing rows in which there are very few reads, we reduce the memory size of the dds data object,
#   and we increase the speed of the transformation and testing functions within DESeq2
keep <- rowSums(counts(dds)) >= 10
dds <- dds[keep,]

# set dds condition with all time points
# set 'blood' as base level, the default base level is determined by alphabet order
# other statement if also available for this purpose
# Caution!: the document only give example for factor with two levels, not sure about accuracy for
#   factor with more than 2 levels
dds$condition <- factor(dds$condition, levels = c("blood","1h",'2h', '4h', '6h'))
# drop levels that with no sample
```

```
dds$condition <- droplevels(dds$condition)
```

PArt 4: Defferential Expression Analysis

4.1 Create dds object

```
# DEseq analysis
dds <- DESeq(dds)
```

```
## estimating size factors
```

```
## estimating dispersions
```

```
## gene-wise dispersion estimates
```

```
## mean-dispersion relationship
```

```
## final dispersion estimates
```

```
## fitting model and testing
```

```
# for different condition you just change 6h to i.e. 1h, 2h, 4h
# may can be done by lapply() function, not sure how to set the func argument
# with different arguments
res_1h <- results(dds,contrast=c("condition","1h","blood"))
res_2h <- results(dds,contrast=c("condition","2h","blood"))
res_4h <- results(dds,contrast=c("condition","4h","blood"))
res_6h <- results(dds,contrast=c("condition","6h","blood"))
resultsNames(dds)
```

```
## [1] "Intercept"             "condition_1h_vs_blood" "condition_2h_vs_blood"
## [4] "condition_4h_vs_blood" "condition_6h_vs_blood"
```

4.2 Build Function for subset result

```
# this function is used to subset the neeeded gene from RESULT object
#function get those LFC greate than 1 and the adjusted P-value is less than 0.1
res_subgroup <- function(res, alpha=0.1, reg_LFC=1, reg_dir='all'){

  # res is an obj from DEseq2.result function
  # alpha gives the significant level for adjusted P-value
  # reg gives the regulation level change in log2 fold change in absolute value
  # reg_dir gives which regulation direction you want to subset you gene
    # three options: all -- up and down
    #                up  -- only up regulated
    #                down -- only down regulated
  res_sig_pos <- (res$padj < alpha)
  res_sig_pos[is.na(res_sig_pos)] <- F
  if(reg_dir == 'all'){
    res_LFC_pos <- (res$log2FoldChange > reg_LFC) | (res$log2FoldChange < -reg_LFC)
  }
  else if(reg_dir == 'up'){
          res_LFC_pos <- (res$log2FoldChange > reg_LFC)
      }
      else if(reg_dir == 'down'){
              res_LFC_pos <- (res$log2FoldChange < -reg_LFC)
           }
```

```
  return(res[res_LFC_pos & res_sig_pos,])

}
```

You can use `lapply` to subset a list of dataset togather

```
#get all 4 upregluated genes
res_list <- list(res_1h,res_2h,res_4h,res_6h)
res_list_up <- lapply(res_list, res_subgroup, reg_dir ='up')
#get all 4 downregluated genes
res_list_down <- lapply(res_list, res_subgroup, reg_dir ='down')
```

This code block is used for easily output xlxs file

```
# output corresponding excel file
library(xlsx)
```

```
## Warning: package 'xlsx' was built under R version 3.5.3
```

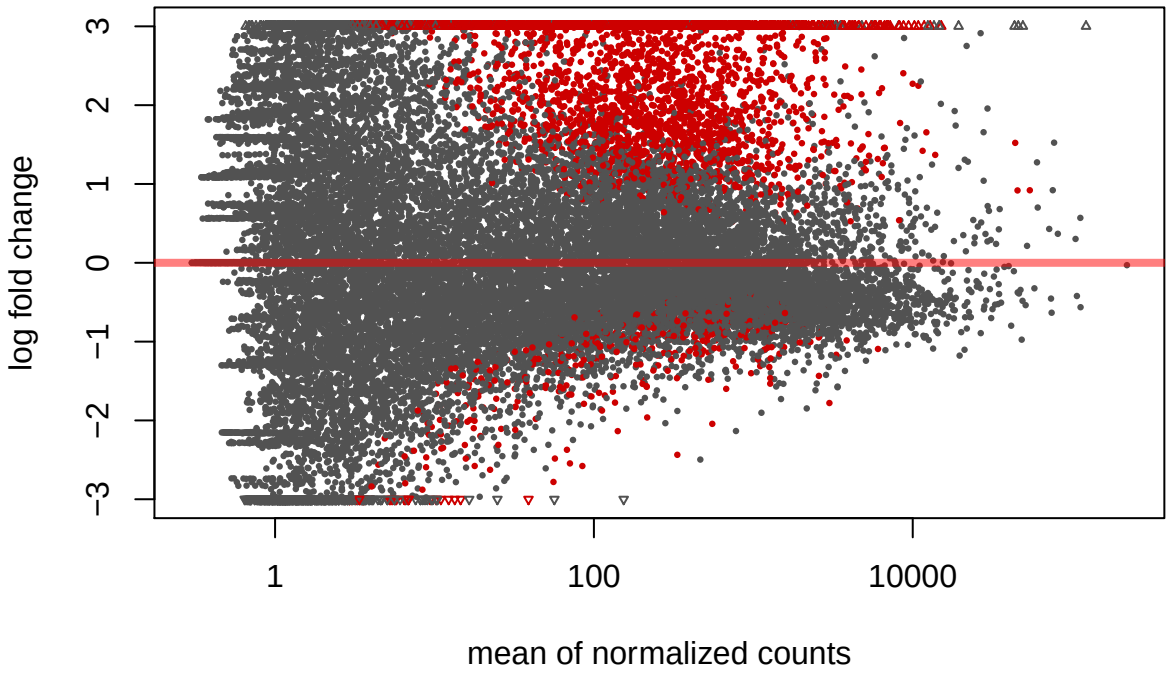
```
# genes that are upregulated
file_up <- paste(getwd(),'/', c("1", '2', '4', '6'), '_up.xlsx', sep="")
for(i in 1:4){
  write.xlsx2(res_list_up[i], file = file_up[i], row.names = T)
}
# genes that are downregulated
file_down <- paste(getwd(),'/', c("1", '2', '4', '6'), '_down.xlsx', sep="")
for(i in 1:4){
  write.xlsx2(res_list_down[i], file = file_down[i], row.names = T)
}
```

PART 5: Analysis and plot

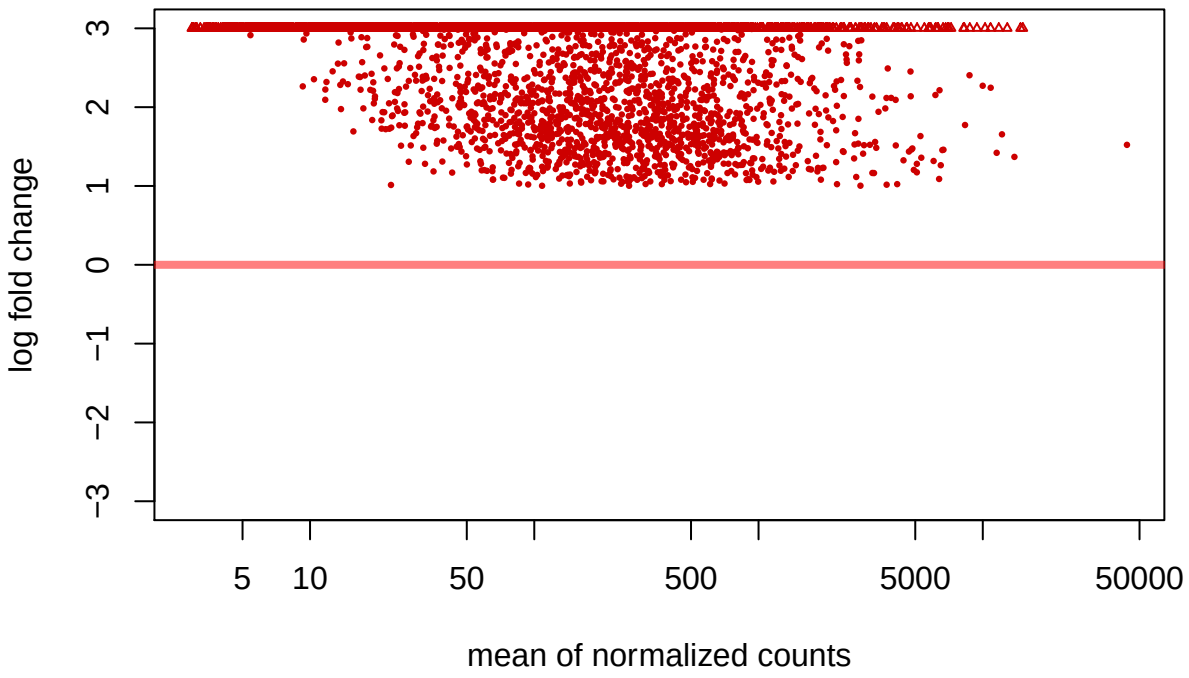
5.1 Plot MA-plot

use `plotMA` function with a `result` object as parameter

```
# plot the MAplot with the subset result
plotMA(res_1h, ylim=c(-3,3))
```



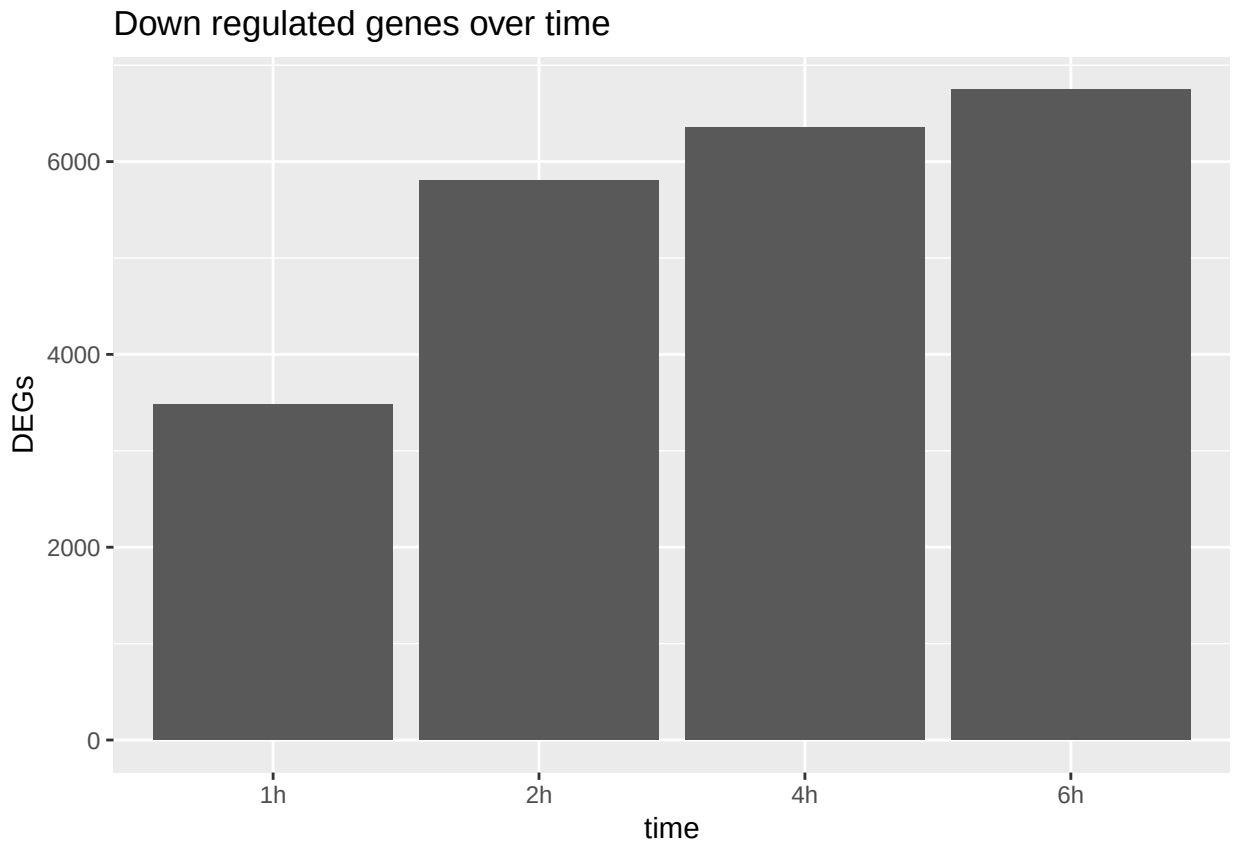

```
plotMA(res_list_up[[1]], ylim=c(-3,3))
```



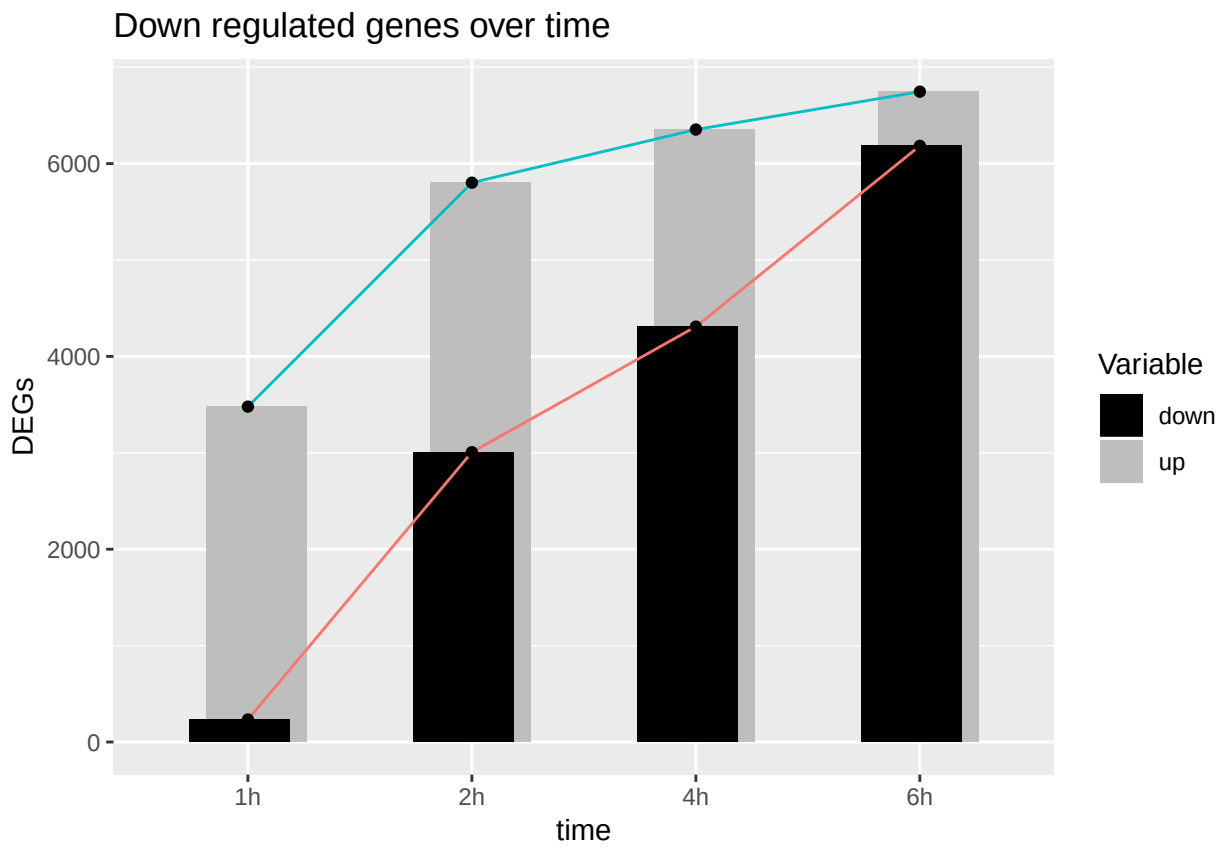
5.2

```
library(ggplot2)
library(tidyr)
dat <- NULL
dat$time <- as.factor(c('1h', '2h', '4h', '6h'))
dat$down <- sapply(res_list_down, nrow)
dat$up <- sapply(res_list_up, nrow)
dat <- as.data.frame(dat)

ggplot(data = dat, aes(time)) +
  geom_bar(aes(time , weight = down, fill = 'red'), show.legend = FALSE) +
  geom_bar(aes(time , weight = up), show.legend = FALSE) +
  labs(title = "Down regulated genes over time", x = "time", y = "DEGs")
```




```
ggplot(data = dat %>% gather(Variable, reg, -time),
       aes(x = time, y = reg, fill = Variable)) +
  geom_bar(stat = 'identity', position = position_dodge(width = 0.15)) +
  geom_line(aes(x = time, y = reg, group = Variable, color = Variable),
            stat="identity", show.legend = F) +
  geom_point(show.legend = F) +
  scale_fill_manual(values=c("black", "grey")) +
  labs(title = "Down regulated genes over time", x = "time", y = "DEGs")
```



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.