Lab12_inclass

Qianqian Tao

Here we will use the DESeq2 package for RNASeq analysis. The data for today's class come from a study of airway smooth muscle cells treated with dexamethasone, a synthetic glucocorticoid steroid with anti-inflammatory effects (Himes et al. 2014)

##Import their data

We need two things for this analysis:

-countData (counts for every transcript/gene in each experiment) -colData (metadata that describes the experimental setup)

```
countData <- read.csv("airway_scaledcounts.csv", row.names=1)
metaData <- read.csv("airway_metadata.csv", row.names=1)
head(countData)
```

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

```
nrow(countData)
```

```
[1] 38694
```

Q1. How many genes are in this dataset?

38694

Q2. How many 'control' cell lines do we have?

```
head(metaData)
```

```
               dex celltype     geo_id
SRR1039508 control   N61311 GSM1275862
SRR1039509 treated   N61311 GSM1275863
SRR1039512 control  N052611 GSM1275866
SRR1039513 treated  N052611 GSM1275867
SRR1039516 control  N080611 GSM1275870
SRR1039517 treated  N080611 GSM1275871
```

```
table(metaData$dex)
```

```
control treated
      4       4
```

There are 4 control cell lines.

Another way

```
sum(metaData$dex == "control")
```

```
[1] 4
```

(Is this gene significantly different between control and treated?)–T test, compare mean of control and treated

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.2     v readr     2.1.4
v forcats   1.0.0     v stringr   1.5.0
v ggplot2   3.4.2     v tibble    3.2.1
v lubridate 1.9.2     v tidyr     1.3.0
v purrr     1.0.1
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becor
```

```
countData %>%
  summarize()
```

```
data frame with 0 columns and 1 row
```

```
apply(countData, 2, mean)
```

```
SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516 SRR1039517 SRR1039520
  546.4878   501.0934   673.6225   405.3088   649.0404   822.4617   509.0292
SRR1039521
  565.7062
```

-Step 1. Calculate the mean of the control samples (i.e. columns in countData) Calculate the mean of the treated samples (i.e. columns in countData)

(a) We need to find all "control" samples

- Look in the metaData $dex column

Q3. How would you make the above code in either approach more robust?

```
control.inds <- metaData$dex=="control"
```

(b) Extract all the control columns from countData and call it `control.counts`

```
head(countData[,control.inds])
```

```
                SRR1039508 SRR1039512 SRR1039516 SRR1039520
ENSG00000000003        723        904       1170        806
ENSG00000000005          0          0          0          0
ENSG00000000419        467        616        582        417
ENSG00000000457        347        364        318        330
ENSG00000000460         96         73        118        102
ENSG00000000938          0          1          2          0
```

```
control.counts <- countData[,control.inds]
```

(c) Calculate the mean value accross the rows of `control.counts` i.e. calculate the mean count values for each gene in the control samples.

```
control.means <- rowMeans(control.counts)
head(control.means)
```

```
ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460
         900.75            0.00          520.50          339.75           97.25
ENSG00000000938
           0.75
```

Q4. Follow the same procedure for the treated samples (i.e. calculate the mean per gene across drug treated samples and assign to a labeled vector called treated.means)

-Step. 2 Do the same thing for treated sample

```
treated.ids <- metaData$dex=="treated"
treated.count <- countData[,treated.ids]
treated.means <- rowMeans(treated.count)
head(treated.means)
```

```
ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460
         658.00            0.00          546.00          316.50           78.75
ENSG00000000938
           0.00
```

We now have control and treated mean count values. For ease of book-keeping I will combine these vectors into a new data.frame called `meancounts`

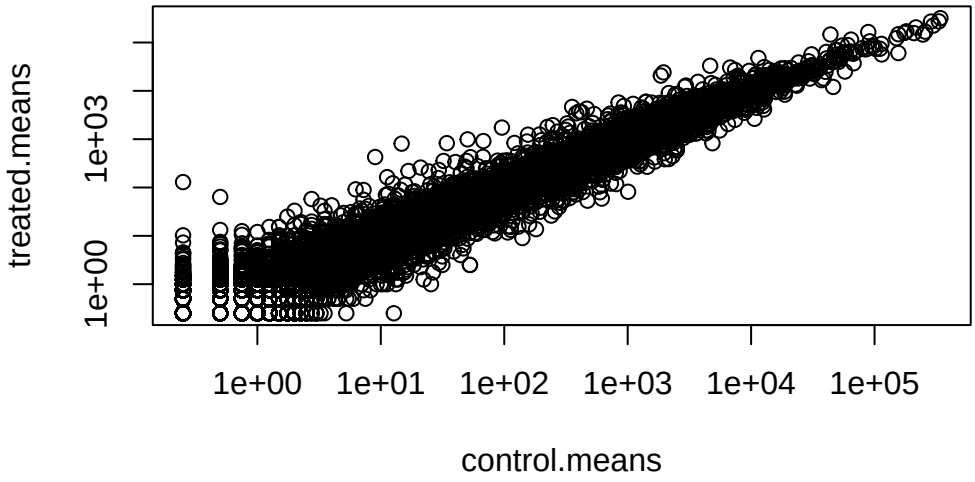
```
meancounts <- data.frame(control.means, treated.means)
```

Q5 (a). Create a scatter plot showing the mean of the treated samples against the mean of the control samples.

```
plot(meancounts, log="xy")
```

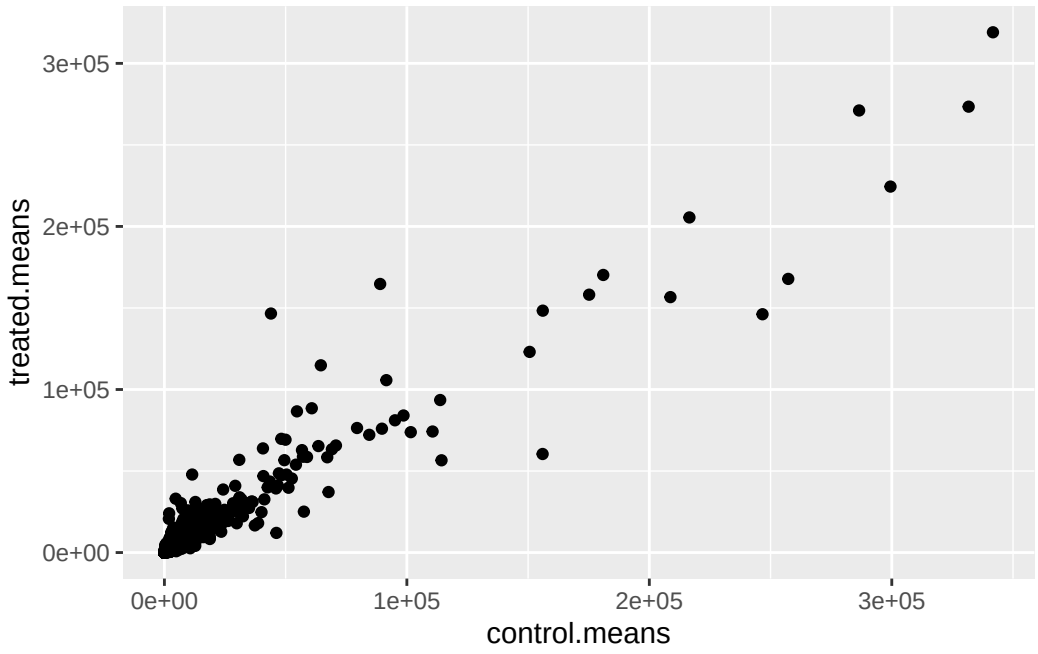
```
Warning in xy.coords(x, y, xlabel, ylabel, log): 15032 x values <= 0 omitted
from logarithmic plot
```

```
Warning in xy.coords(x, y, xlabel, ylabel, log): 15281 y values <= 0 omitted
from logarithmic plot
```



Q5 (b).You could also use the ggplot2 package to make this figure producing the plot below. What geom_?() function would you use for this plot?

```
library(ggplot2)
ggplot(meancounts, aes(control.means, treated.means))+
  geom_point()
```

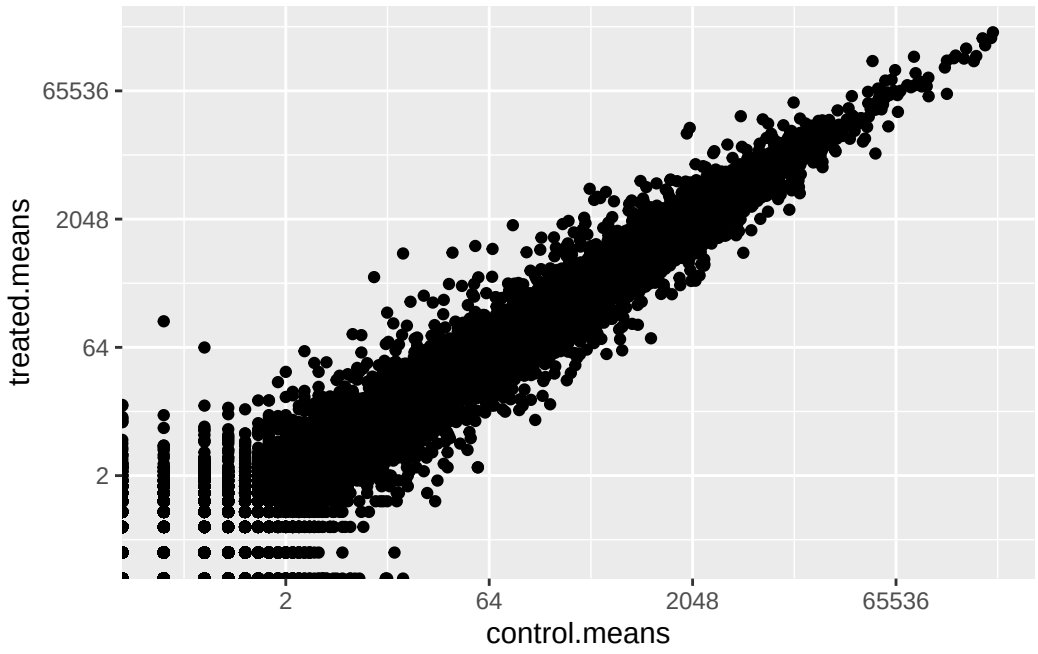


Q6. Try plotting both axes on a log scale. What is the argument to plot() that allows you to do this?

```
ggplot(meancounts, aes(control.means, treated.means))+
  geom_point()+
 scale_x_continuous(trans="log2")+
  scale_y_continuous(trans="log2")
```

```
Warning: Transformation introduced infinite values in continuous x-axis
```

```
Warning: Transformation introduced infinite values in continuous y-axis
```



We use log transforms for skewed data such as this and because we really care most about relative changes in magnitude.

We most often use log2 as our transform as the math is easier to interpret than log 10 or others.

If we have no change -i.e. some values in control and treated who have zero log 2 value

```
log2(10/10)
```

```
[1] 0
```

```
log2(20/10)
```

```
[1] 1
```

```
log2(10/20)
```

```
[1] -1
```

```
log2(40/10)
```

```
[1] 2
```

If I have double the amount i.e. 20 compared to 10 for example I will have a log2 fold-change of +1

```
meancounts$log2fc <- log2(meancounts$treated.means/meancounts$control.means)
head(meancounts)
```

```
                control.means treated.means      log2fc
ENSG00000000003        900.75        658.00 -0.45303916
ENSG00000000005          0.00          0.00         NaN
ENSG00000000419        520.50        546.00  0.06900279
ENSG00000000457        339.75        316.50 -0.10226805
ENSG00000000460         97.25         78.75 -0.30441833
ENSG00000000938          0.75          0.00        -Inf
```

> Q. How many genes are up regulated at the common threshold of +2 log2FC values? Do you trust the result?

```
sum(meancounts$log2fc>=2, na.rm=TRUE)
```

```
[1] 1910
```

> Q7. What is the purpose of the arr.ind argument in the which() function call above? Why would we then take the first column of the output and need to call the unique() function?

arr.ind =TRUE in which() allows the function to return the indices of the zero values. We need to take the first column because that is the column with the row numbers that we need to remove in the countData. unique() function remove the row names and return only a vector of row numbers that we need to remove.

```
zero.vals <- which(meancounts[,1:2]==0, arr.ind=TRUE)
to.rm <- unique(zero.vals[,1])
mycounts <- meancounts[-to.rm,]
head(mycounts)
```

```
                control.means treated.means      log2fc
ENSG00000000003        900.75        658.00 -0.45303916
ENSG00000000419        520.50        546.00  0.06900279
ENSG00000000457        339.75        316.50 -0.10226805
ENSG00000000460         97.25         78.75 -0.30441833
ENSG00000000971       5219.00       6687.50  0.35769358
ENSG00000001036       2327.00       1785.75 -0.38194109
```

Q8. Using the up.ind vector above can you determine how many up regulated genes we have at the greater than 2 fc level?

```
up.ind <- mycounts$log2fc > 2
sum(up.ind)
```

```
[1] 250
```

Q9. Using the down.ind vector above can you determine how many down regulated genes we have at the greater than 2 fc level?

```
down.ind <- mycounts$log2fc < (-2)
sum(down.ind)
```

```
[1] 367
```

Q10. Do you trust these results? Why or why not?

No, because we have the chance to get many false-positive results.

How to prove if the data are significantly different in a different way?

To do this we need to use DESeq2 package ##DESeq2 analysis

```
head(library(DESeq2))
```

```
Loading required package: S4Vectors

Loading required package: stats4

Loading required package: BiocGenerics


Attaching package: 'BiocGenerics'
```

```
The following objects are masked from 'package:lubridate':

    intersect, setdiff, union

The following objects are masked from 'package:dplyr':

    combine, intersect, setdiff, union

The following objects are masked from 'package:stats':

    IQR, mad, sd, var, xtabs

The following objects are masked from 'package:base':

    anyDuplicated, aperm, append, as.data.frame, basename, cbind,
    colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
    get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,
    match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
    Position, rank, rbind, Reduce, rownames, sapply, setdiff, sort,
    table, tapply, union, unique, unsplit, which.max, which.min


Attaching package: 'S4Vectors'

The following objects are masked from 'package:lubridate':

    second, second<-

The following objects are masked from 'package:dplyr':

    first, rename

The following object is masked from 'package:tidyr':

    expand

The following object is masked from 'package:utils':

    findMatches
```

```
The following objects are masked from 'package:base':

    expand.grid, I, unname

Loading required package: IRanges


Attaching package: 'IRanges'

The following object is masked from 'package:lubridate':

    %within%

The following objects are masked from 'package:dplyr':

    collapse, desc, slice

The following object is masked from 'package:purrr':

    reduce

Loading required package: GenomicRanges

Loading required package: GenomeInfoDb

Loading required package: SummarizedExperiment

Loading required package: MatrixGenerics

Loading required package: matrixStats


Attaching package: 'matrixStats'

The following object is masked from 'package:dplyr':

    count
```

```
Attaching package: 'MatrixGenerics'

The following objects are masked from 'package:matrixStats':

    colAlls, colAnyNAs, colAnys, colAvgsPerRowSet, colCollapse,
    colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
    colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
    colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
    colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
    colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
    colWeightedMeans, colWeightedMedians, colWeightedSds,
    colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,
    rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
    rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
    rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
    rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
    rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
    rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
    rowWeightedSds, rowWeightedVars

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")'.


Attaching package: 'Biobase'

The following object is masked from 'package:MatrixGenerics':

    rowMedians

The following objects are masked from 'package:matrixStats':

    anyMissing, rowMedians
```

```
Warning: replacing previous import 'S4Arrays::read_block' by
'DelayedArray::read_block' when loading 'SummarizedExperiment'
```

```
[1] "DESeq2"               "SummarizedExperiment" "Biobase"
[4] "MatrixGenerics"       "matrixStats"          "GenomicRanges"
```

To use DESeq we need our input countData and colData in a specific format that DESeq2 wants:

```
dds <- DESeqDataSetFromMatrix(countData = countData, colData = metaData, design = ~dex)
```

```
converting counts to integer mode
```

```
Warning in DESeqDataSet(se, design = design, ignoreRank): some variables in
design formula are characters, converting to factors
```

To run the analysiss I can now use the main DESeq2 function called `DESeq()` with `dds` as input

```
dds <- DESeq(dds)
```

```
estimating size factors
```

```
estimating dispersions
```

```
gene-wise dispersion estimates
```

```
mean-dispersion relationship
```

```
final dispersion estimates
```

```
fitting model and testing
```

To get the results out of this `dds` object we can use the `results()` function from the package.

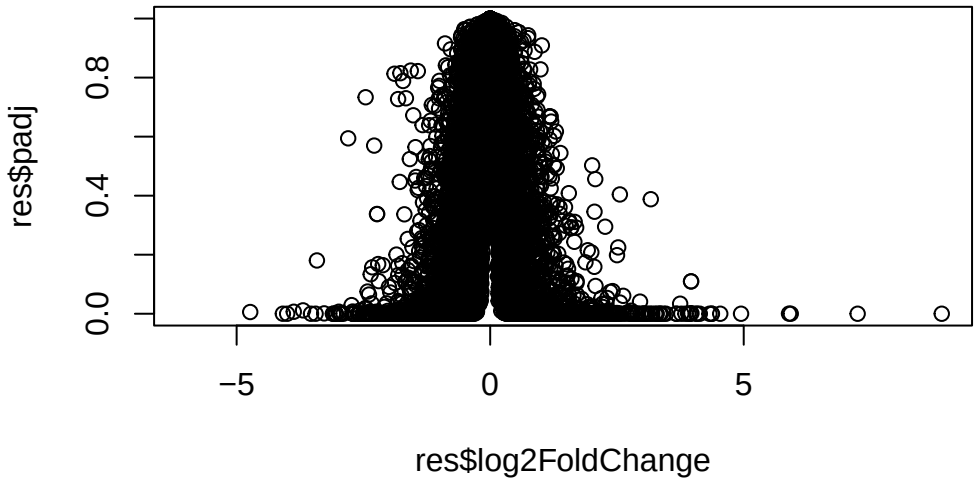
`padj` = BH adjusted p-value, corrected for multiple testing using BH method; p-value = 0.05, 95% confidence, 5% of conincidence

```
res <- results(dds)
head(res)
```

```
log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 6 rows and 6 columns
                  baseMean log2FoldChange     lfcSE      stat    pvalue
                 <numeric>      <numeric> <numeric> <numeric> <numeric>
ENSG00000000003 747.194195     -0.3507030  0.168246 -2.084470 0.0371175
ENSG00000000005   0.000000             NA        NA        NA        NA
ENSG00000000419 520.134160      0.2061078  0.101059  2.039475 0.0414026
ENSG00000000457 322.664844      0.0245269  0.145145  0.168982 0.8658106
ENSG00000000460  87.682625     -0.1471420  0.257007 -0.572521 0.5669691
ENSG00000000938   0.319167     -1.7322890  3.493601 -0.495846 0.6200029
                     padj
                <numeric>
ENSG00000000003  0.163035
ENSG00000000005        NA
ENSG00000000419  0.176032
ENSG00000000457  0.961694
ENSG00000000460  0.815849
ENSG00000000938        NA
```

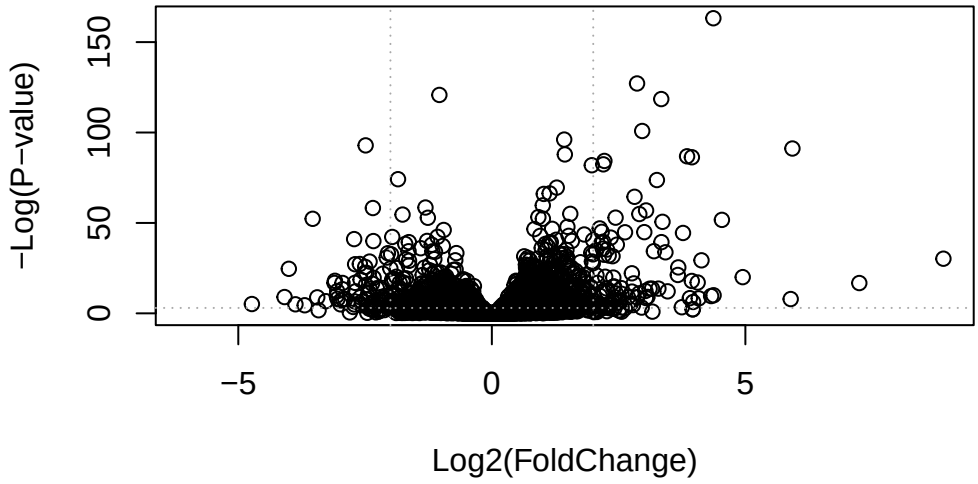
Let's make a final (for today) plot of log2 fold-change vs. the adjusted P-value 0 x-axis mean no change; other values either up regulated or down regulated

```
plot( res$log2FoldChange,  res$padj)
```



It is the low P-values that we care about and there are lost in the skewed plot above. Let's take the -log of the res$pdj values for our plot. Points on the top are significantly different, having low p-values. On the right, upregulated; On the left, down regulated

```
plot( res$log2FoldChange,  -log(res$padj),
      xlab="Log2(FoldChange)",
      ylab="-Log(P-value)")
abline(v=c(-2,2), col="darkgray", lty=9) # in the middle not up or down regulated enough
abline(h=-log(0.05), col="darkgray", lty=9) # below this not statistically significant
```

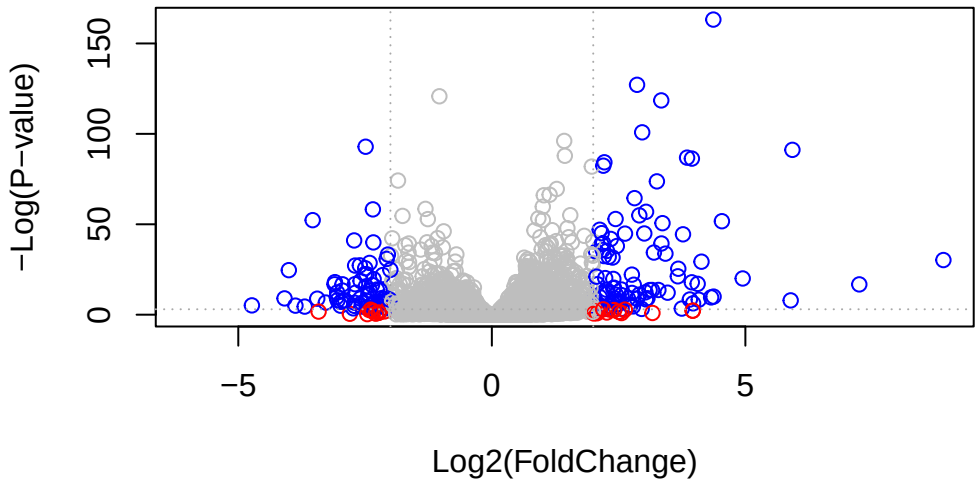


Finally, we can make a color vector to use in the plot to better highlight the genes we care about

```
mycols <- rep("gray", nrow(res))

mycols[res$log2FoldChange >= 2] <- "red"
mycols[res$log2FoldChange <= -2] <- "red"
mycols[res$padj <=0.05] <- "blue"
mycols[res$log2FoldChange <2 & res$log2FoldChange >-2 ] <- "grey"

plot( res$log2FoldChange,  -log(res$padj),
      col=mycols,
      xlab="Log2(FoldChange)",
      ylab="-Log(P-value)")
abline(v=c(-2,2), col="darkgray", lty=9)
abline(h=-log(0.05), col="darkgray", lty=9)
```



Still To Do: - Add annotation - Save results as CSV file - Do some Pathway Analysis

We can use AnnotationDbi package to add annotations data such as gene identifiers from different sources to our results object

```
res
```

```
log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 38694 rows and 6 columns
                baseMean log2FoldChange     lfcSE      stat    pvalue
               <numeric>      <numeric> <numeric> <numeric> <numeric>
ENSG00000000003  747.1942     -0.3507030  0.168246 -2.084470 0.0371175
ENSG00000000005    0.0000             NA        NA        NA        NA
ENSG00000000419  520.1342      0.2061078  0.101059  2.039475 0.0414026
ENSG00000000457  322.6648      0.0245269  0.145145  0.168982 0.8658106
ENSG00000000460   87.6826     -0.1471420  0.257007 -0.572521 0.5669691
...                    ...            ...       ...       ...       ...
ENSG00000283115  0.000000             NA        NA        NA        NA
ENSG00000283116  0.000000             NA        NA        NA        NA
ENSG00000283119  0.000000             NA        NA        NA        NA
ENSG00000283120  0.974916      -0.668258   1.69456 -0.394354  0.693319
ENSG00000283123  0.000000             NA        NA        NA        NA
                      padj
                 <numeric>
ENSG00000000003  0.163035
ENSG00000000005        NA
ENSG00000000419  0.176032
```

```
ENSG00000000457  0.961694
ENSG00000000460  0.815849
...                   ...
ENSG00000283115        NA
ENSG00000283116        NA
ENSG00000283119        NA
ENSG00000283120        NA
ENSG00000283123        NA
```

```
library("AnnotationDbi")
```

```
Attaching package: 'AnnotationDbi'

The following object is masked from 'package:dplyr':

    select
```

```
library("org.Hs.eg.db")
```

We can translate (a.k.a. map) the genes to the following formats

```
columns(org.Hs.eg.db)
```

```
 [1] "ACCNUM"       "ALIAS"        "ENSEMBL"      "ENSEMBLPROT"  "ENSEMBLTRANS"
 [6] "ENTREZID"     "ENZYME"       "EVIDENCE"     "EVIDENCEALL"  "GENENAME"
[11] "GENETYPE"     "GO"           "GOALL"        "IPI"          "MAP"
[16] "OMIM"         "ONTOLOGY"     "ONTOLOGYALL"  "PATH"         "PFAM"
[21] "PMID"         "PROSITE"      "REFSEQ"       "SYMBOL"       "UCSCKG"
[26] "UNIPROT"
```

My id are stored as rownames of `res`

```
head(rownames(res))
```

```
[1] "ENSG00000000003" "ENSG00000000005" "ENSG00000000419" "ENSG00000000457"
[5] "ENSG00000000460" "ENSG00000000938"
```

```
res$symbol <- mapIds(org.Hs.eg.db,
                     keys=row.names(res), # Our genenames
                     keytype="ENSEMBL",        # The format of our genenames
                     column="SYMBOL",          # The new format we want to add
                     multiVals="first")
```

```
'select()' returned 1:many mapping between keys and columns
```

```
head(res)
```

```
log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 6 rows and 7 columns
                  baseMean log2FoldChange     lfcSE      stat    pvalue
                 <numeric>      <numeric> <numeric> <numeric> <numeric>
ENSG00000000003 747.194195     -0.3507030  0.168246 -2.084470 0.0371175
ENSG00000000005   0.000000             NA        NA        NA        NA
ENSG00000000419 520.134160      0.2061078  0.101059  2.039475 0.0414026
ENSG00000000457 322.664844      0.0245269  0.145145  0.168982 0.8658106
ENSG00000000460  87.682625     -0.1471420  0.257007 -0.572521 0.5669691
ENSG00000000938   0.319167     -1.7322890  3.493601 -0.495846 0.6200029
                     padj      symbol
                <numeric> <character>
ENSG00000000003  0.163035      TSPAN6
ENSG00000000005        NA        TNMD
ENSG00000000419  0.176032        DPM1
ENSG00000000457  0.961694       SCYL3
ENSG00000000460  0.815849       FIRRM
ENSG00000000938        NA         FGR
```

Q11. Run the mapIds() function two more times to add the Entrez ID and UniProt accession and GENENAME as new columns called res$entrez$, res$uniprot and res$genename.

```
res$entrez <- mapIds(org.Hs.eg.db,
                     keys=row.names(res), # Our genenames
                     keytype="ENSEMBL",        # The format of our genenames
                     column="ENTREZID",          # The new format we want to add
                     multiVals="first")
```

```
'select()' returned 1:many mapping between keys and columns
```

```
res$genename <- mapIds(org.Hs.eg.db,
                       keys=row.names(res), # Our genenames
                       keytype="ENSEMBL",        # The format of our genenames
                       column="GENENAME",          # The new format we want to add
                       multiVals="first")
```

```
'select()' returned 1:many mapping between keys and columns
```

```
res$uniprot <- mapIds(org.Hs.eg.db,
                       keys=row.names(res), # Our genenames
                       keytype="ENSEMBL",        # The format of our genenames
                       column="UNIPROT",          # The new format we want to add
                       multiVals="first")
```

```
'select()' returned 1:many mapping between keys and columns
```

Let's write a csv file for our results

```
write.csv(res, file="deseq_results.csv")
```

Pathway Analysis

We can use the KEGG database of biological pathways to get some more insight into our differentially expressed genes

#pathview package for drawing illustration

```
library(pathview)
```

```
##############################################################################
Pathview is an open source software package distributed under GNU General
Public License version 3 (GPLv3). Details of GPLv3 is available at
http://www.gnu.org/licenses/gpl-3.0.html. Particullary, users are required to
formally cite the original Pathview paper (not just mention it) in publications
or products. For details, do citation("pathview") within R.
```

```
The pathview downloads and uses KEGG data. Non-academic uses may require a KEGG
license agreement (details at http://www.kegg.jp/kegg/legal.html).
##############################################################################
```

```
library(gage)
```

```
library(gageData)
```

Look at the first two KEGG pathways

```
data(kegg.sets.hs)

# Examine the first 2 pathways in this kegg set for humans
head(kegg.sets.hs, 2)
```

```
$`hsa00232 Caffeine metabolism`
[1] "10"   "1544" "1548" "1549" "1553" "7498" "9"

$`hsa00983 Drug metabolism - other enzymes`
 [1] "10"     "1066"   "10720"  "10941"  "151531" "1548"   "1549"   "1551"
 [9] "1553"   "1576"   "1577"   "1806"   "1807"   "1890"   "221223" "2990"
[17] "3251"   "3614"   "3615"   "3704"   "51733"  "54490"  "54575"  "54576"
[25] "54577"  "54578"  "54579"  "54600"  "54657"  "54658"  "54659"  "54963"
[33] "574537" "64816"  "7083"   "7084"   "7172"   "7363"   "7364"   "7365"
[41] "7366"   "7367"   "7371"   "7372"   "7378"   "7498"   "79799"  "83549"
[49] "8824"   "8833"   "9"      "978"
```

Make a new vector of fold-change values that I will use as input for `gage` this weill have the ENTREZ IDs as names

```
foldchanges = res$log2FoldChange
names(foldchanges) = res$entrez
head(foldchanges)
```

```
       7105       64102        8813       57147       55732        2268
-0.35070302          NA  0.20610777  0.02452695 -0.14714205 -1.73228897
```

```
x <- 1:3
names(x) <- c("cgabdra", "lisa","tina")
x
```

```
cgabdra    lisa    tina
      1       2       3
```

#order by overlap, largest overlap to smallest; take res objects, pathways as two inputs

```
keggres = gage(foldchanges, gsets=kegg.sets.hs)
```

```
attributes(keggres)
```

```
$names
[1] "greater" "less"    "stats"
```

Look at the top 3 "less"; keg identifier e.g.hsa05310

```
head(keggres$less,3)
```

```
                                     p.geomean stat.mean        p.val
hsa05332 Graft-versus-host disease 0.0004250461 -3.473346 0.0004250461
hsa04940 Type I diabetes mellitus  0.0017820293 -3.002352 0.0017820293
hsa05310 Asthma                    0.0020045888 -3.009050 0.0020045888
                                       q.val set.size         exp1
hsa05332 Graft-versus-host disease 0.09053483       40 0.0004250461
hsa04940 Type I diabetes mellitus  0.14232581       42 0.0017820293
hsa05310 Asthma                    0.14232581       29 0.0020045888
```

Now I can use the KEGG IDs of these pathways from gage to view our genes mapped to these pathways.

```
pathview(gene.data=foldchanges, pathway.id="hsa05310")
```

```
'select()' returned 1:1 mapping between keys and columns
```

```
Info: Working in directory /Users/qianqiantina/Desktop/BIMM143/Lab12_inclass
```

```
Info: Writing image file hsa05310.pathview.png
```

Q12. Can you do the same procedure as above to plot the pathview figures for the top 2 down-reguled pathways?

```
pathview(gene.data=foldchanges, pathway.id="hsa05332")
```

```
'select()' returned 1:1 mapping between keys and columns
```

```
Info: Working in directory /Users/qianqiantina/Desktop/BIMM143/Lab12_inclass
```

```
Info: Writing image file hsa05332.pathview.png
```

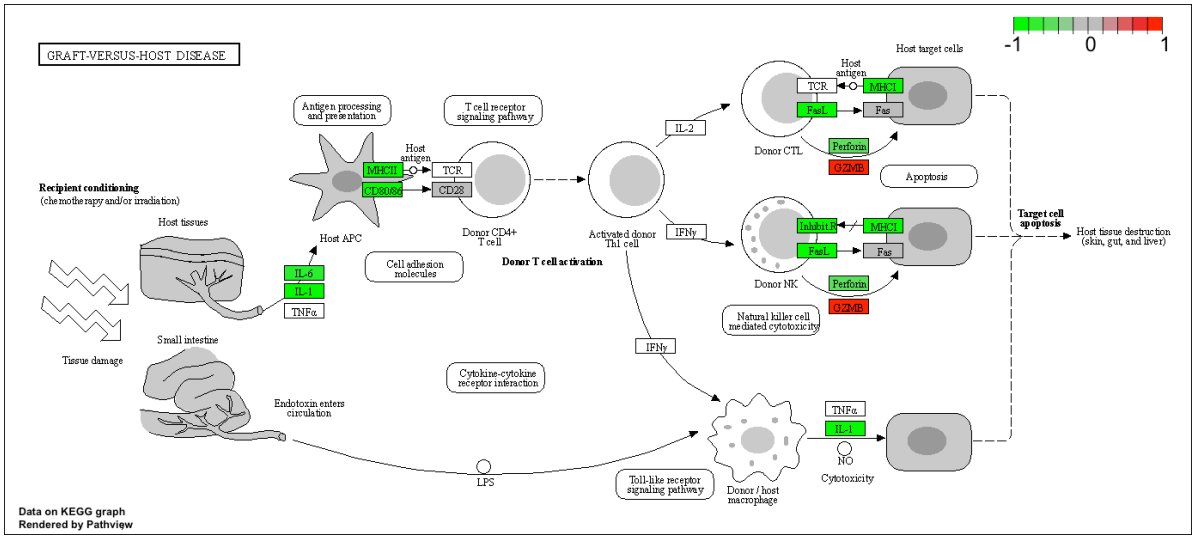


Figure 1: Asthma pathway from KEGG with our down regulated gene hsa05332shown in color

```
pathview(gene.data=foldchanges, pathway.id="hsa04940")
```

```
'select()' returned 1:1 mapping between keys and columns
```

```
Info: Working in directory /Users/qianqiantina/Desktop/BIMM143/Lab12_inclass
```

```
Info: Writing image file hsa04940.pathview.png
```

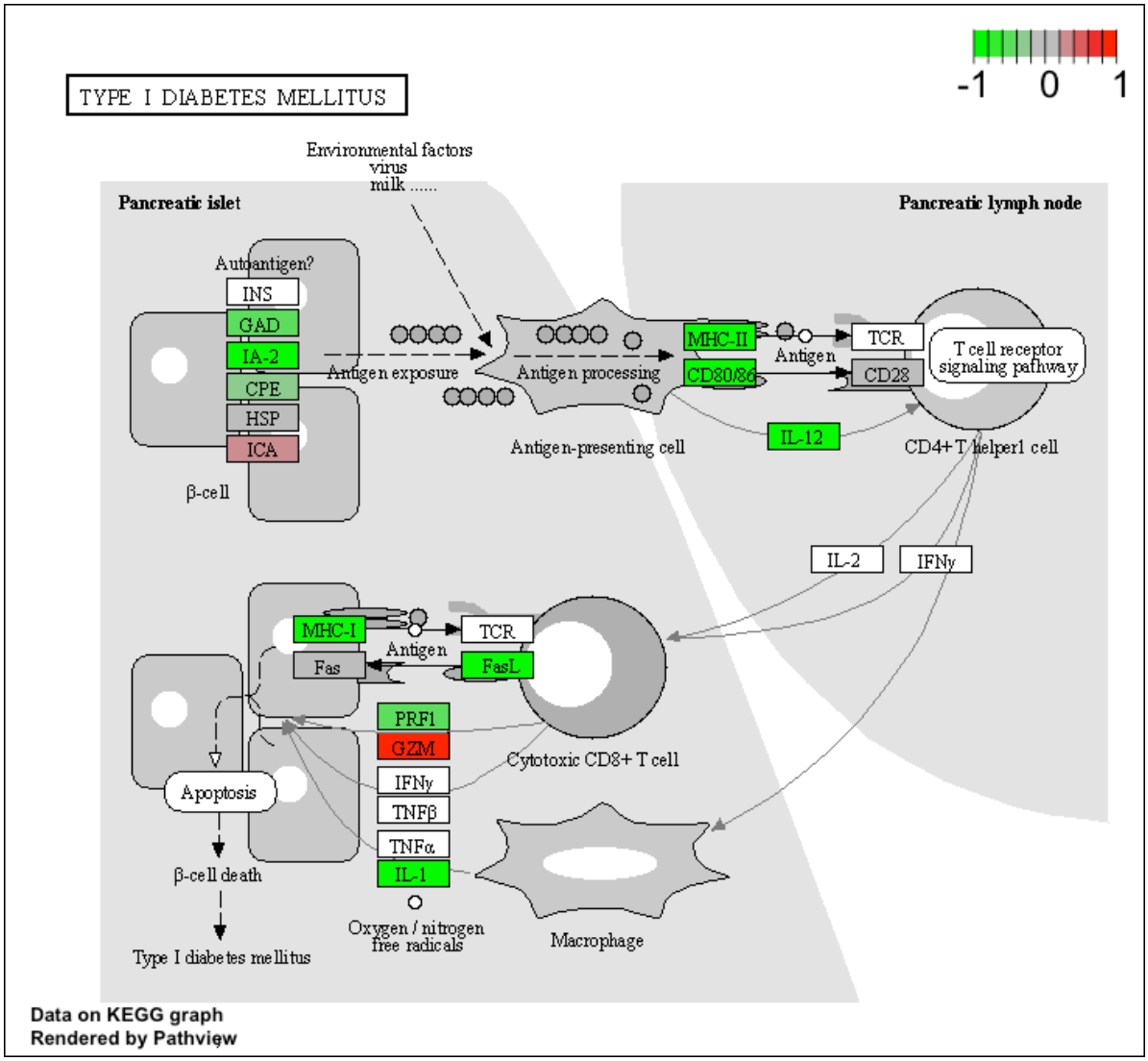


Figure 2: Asthma pathway from KEGG with our down regulated gene hsa04940 shown in color

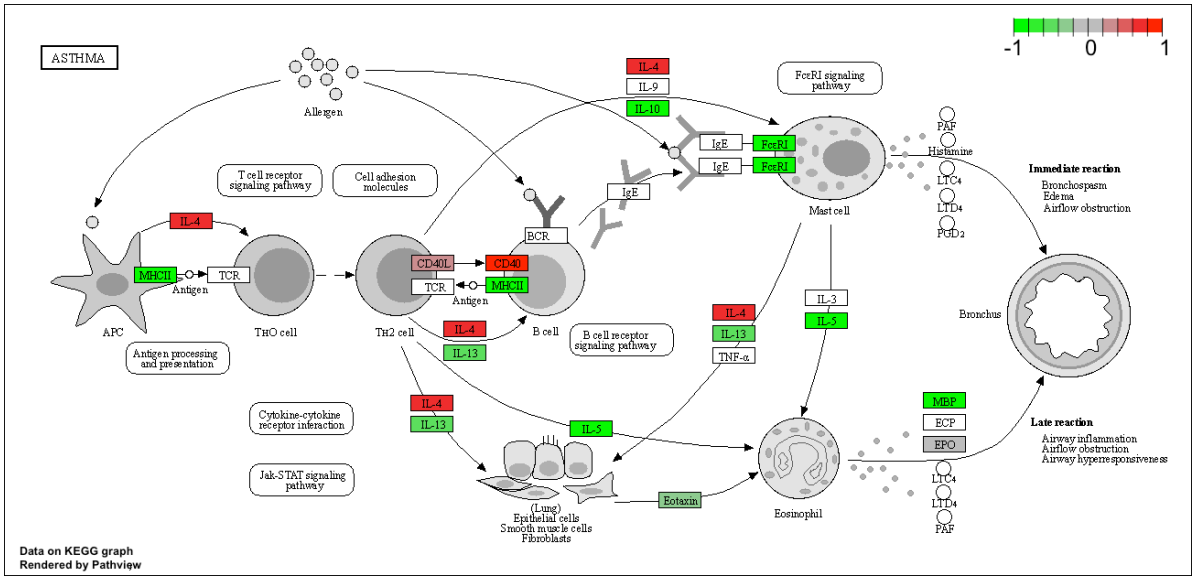


Figure 3: Asthma pathway from KEGG with our down regulated gene hsa05310 shown in color