Class 13

Derek Chang (PID: A16942232)

```
library(BiocManager)
library(DESeq2)
```

```
Loading required package: S4Vectors
```

```
Loading required package: stats4
```

```
Loading required package: BiocGenerics
```

```

Attaching package: 'BiocGenerics'
```

```
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, saveRDS, setdiff,
    table, tapply, union, unique, unsplit, which.max, which.min
```

```

Attaching package: 'S4Vectors'
```

```
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

Loading required package: GenomicRanges

Loading required package: GenomeInfoDb

Loading required package: SummarizedExperiment

Loading required package: MatrixGenerics

Loading required package: matrixStats


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
```

Today we will analyze some RNASeq data from Himes et al. on the effects of dexamethasome (dex), a synthetic glucocorticoid steroid on airway smooth muscle cells (ASM).

3. Importing countData and colData

```
counts <- read.csv("airway_scaledcounts.csv", row.names = 1)
metadata <- read.csv("airway_metadata.csv")
```

```
head(counts)
```

```
                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
```

```
head(metadata)
```

```
          id     dex celltype     geo_id
1 SRR1039508 control   N61311 GSM1275862
2 SRR1039509 treated   N61311 GSM1275863
3 SRR1039512 control  N052611 GSM1275866
4 SRR1039513 treated  N052611 GSM1275867
5 SRR1039516 control  N080611 GSM1275870
6 SRR1039517 treated  N080611 GSM1275871
```

Q1. How many genes are in this dataset?

```
nrow(counts)
```

```
[1] 38694
```

There are 38694 genes in the dataset.

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

```
table(metadata$dex)
```

```
control treated
      4       4
```

We have 4 'control' cell lines.

Toy differential expression analysis

Calculate the mean per gene count values for all "control" samples (`count` columns) and do the same for the "treated" samples. We will compare the mean per gene counts against each other.

1. Find all control values/columns in `count`

```
control <- metadata[metadata[,"dex"]=="control",]
```

2. Find the mean per gene for all "control" columns

   Q3. How would you make the above code in either approach more robust? Is there a function that could help here?

```
#control.mean <- rowSums(control.counts)/4
```

I will use the rowSums function to make the approach more robust even if more samples will be added.

```
control.counts <- counts[ ,control$id]
control.mean <- rowSums( control.counts )/4
head(control.mean)
```

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

3. Do the same steps to find the `treated.mean` values

   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.mean)

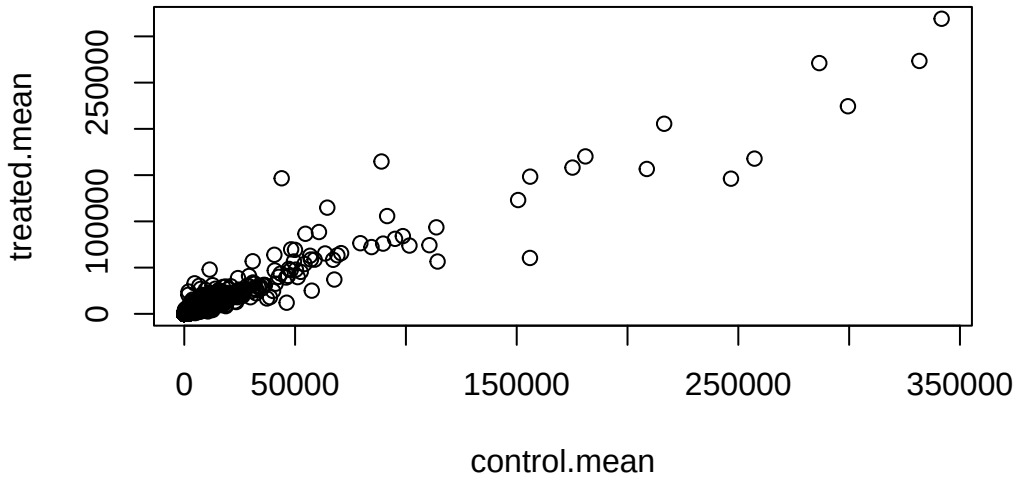
```
treated <- metadata[metadata[,"dex"]=="treated",]
treated.counts <- counts[ ,treated$id]
treated.mean <- rowSums( treated.counts )/4
head(treated.mean)
```

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

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

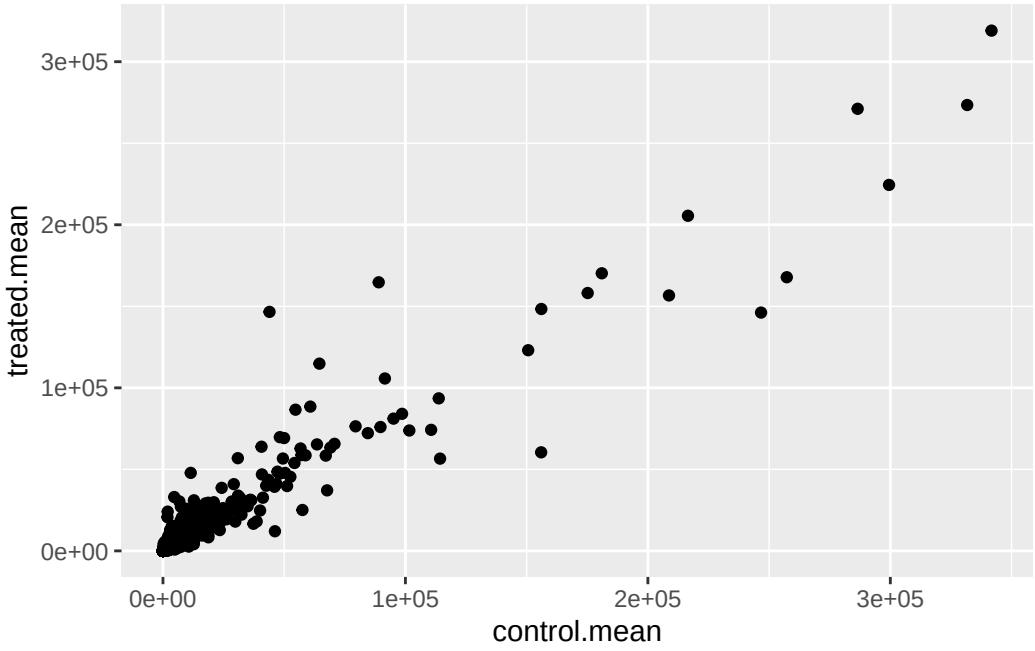
Q5 (a). Create a scatter plot showing the mean of the treated samples against the mean of the control samples. Your plot should look something like the following.

```
plot(meancounts)
```



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.mean, treated.mean)) + geom_point()
```



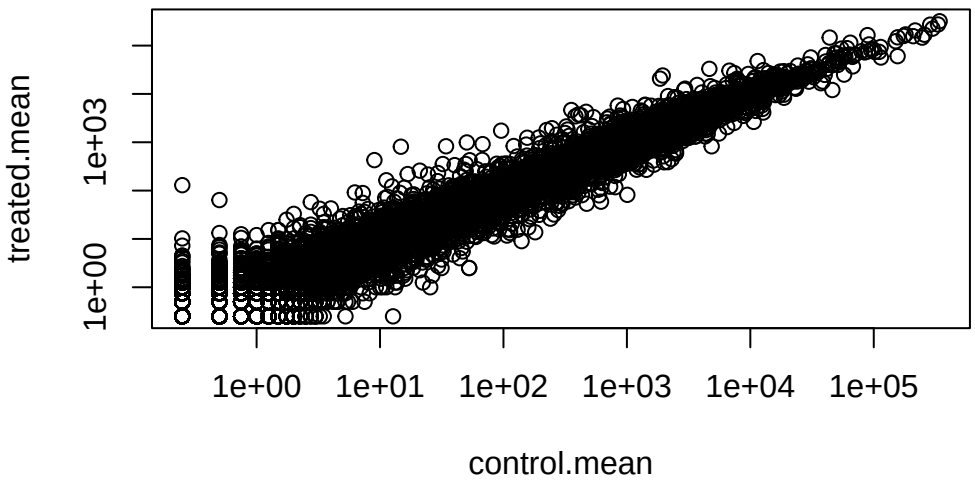
I used `geom_point`for this graph.

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

```
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
```



We most frequently use log2 transformation for this type of data.

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

```
[1] 0
```

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

```
[1] -1
```

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

```
[1] 1
```

These log2 values make the interpretation of "fold change" much easier, a rule of thumb in the field is a log2 fold of +2 or -2 is where we start to pay attention.

Let's calculate the log2 fold-change and add to our `meancounts` data.frame.

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

head(meancounts)
```

```
                control.mean treated.mean      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
```

> 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?

The purpose is to find which row or columns have zero counts for genes and samples. We will use `Unique` so that we don't count any row twice.

```
to.rm <- rowSums(meancounts[,1:2]==0) > 0
mycounts <- meancounts[!to.rm,]
```

> Q. How many genes do i have left after the count filtering

```
nrow(mycounts)
```

```
[1] 21817
```

> Q8. How many genes are "up" regulated due to drug treatement with a threshold of +2 log2-fold-changes?

1. I need to extract the log2fc values
2. I need to find those that are above +2
3. I need to count them

```
table(mycounts$log2fc > 2)
```

```
FALSE  TRUE
21567   250
```

There are 250 genes

> Q9. How many genes are "down" regulate due to drug treatement with a threshold of -2 log2-fold-changes?

```
table(mycounts$log2fc < -2)
```

```
FALSE  TRUE
21450   367
```

There are 367 genes.

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

No I don't trust these results as I don't know the significance of the results.

We are missing the stats here. Are the difference in the mean counts significant?

Let's do this analysis the right way and use the **DESeq2** package

##DESeq analysis

```
library(DESeq2)
```

The first function that we will use will setup the data in the way (format) DESeq wants it.

```
dds <- DESeqDataSetFromMatrix(countData = counts,
                        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
```

The function in the package is called `DEseq()` and we can run it on our new `dds` object. `dds` has empty columns for the results of DESeq.

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

```
estimating size factors
```

```
estimating dispersions
```

```
gene-wise dispersion estimates
```

```
mean-dispersion relationship
```

```
final dispersion estimates
```

```
fitting model and testing
```

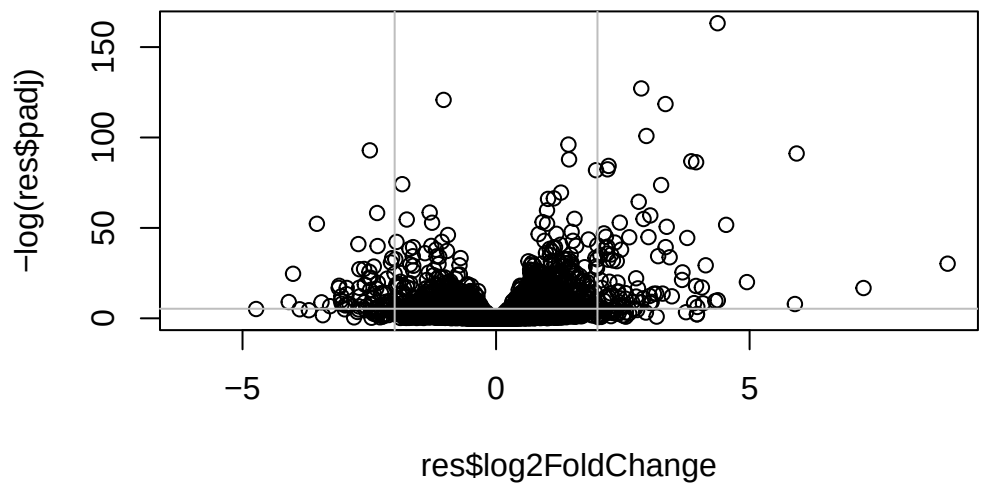
I will get the results from dds with the `results()` function:

```
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
```

Make a common overall results figure from this analysis called a volcano plot. This is designed to keep our inner biologist and stats happy, as it has the fold change and statistics. P-value vs Plot-fold-change. padj is a p value that has been adjusted

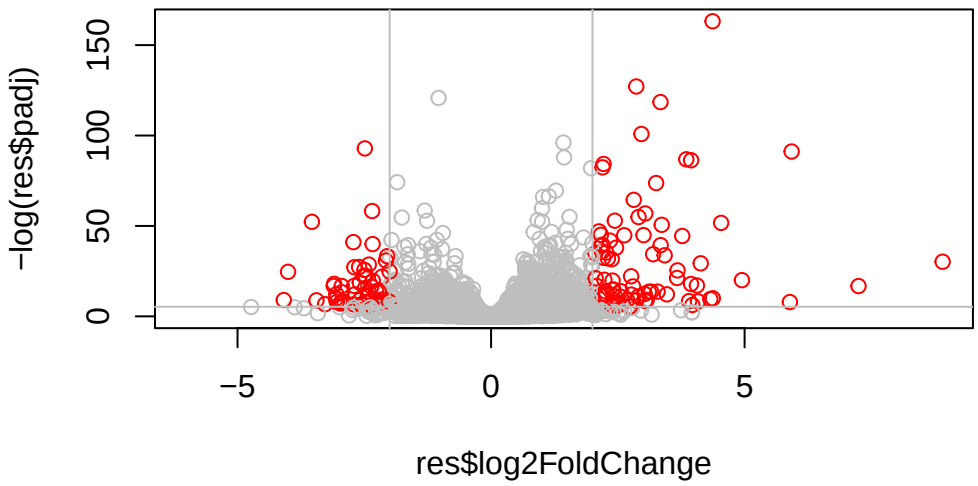
```
plot(res$log2FoldChange, -log(res$padj))
abline(v = c(-2,+2), col = "gray")
abline(h = -log(0.005), col = "gray")
```



Add some color to the plot:

```
mycols <- rep("gray", nrow(res))
mycols[res$log2FoldChange > 2] <- "red"
mycols[res$log2FoldChange < -2] <- "red"
mycols[-log(res$padj) < -log(0.005)] <- "gray"


plot(res$log2FoldChange, -log(res$padj), col = mycols)
abline(v = c(-2,+2), col = "gray")
abline(h = -log(0.005), col = "gray")
```



I want to save my results to date out to disc

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

What are these genes of interest

Gene Annotation

I need to translate our gene identifiers "ENSG" into gene names that the rest of the world can understand.

To do this "annotation" I can use the *"AnnotationDbi"* package using `BiocManager::install("AnnotationDbi"`

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

```
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"
```

I will use the `mapIds()` function to "map" identifiers to those from different databases. I will go between "ENSEMBL" and "SYMBOL" (and then after "GENENAME")

```
res$symbol <- mapIds(org.Hs.eg.db,
       keys = rownames(res),
        keytype = "ENSEMBL",
        column = "SYMBOL")
```

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

ADD "GENENAME"

```
res$genename <- mapIds(org.Hs.eg.db,
       keys = rownames(res),
        keytype = "ENSEMBL",
        column = "GENENAME")
```

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

ADD "ENTREZ"

```
res$entrez <- mapIds(org.Hs.eg.db,
       keys = rownames(res),
        keytype = "ENSEMBL",
        column = "ENTREZID")
```

```
'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 9 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               genename      entrez
                <numeric> <character>            <character> <character>
ENSG00000000003  0.163035      TSPAN6          tetraspanin 6        7105
ENSG00000000005        NA        TNMD            tenomodulin       64102
ENSG00000000419  0.176032        DPM1 dolichyl-phosphate m..        8813
ENSG00000000457  0.961694       SCYL3 SCY1 like pseudokina..       57147
ENSG00000000460  0.815849       FIRRM FIGNL1 interacting r..       55732
ENSG00000000938        NA         FGR FGR proto-oncogene, ..        2268
```

Save our annotated results object.

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

Pathway Analysis

Now that we have our results with added annotations we can do some pathway mapping.

Let's use the **gage** package to look for KEGG pathways in our results "Genes of Interest" I will also use the **pathview** package to draw little pathway figures.

```
library(pathview)
library(gage)
library(gageData)
data(kegg.sets.hs)

head(kegg.sets.hs,1)
```

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

what **gage** wants as input is not the whole table/data.frame of results. It just wants a "vector of importance". For RNASeq data like this we have our log2FC values.

```
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
```

Now, let's run the gage pathway analysis

```
# Get the results
keggres = gage(foldchanges, gsets = kegg.sets.hs)
```

What is in this `keggres` object?

```
attributes(keggres)
```

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

```
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
```

Let's use the pathview package to look at one of the highlighted KEGG pathways with our genes highlighted."hsa05310 Asthma".

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

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

```
Info: Working in directory /Users/derek/Desktop/BIMM143/class13

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

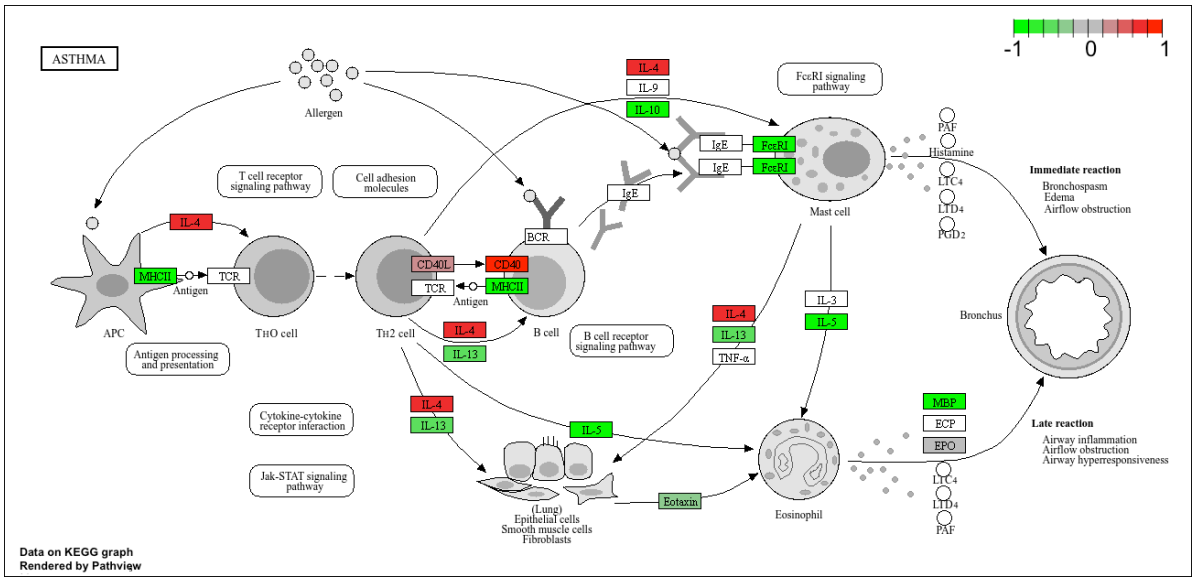


Figure 1: Asthma pathway with my DEGs