

# Class 13: RNASeq with DESeq2

Chloe Wong (PID: A16893383)

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

## Data import

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

A wee peak

```
head(counts)
```

	SRR1039508	SRR1039509	SRR1039512	SRR1039513	SRR1039516
ENSG000000000003	723	486	904	445	1170
ENSG000000000005	0	0	0	0	0
ENSG000000000419	467	523	616	371	582
ENSG000000000457	347	258	364	237	318
ENSG000000000460	96	81	73	66	118
ENSG000000000938	0	0	1	0	2
	SRR1039517	SRR1039520	SRR1039521		
ENSG000000000003	1097	806	604		
ENSG000000000005	0	0	0		
ENSG000000000419	781	417	509		
ENSG000000000457	447	330	324		
ENSG000000000460	94	102	74		
ENSG000000000938	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
```

Q2. How many ‘control’ cell lines do we have?

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

```
[1] 4
```

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

control	treated
4	4

## Toy differential expression analysis

Calculate the mean per gene count values for all “control” samples (i.e. columns in counts) and do the same for “treated” and then compare them.

1. Find all “control” values/columns in counts

```
control.inds <- metadata$dex == "control"  
control.counts <- counts[,control.inds]  
head(control.counts)
```

	SRR1039508	SRR1039512	SRR1039516	SRR1039520
ENSG000000000003	723	904	1170	806
ENSG000000000005	0	0	0	0
ENSG000000000419	467	616	582	417
ENSG000000000457	347	364	318	330
ENSG000000000460	96	73	118	102
ENSG000000000938	0	1	2	0

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

To make the code more robust, you would use the `apply` function to apply the mean to the column above.

2. Find the mean per gene across all control columns.

```
control.mean <- apply(control.counts, 1, mean)
```

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`)

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

```
treated.inds <- metadata$dex == "treated"
treated.counts <- counts[,treated.inds]
head(treated.counts)
```

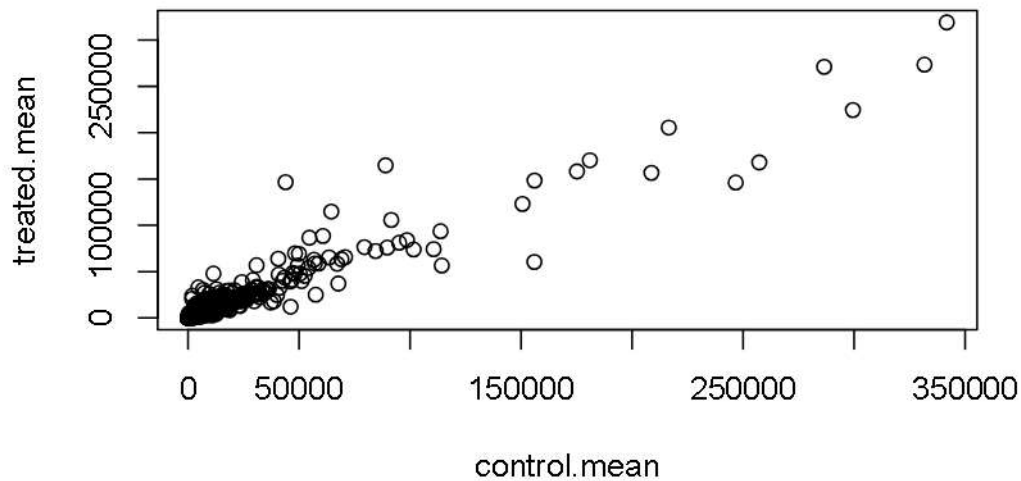
	SRR1039509	SRR1039513	SRR1039517	SRR1039521
ENSG000000000003	486	445	1097	604
ENSG000000000005	0	0	0	0
ENSG000000000419	523	371	781	509
ENSG000000000457	258	237	447	324
ENSG000000000460	81	66	94	74
ENSG000000000938	0	0	0	0

```
treated.mean <- apply(treated.counts, 1, mean)
```

```
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?

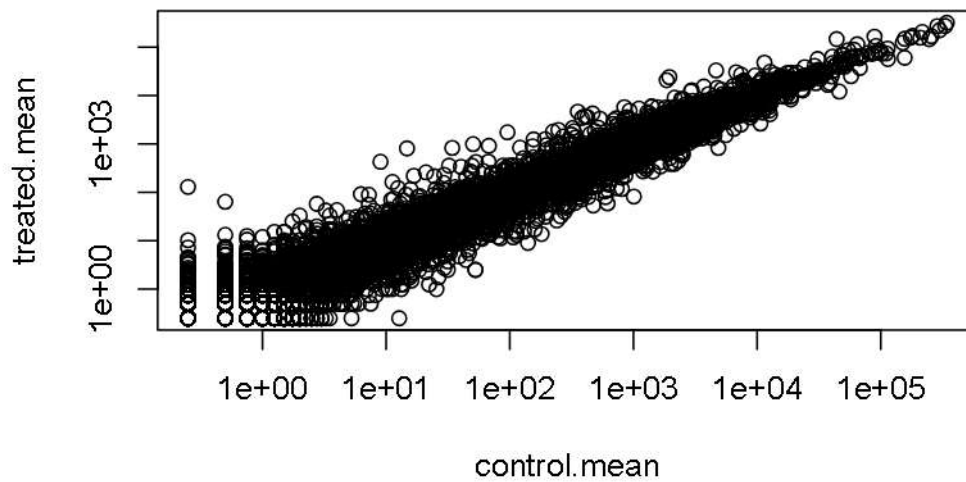
```
geom_(point)
```

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  $\leq 0$  omitted from logarithmic plot

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



We most frequently use  $\log_2$  transformations for this type of data.

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

```
[1] 0
```

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

```
[1] 1
```

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

```
[1] -1
```

These  $\log_2$  values make the interpretation of “fold-change” a little easier and a rule-of-thumb in the field is a  $\log_2$  fold-change of +2 or -2 is where we start to pay attention.

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

```
[1] 2
```

Let's calculate the  $\log_2(\text{fold-change})$  and add it to our `meancounts` data.frame.

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

	control.mean	treated.mean	log2fc
ENSG000000000003	900.75	658.00	-0.45303916
ENSG000000000005	0.00	0.00	NaN
ENSG000000000419	520.50	546.00	0.06900279
ENSG000000000457	339.75	316.50	-0.10226805
ENSG000000000460	97.25	78.75	-0.30441833
ENSG000000000938	0.75	0.00	-Inf

```
# meancounts[,1:2]==0
```

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

	control.mean	treated.mean	log2fc
ENSG000000000003	900.75	658.00	-0.45303916
ENSG000000000419	520.50	546.00	0.06900279
ENSG000000000457	339.75	316.50	-0.10226805
ENSG000000000460	97.25	78.75	-0.30441833
ENSG000000000971	5219.00	6687.50	0.35769358
ENSG000000001036	2327.00	1785.75	-0.38194109

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 of the `arr.ind` argument in the `which()` function call above is to return both the row and column indices (i.e. positions) where there are TRUE values. Genes (rows) and samples (columns) with 0 counts will be displayed. Any genes that have 0 counts in any sample will be ignored (focus on the row answer). Calling `unique()` means no row will be counted twice if it has zero entries in both samples.

Q. How many genes do I have left after this zero count filtering?



```
nrow(mycounts)
```

```
[1] 21817
```

Q. How many genes are “up” regulated upon drug treatment at a threshold of +2 log2-fold-change?

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

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

```
[1] 250
```

Q. How many genes are “down” regulated upon drug treatment?

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

```
[1] 367
```

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

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

```
[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?

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

```
[1] 367
```

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

I do not trust these results because fold-change can be large without being statistically significant as well as missing the stats of whether the differences we are seeing are significant. In order to obtain the stats, we need to use the DESeq2 package.

Wow hold on we are missing the stats here. Is the difference in the mean counts significant?

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

## DESeq analysis

```
#!/ message: false  
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

Attaching package: 'IRanges'

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

windows

Loading required package: GenomicRanges

Loading required package: GenomeInfoDb

Loading required package: SummarizedExperiment

Loading required package: MatrixGenerics

Loading required package: matrixStats

Warning: package 'matrixStats' was built under R version 4.4.2

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

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 dds object.

```
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

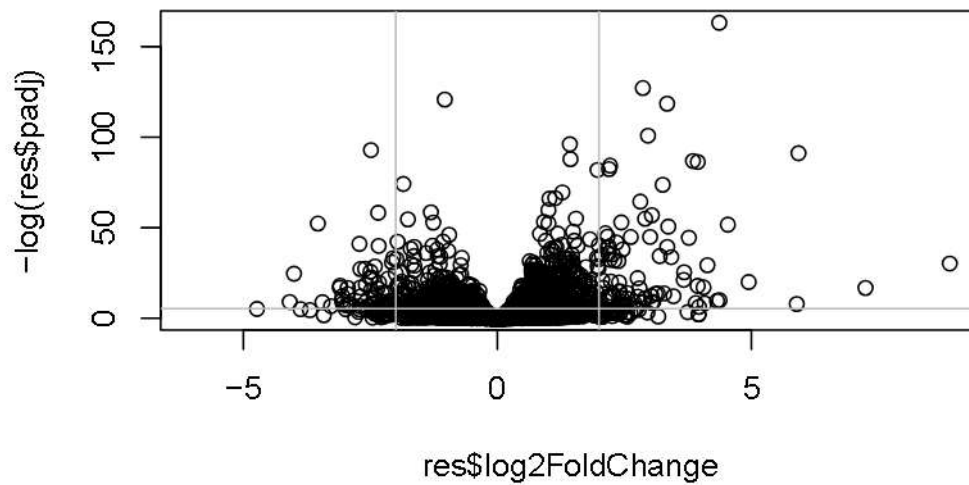
Wald test p-value: dex treated vs control

DataFrame with 6 rows and 6 columns

	baseMean	log2FoldChange	lfcSE	stat	pvalue
	<numeric>	<numeric>	<numeric>	<numeric>	<numeric>
ENSG000000000003	747.194195	-0.3507030	0.168246	-2.084470	0.0371175
ENSG000000000005	0.000000	NA	NA	NA	NA
ENSG000000000419	520.134160	0.2061078	0.101059	2.039475	0.0414026
ENSG000000000457	322.664844	0.0245269	0.145145	0.168982	0.8658106
ENSG000000000460	87.682625	-0.1471420	0.257007	-0.572521	0.5669691
ENSG000000000938	0.319167	-1.7322890	3.493601	-0.495846	0.6200029
	padj				
	<numeric>				
ENSG000000000003	0.163035				
ENSG000000000005	NA				
ENSG000000000419	0.176032				
ENSG000000000457	0.961694				
ENSG000000000460	0.815849				
ENSG000000000938	NA				

Make a common overall results figure from this analysis. This is designed to keep our inner biologist and inner stats nerd happy.

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



```
log(0.0005)
```

```
[1] -7.600902
```

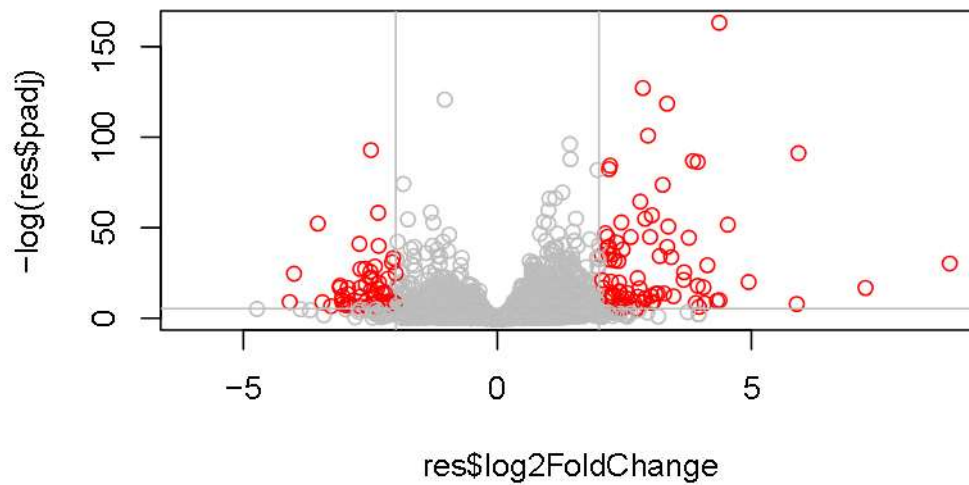
```
log(0.00000000005)
```

```
[1] -23.719
```

Add some color to this plot:

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

We will pick this up next day and add annotation i.e. what are these genes of interest) and do pathway analysis (what biology) are they known to be involved with.

```
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>
ENSG000000000003	747.194195	-0.3507030	0.168246	-2.084470	0.0371175
ENSG000000000005	0.000000	NA	NA	NA	NA
ENSG000000000419	520.134160	0.2061078	0.101059	2.039475	0.0414026
ENSG000000000457	322.664844	0.0245269	0.145145	0.168982	0.8658106
ENSG000000000460	87.682625	-0.1471420	0.257007	-0.572521	0.5669691
ENSG000000000938	0.319167	-1.7322890	3.493601	-0.495846	0.6200029
	padj				
	<numeric>				

```

ENSG000000000003 0.163035
ENSG000000000005      NA
ENSG000000000419 0.176032
ENSG000000000457 0.961694
ENSG000000000460 0.815849
ENSG000000000938      NA

```

## Annotation

I need to translate our gene identifiers “ENSG0000...” into gene names that the rest of the world can understand.

To this “annotation” I will use the AnnotationDbi package. I can install this with `BiocManager::install()`

```

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” my 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



```
#head(res)
```

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

```
#head(res)
```

Add "ENTREZID"

```
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)
```

Save our annotated results object.

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

## Pathway Analysis

Now that we have our results with added annotation 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.

```
#!/ message: false  
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)
```

```
data(kegg.sets.hs)
```

```
# Examine the first 2 pathways in this kegg set for humans
head(kegg.sets.hs, 1)
```

```
$`hsa00232 Caffeine metabolism`
```

```
[1] "10" "1544" "1548" "1549" "1553" "7498" "9"
```

What `gage` wants as input is not my big table/data.frame of results. It just wants a “vector of importance”. For RNASeq data like we have this is 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 these 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 C:/Users/chloe/OneDrive/Desktop/BIMM143/class13

Info: Writing image file hsa05310.pathview.png

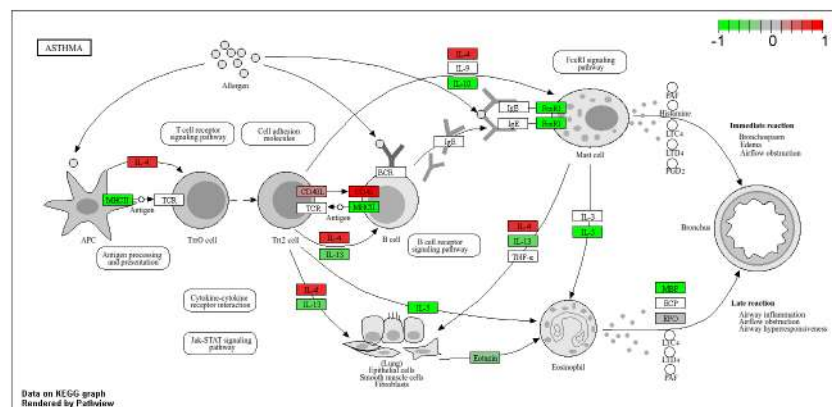


Figure 1: Asthma pathway with my DEGs