Class 13

Marina Puffer (PID: A16341339)

The data for this session comes from a published RNA-seq experiment where airway smooth muscle cells were treated with dexomethasone, a synthetic glucocorticoid steroid with anti-inflammatory effects.

3. Import 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
```

38694 genes in the dataset.

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

```
table(metadata [,2])
```

```
control treated
      4       4
```

There are 4 control cell lines

4. Toy differential gene expression

Some exploratory differential gene expression analysis: First find the sample id for those labeled control, then canculate the mean counts per gene across these samples. The values of the mean counts are sored in `control.mean`.

```
# find the sample IDs for those labeled control
control <- metadata[metadata[,"dex"]=="control",]
# take counts of nunmber of genes
control.counts <- counts[ ,control$id]
# summarize the amount of expression for each gene and stored as control.mean
```

```
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
```

Alternative is to use dplyr to do the same thing:

```
library(dplyr)
```

```
Attaching package: 'dplyr'

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

    filter, lag

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

    intersect, setdiff, setequal, union
```

```
control <- metadata %>% filter(dex=="control")
control.counts <- counts %>% select(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
```

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

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

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

Can find the mean of each gene by using the function `rowMeans`. This will allow the third line of code to be simplified and allows for a change in the total number of control samples without causing an error.

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 <- rowMeans( treated.counts )
head(treated.mean)
```

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

Combine the meancount data into a data frame

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

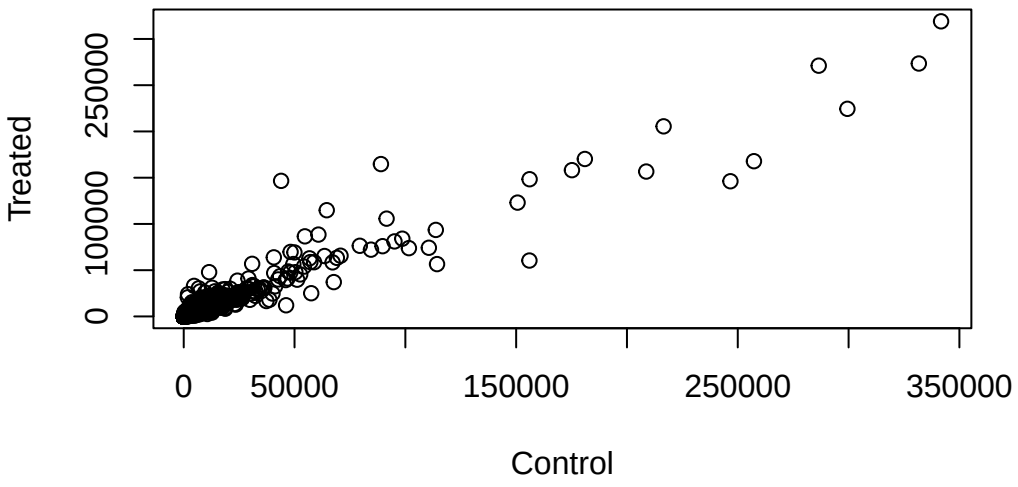
The sum of mean counts across all genes for each group

```
colSums(meancounts)
```

```
control.mean treated.mean
    23005324     22196524
```

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.

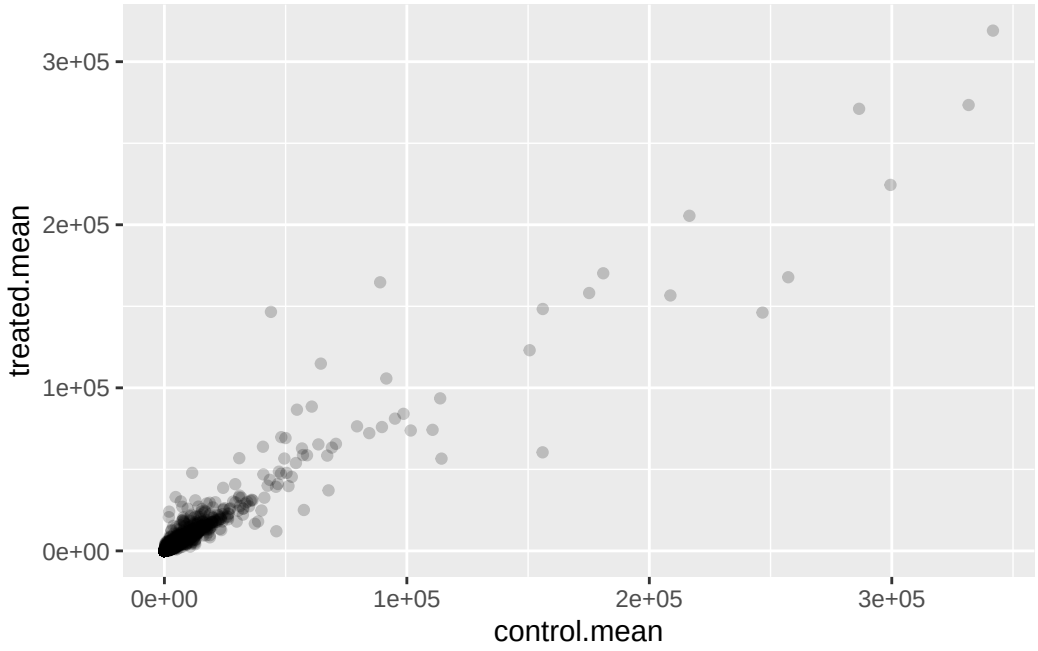
```
plot(meancounts, xlab="Control", ylab="Treated")
```



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?

The geom_point function should be used.

```
library(ggplot2)
ggplot(meancounts)+aes(control.mean, treated.mean)+geom_point(alpha=0.2)
```



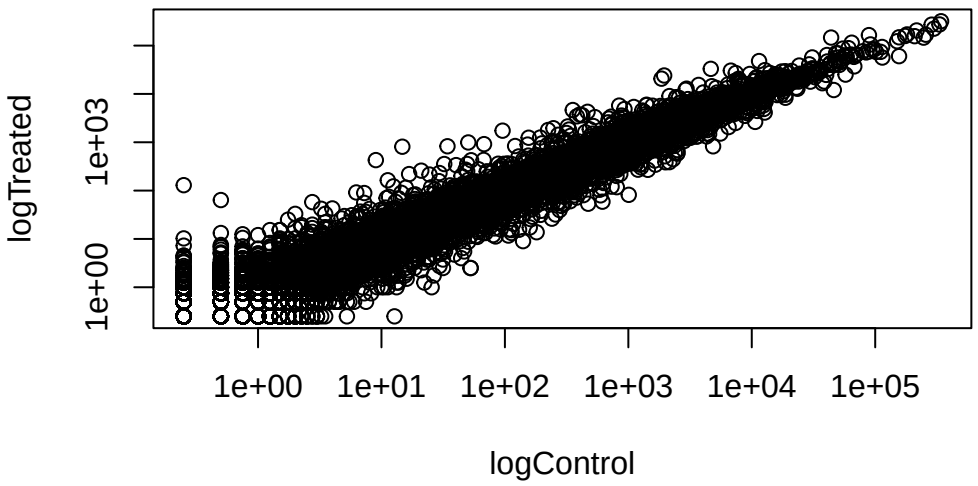
Q6. Try plotting both axes on a log scale. What is the argument to plot() that allows you to do this?

The argument `log=xy` allows both axes to be on the log scale.

```
plot(meancounts, log="xy", xlab="logControl", ylab="logTreated")
```

```
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 find candidate differentially expressed genes by looking for genes with a large change between control and dex=treated samples. Log transformations are useful when the data is skewed and measured over a long range like this We can use different log transformations, such as the log2 of the fold change, because this has better mathmatical properties.

```
#Treated/Control
log2(10/10)
```

```
[1] 0
```

```
#Treated/Control
log2(20/10)
```

```
[1] 1
```

1 when doubled fold change

```
#Treated/Control
log2(10/20)
```

```
[1] -1
```

-1 when halved

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

```
[1] 2
```

```
log10(40/10)
```

```
[1] 0.60206
```

Overall, log2 units are much easier to understand than other log transformations.

Calculate the log2foldchange and add it to the **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
```

Negative values = downregulated Positive values = upregulated

A couple of odd things here: NaN which is returned when you divide by 0 and take a log, and -Inf which is returned when you try to take the log of 0. This is because there are a lot of genes with 0 expression in the dataset. We need to filter the data to remove those genes.

```
#meancounts[,1:2] identifies values in first 2 columns
#==0 returns TRUE/FALSE
#with rowSums, all values above 0 are genes that have at least some expression.
to.rm.inds <- rowSums(meancounts[,1:2]==0) > 0
# use a `!` to flip the trues and falses
mycounts <- meancounts[!to.rm.inds,]
```

```
dim(mycounts)
```

```
[1] 21817     3
```

```
head(mycounts)
```

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

Confirmed this worked, the second gene with 0 expression is gone.

Common threshhold for calling something differentially expressed is a log2(FoldChange) of greater than 2 or less than -2. Let's filter the dataset both ways to see how many genes are up or down-regulated.

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

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(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?

```
sum(down.ind)
```

```
[1] 367
```

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

The fold change of 2 and -2 are kind of arbitrary thresholds, we don't actually know if those fold changes ae statistically significant. So, I don't trust those numbers yet.

5. Setting up for DESeq

We will use the DESeq2 package to do the analysis properly.

```
library(DESeq2)
```

```
Loading required package: S4Vectors

Loading required package: stats4

Loading required package: BiocGenerics


Attaching package: 'BiocGenerics'

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:dplyr':

    first, rename

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 objects are masked from 'package:dplyr':

    collapse, desc, slice

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
```

```
citation("DESeq2")
```

```
To cite package 'DESeq2' in publications use:

  Love, M.I., Huber, W., Anders, S. Moderated estimation of fold change
  and dispersion for RNA-seq data with DESeq2 Genome Biology 15(12):550
  (2014)

A BibTeX entry for LaTeX users is

  @Article{,
    title = {Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2
    author = {Michael I. Love and Wolfgang Huber and Simon Anders},
    year = {2014},
    journal = {Genome Biology},
    doi = {10.1186/s13059-014-0550-8},
    volume = {15},
    issue = {12},
    pages = {550},
  }
```

Set up the input object resady for DESeq

```
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
```

```
# ~ means go to column dex
dds
```

```
class: DESeqDataSet
dim: 38694 8
metadata(1): version
assays(1): counts
```

```
rownames(38694): ENSG00000000003 ENSG00000000005 ... ENSG00000283120
  ENSG00000283123
rowData names(0):
colnames(8): SRR1039508 SRR1039509 ... SRR1039520 SRR1039521
colData names(4): id dex celltype geo_id
```

Run the DESeq analysis

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

```
estimating size factors
```

```
estimating dispersions
```

```
gene-wise dispersion estimates
```

```
mean-dispersion relationship
```

```
final dispersion estimates
```

```
fitting model and testing
```

Get results back from `dds` object

```
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
```

padj = adjusted P value P value threshold is 5%, when dealing with these large datasets, 5% error becomes a very large population, which is not acceptable. So, the adjusted P value increases the P values and decreases the amount that are below the 5% threshhold.

A summary results plot

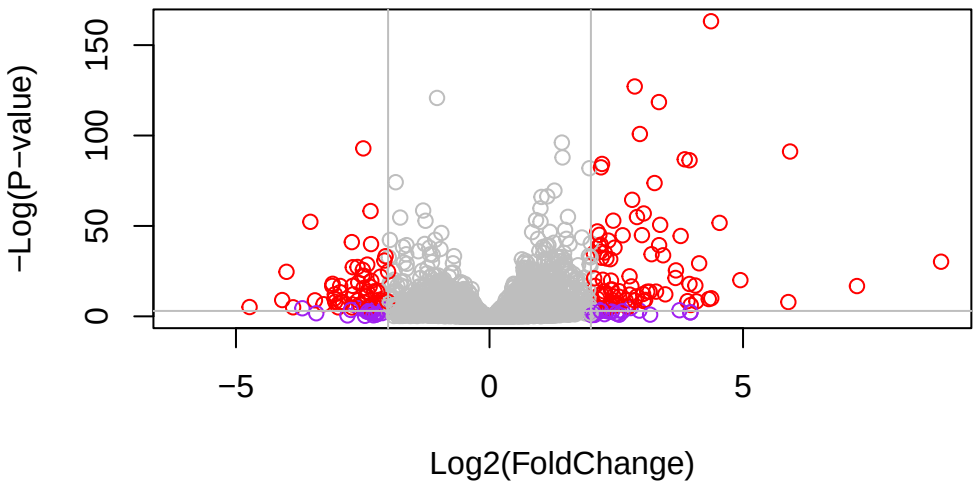
Volcano plot This is a common type of summary figure that keeps both our inner biologist and inner stats nerd happy. It shows both P value and Log2(Fold-changes)

```
#custom colors
mycols <- rep("gray", nrow(res))
mycols[ abs(res$log2FoldChange) > 2 ]  <- "purple"

inds <- (res$padj < 0.01) & (abs(res$log2FoldChange) > 2 )
mycols[ inds ] <- "red"

plot(res$log2FoldChange, -log(res$padj), col=mycols, xlab="Log2(FoldChange)", ylab="-Log(P

# Add some cut-off lines for fold change and P-value
abline(v=-2, col="grey")
abline(v=--2, col="grey")
abline(h=-log(0.05), col="grey")
```



On this plot, the purple points are the ones scientists should investigate, they have both low p-value and high fold change.

Save our results to date

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

8. Adding annotation data

The results table only contains the gene IDs, but alternative gene names and extra annotation are usually required for informative interpretation of our results.

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

```
Attaching package: 'AnnotationDbi'

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

    select
```

```
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"
```

The main function we will use here is called `mapIds()`

The current IDs are here

```
#mapIds
head(row.names(res))
```

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

These are in ENSEMBLE format, I want "SYMBOL" IDs.

```
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$symbol)
```

```
ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460
       "TSPAN6"          "TNMD"          "DPM1"         "SCYL3"         "FIRRM"
ENSG00000000938
          "FGR"
```

Now have gene symbols associated with the ENSEMBLE IDs. Let's add GENENAME

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

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

```
  head(res$genename)
```

```
                                                    ENSG00000000003
                                                    "tetraspanin 6"
                                                    ENSG00000000005
                                                      "tenomodulin"
                                                    ENSG00000000419
"dolichyl-phosphate mannosyltransferase subunit 1, catalytic"
                                                    ENSG00000000457
                                     "SCY1 like pseudokinase 3"
                                                    ENSG00000000460
  "FIGNL1 interacting regulator of recombination and mitosis"
                                                    ENSG00000000938
            "FGR proto-oncogene, Src family tyrosine kinase"
```

Now add entrez

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

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

```
  head(res$entrez)
```

```
ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460
         "7105"          "64102"           "8813"          "57147"          "55732"
ENSG00000000938
         "2268"
```

10. Pathway analysis

Use the **gage** package along with **pathview** here to do geneset enrichment (aka pathway analysis) and figure generation respectively.

Let's look at the first 2 pathways in KEGG

```
library(pathview)
library(gage)
library(gageData)

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"
```

What we need for `gage()` is our genes in ENTREZ ID format with a measure of their importance.

It wants a vector of e.g. fold changes.

```
foldchanges <-  res$log2FoldChange
head(foldchanges)
```

```
[1] -0.35070302          NA  0.20610777  0.02452695 -0.14714205 -1.73228897
```

Add ENTREZ IDs as `names()` to my `foldchanges` vector

```
names(foldchanges) <- res$entrez
head(foldchanges)
```

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

Now we can run `gage()` with this input vector and the geneset we want to examine for overlap/enrichment.

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

Look at the results

```
attributes(keggres)
```

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

```
# Look at the first three down (less) pathways
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
```

We can view these pathways with our geneset genes highlighted using the `pathview()` function. Ex: for asthma I will use the pathway ID hsa05310 as seen above.

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

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

```
Info: Working in directory /Users/marin/Desktop/BIMM 143/class13

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

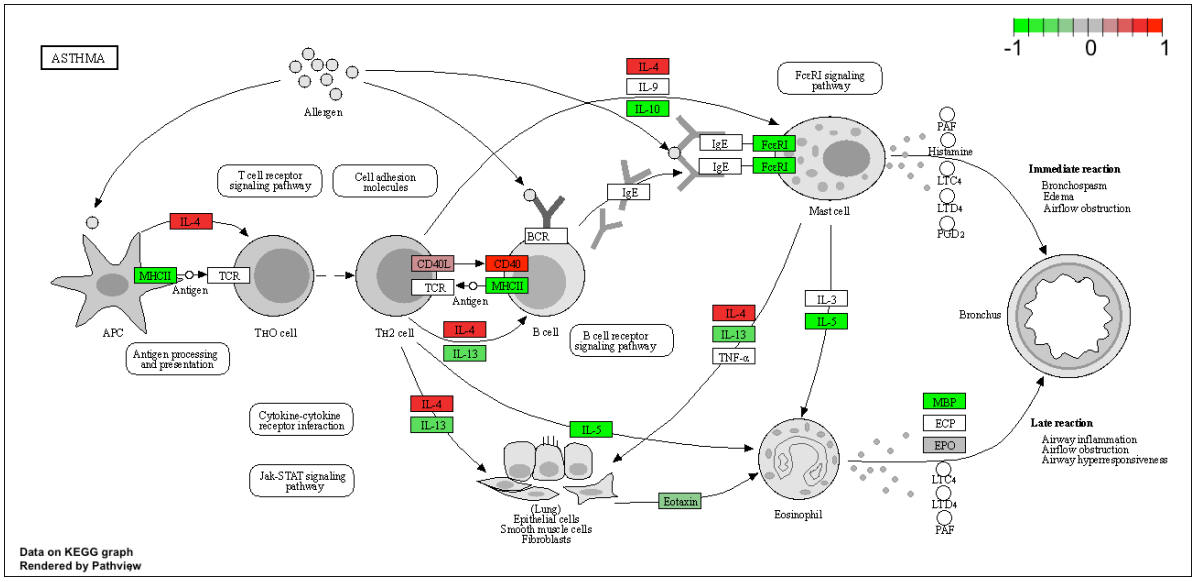


Figure 1: Genes involveed in asthma pathway