November 15, 2023 Class 13: Transcriptomics and the analysis of RNA-Seq data

Savannah Bogus A69027475

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

```r
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
```

```r
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 the dataset? 38,694 genes.

Q2

How many control cell lines are there?

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

```
control treated 
      4       4 
```

4 control cell lines.

Toy differential gene expression

I want to compare the treated and control columns. In order to do this, we have to 1. identify/separate out control columns 2. calculate mean value per gene, save as "control.mean" 3. repeat for treated 4. compare

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

Or we can use tidyverse

```r
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
```

```r
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
```

I personally think I like tidyverse better because I really like that %>% filter function.

Q3

How could you add a function that would help? The code above only works if you know how many control values you have because it is hard coded in.

Q4

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

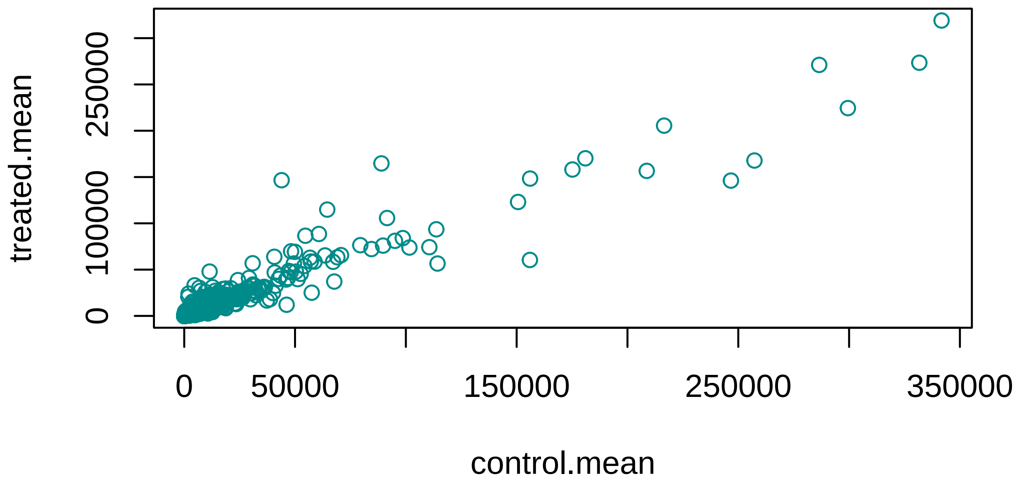
```
mean.counts<-data.frame(control.mean, treated.mean)
colSums(mean.counts)
```

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

Q5a

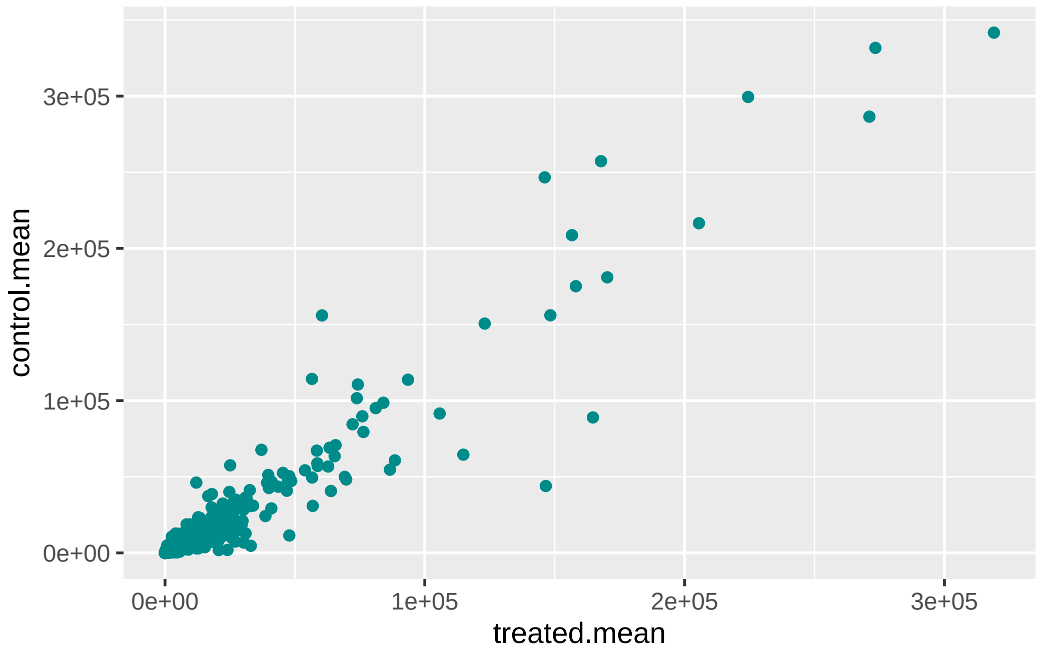
plot of treated samples vs control samples

```
plot(control.mean,treated.mean, col=c("darkcyan"))
```



Now I'm gonna use ggplot :)

```r
library(ggplot2)
ggplot(mean.counts)+aes(treated.mean,control.mean)+
  geom_point(color="darkcyan")
```

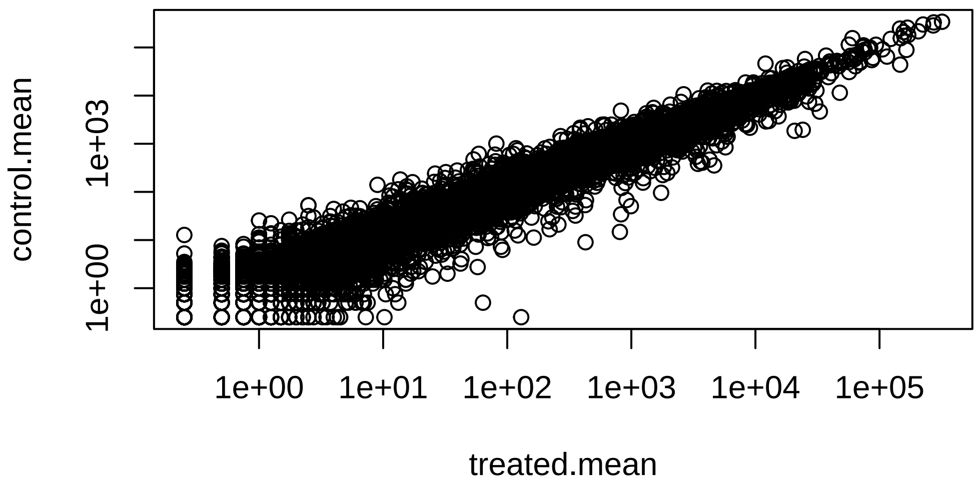


There's like 60k data points, but I cannot hardly see them, so I'll use a log scale to try to see them.

```r
plot(treated.mean,control.mean,log="xy")
```

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

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



Now, we're going to add log2 info because it tends to have better mathematical properties. And then we'll add it to the mean.counts data.frame.

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

```
                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
```

There's several examples with no expression (ie. the NaN, which result from trying to divide by 0, and the -Inf, which results from trying to take the log of a 0.). We should remove those.

```
zero.values<- which(mean.counts[,1:2]==0, arr.ind=TRUE)

to.rm<-unique(zero.values[,1])
```

```
mycounts<-mean.counts[-to.rm,]
```

#Q7

The purpose of the arr.ind I didn't understand initially, so I had to ask Claude. It is used to extract elements from arrays, and without that portion of the code, `zero.values` is listed as values, and with that bit of code, it is listed as data and is something I can actually click on and look at. And `unique` is going to keep us from double counting rows that have 0 in multiple values.

Q9 and Q10

Next, we're going to do up and down regulation.

```
nrow(mycounts)
```

```
[1] 21817
```

```
upreg<-mycounts$log2fc>2
downreg<-mycounts$log2fc< (-2)

sum(upreg)
```

```
[1] 250
```

```
sum(downreg)
```

```
[1] 367
```

There are 21817 genes left that did not have a 0, and there are 250 upregulated and 367 downregulated genes.

Q11

Do I trust these results? We haven't look at any p-values yet, which are usually a huge portion of volcano plots.

Setting up for DESeq

```r
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

Warning: package 'GenomeInfoDb' was built under R version 4.3.2

Loading required package: SummarizedExperiment

Warning: package 'SummarizedExperiment' was built under R version 4.3.2

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
```

```r
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
```

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

```
estimating size factors
```

```
estimating dispersions
```

```
gene-wise dispersion estimates
```

```
mean-dispersion relationship
```

```
final dispersion estimates
```

```
fitting model and testing
```

To get results from this `dds` thing in a usable way, use the DESeq `results()` function

```r
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
```

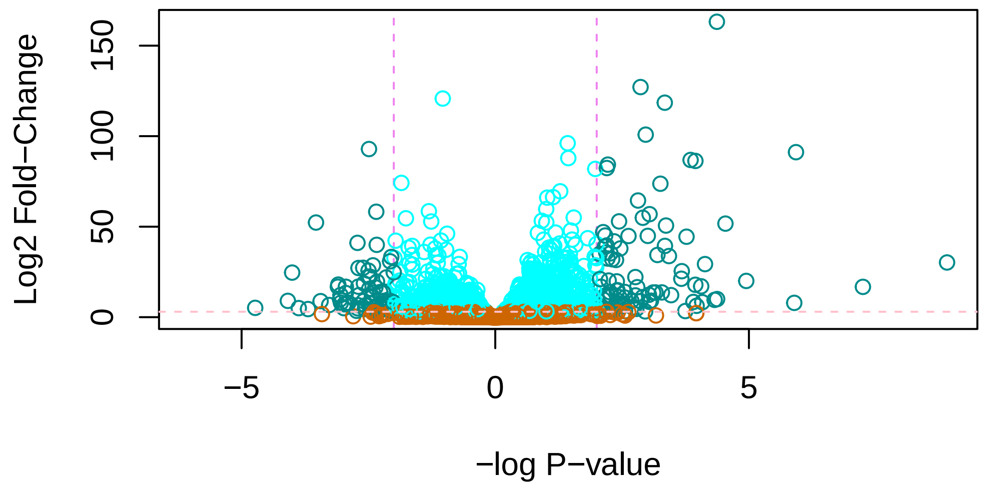
```
ENSG00000000460  87.682625     -0.1471420  0.257007 -0.572521 0.5669691
ENSG00000000938   0.319167     -1.7322890  3.493601 -0.495846 0.6200029
                     padj
                <numeric>
ENSG00000000003  0.163035
ENSG00000000005        NA
ENSG00000000419  0.176032
ENSG00000000457  0.961694
ENSG00000000460  0.815849
ENSG00000000938        NA
```

Data Visualization

Do you remember the volcano plot from the paper discussion today? We're gonna make one.

```
mycolors<- rep("cyan",nrow(res))
mycolors[res$log2FoldChange>2]<-"darkcyan"
mycolors[res$log2FoldChange<(-2)]<-"darkcyan"
mycolors[res$padj > 0.05]<-"darkorange3"
```

```
plot(res$log2FoldChange,-log(res$padj),
     xlab="-log P-value",
     ylab="Log2 Fold-Change",
     col=mycolors,
     abline(v=c(2,-2),col="violet",lty=2)
     )
abline(h=-log(0.05), col="pink", lty=2)
```



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

Adding Annotation Data

We need to translate/map our ensemble IDs into gene names or else we don't know what the f is going on.

```r
library(AnnotationDbi)
```

```

Attaching package: 'AnnotationDbi'

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

    select
```

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

`org.Hs.eg.db` is in a special format, so we have to use a special function 'columns(org.Hs.eg.db") in order to read it.

```r
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"
```

Our current data uses ENSEMBL IDs, and we're gonna map to SYMBOL, via `mapIds()`

```r
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") #this means what do we do if there's multiple valu
```

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

```r
head(res$symbol)
```

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

We're going to add a few more mappings because we want to look at pathways.

Q11

We're gonna run the mapIds 2x more to add Entrez ID and UniProt accession

```r
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
```

```
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
```

```
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
```

We can order results by adjusted p value

```
ordered<-order(res$padj)

head(res[ordered,])
```

```
log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 6 rows and 10 columns
                 baseMean log2FoldChange     lfcSE      stat      pvalue
                <numeric>      <numeric> <numeric> <numeric>   <numeric>
ENSG00000152583   954.771        4.36836 0.2371268   18.4220 8.74490e-76
ENSG00000179094   743.253        2.86389 0.1755693   16.3120 8.10784e-60
ENSG00000116584  2277.913       -1.03470 0.0650984  -15.8944 6.92855e-57
ENSG00000189221  2383.754        3.34154 0.2124058   15.7319 9.14433e-56
ENSG00000120129  3440.704        2.96521 0.2036951   14.5571 5.26424e-48
```

```
ENSG00000148175 13493.920        1.42717 0.1003890   14.2164 7.25128e-46
                       padj      symbol      entrez     uniprot
                  <numeric> <character> <character> <character>
ENSG00000152583 1.32441e-71     SPARCL1        8404  A0A024RDE1
ENSG00000179094 6.13966e-56        PER1        5187      O15534
ENSG00000116584 3.49776e-53     ARHGEF2        9181      Q92974
ENSG00000189221 3.46227e-52        MAOA        4128      P21397
ENSG00000120129 1.59454e-44       DUSP1        1843      B4DU40
ENSG00000148175 1.83034e-42        STOM        2040      F8VSL7
                              genename
                           <character>
ENSG00000152583           SPARC like 1
ENSG00000179094 period circadian reg..
ENSG00000116584 Rho/Rac guanine nucl..
ENSG00000189221    monoamine oxidase A
ENSG00000120129 dual specificity pho..
ENSG00000148175               stomatin
```

Pathway Analysis

To do this, we're gonna install a few more packages.

```r
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).
##############################################################################
```

```r
library(gage)
library(gageData)
```

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

gage needs a vector it won't know what to do with DESeq stuff, and gage speaks ENTREZ, so names have to be in ENTREZ format

```r
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
```

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

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

Next, look at the first 3 less than genes.

```r
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
```

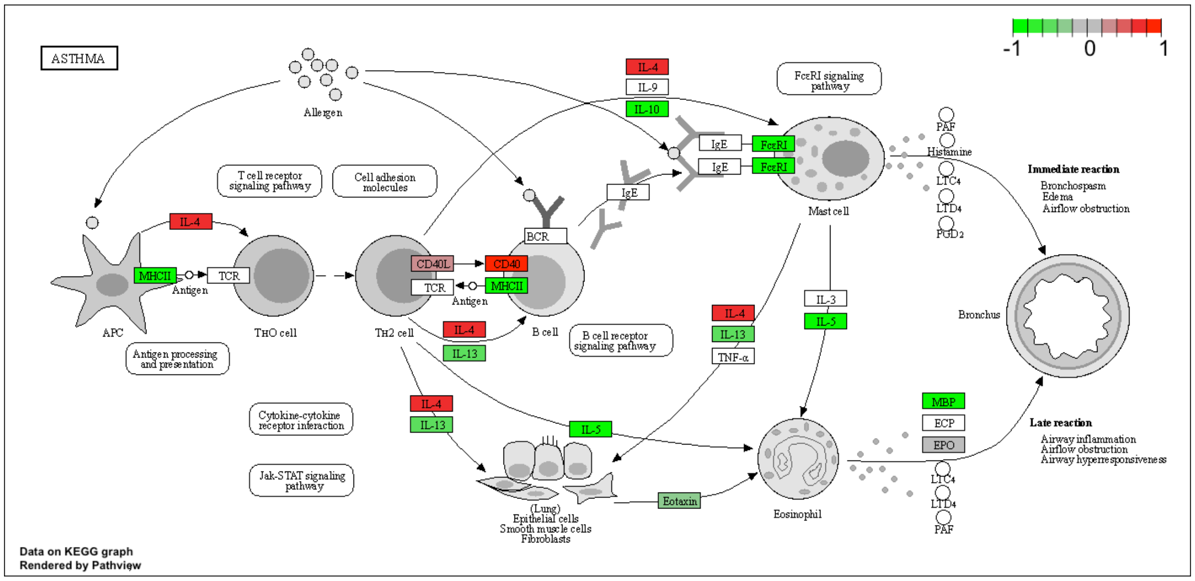
hsa05332 is the kegg identifier. We're gonna look at asthma because that's what he used to work on. So now we'll make a pathway viewer.

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

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

```
Info: Working in directory /Users/savannahbogus/Documents/Classes/BGGN 213 Bioinformatics/20
```

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



ASTHMA
-1 0 1
Allergen
IL-4
IL-9
IL-10
FcεRI signaling pathway
IgE
FcεRI
Mast cell
PAF
Histamine
LTC4
LTD4
PGD2
Immediate reaction
Bronchospasm
Edema
Airflow obstruction
T cell receptor signaling pathway
Cell adhesion molecules
MHCII
Antigen
TCR
APC
TH0 cell
CD40L
CD40
TH2 cell
B cell
BCR
B cell receptor signaling pathway
Antigen processing and presentation
IL-13
IL-3
IL-5
TNF-α
Bronchus
Cytokine-cytokine receptor interaction
Jak-STAT signaling pathway
MBP
ECP
EPO
Late reaction
Airway inflammation
Airflow obstruction
Airway hyperresponsiveness
Eotaxin
Eosinophil
(Lung) Epithelial cells Smooth muscle cells Fibroblasts
Data on KEGG graph
Rendered by Pathview