

Class 12

Jimmi Nguyen

Transcriptomics and the analysis of RNA-Seq data

install the core Bioconductor packages

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

```
Attaching package: 'S4Vectors'
```

```
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
```

```
Attaching package: 'MatrixGenerics'
```

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

```
colAlls, colAnyNAs, colAnyNs, colAvgsPerRowSet, colCollapse,  
colCounts, colCummакс, colCummins, colCumprod, 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, rowAnyNs, 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
```

This week we are looking at differential expression analysis.

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

Import/read the Data Himes et al.

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

Let's have a wee peak at this data

```
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
```

Sanity check on corespondence of counts and metadata

```
all(metadata$id ==
  colnames(counts))
```

```
[1] TRUE
```

Q1. How many genes are in this dataset?

There are 38694 genes in this dataset.

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

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

There are 4 control cell lines in this dataset.

Extract and summarize the control samples

To find out where the control samples are we need the metadata

```
control = metadata[metadata$dex == "control",]
control.counts = counts [ ,control$id]

control.mean = rowMeans(control.counts)
head(control.mean)
```

```
ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460
         900.75          0.00        520.50        339.75         97.25
ENSG000000000938
         0.75
```

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

We will make it more robust by not storing either metadata/counts into addition variables and refining it into control.means.

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)

Extract and summarize the treated samples

```
treated = metadata[metadata$dex == "treated",]  
treated.counts = counts[,treated$id]  
treated.mean = rowMeans(treated.counts)
```

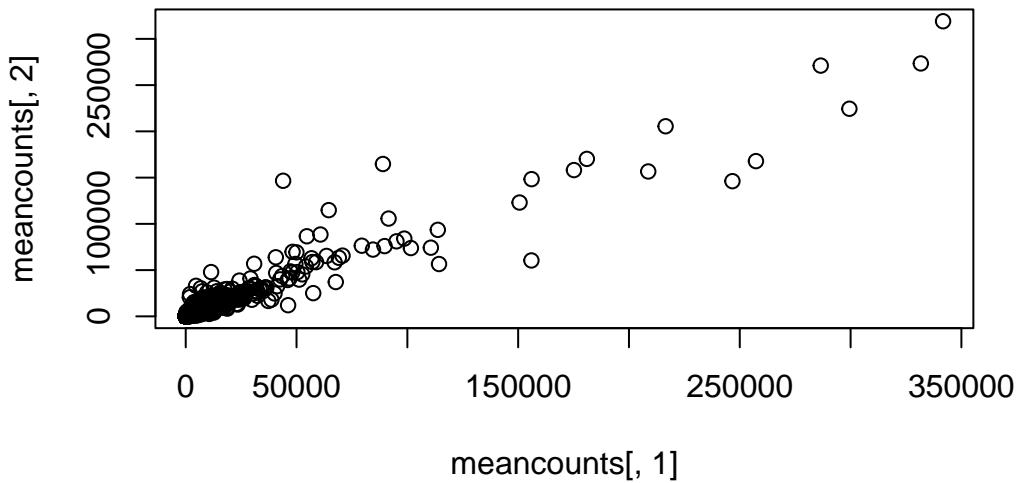
Store these results together in a new data frame called meancounts

```
meancounts = data.frame(control.mean, treated.mean)
```

Lets make a plot to explore thee results a little

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[,1], meancounts[,2])
```

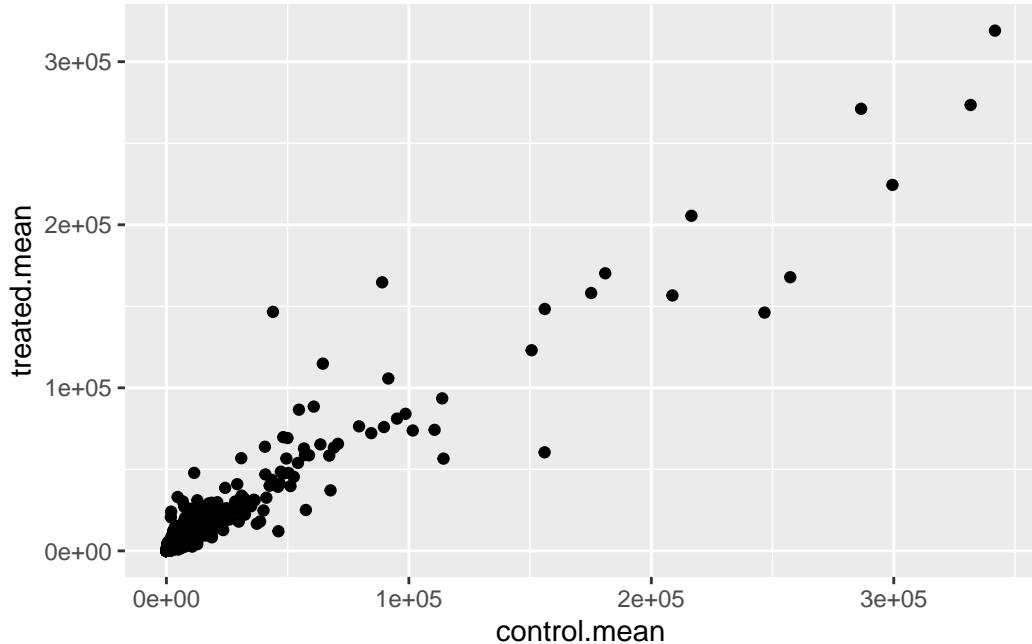


Q5 (b). You could also use the ggplot2 package to make this figure producing the plot below. What geom_?() function would you use for this plot?

We will use `geom_point()` function to produce a scatter plot.

```
library(ggplot2)

ggplot(meancounts) +
  aes(control.mean, treated.mean) +
  geom_point()
```



We will make a log-log plot to draw out this skewed data and see what is going on.

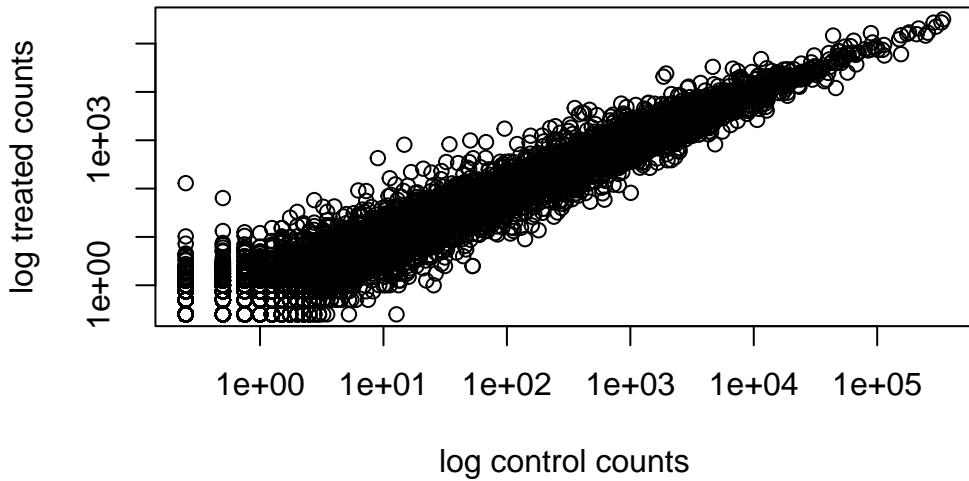
Q6. Try plotting both axes on a log scale. What is the argument to plot() that allows you to do this?

`log="xy"` argument allows us to plot both axes on a log scale.

```
plot(meancounts[,1], meancounts[,2], log="xy", xlab="log control counts", ylab="log treated counts")
```

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 often log2 transformations when dealing with this sort of data.

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

```
[1] 0
```

```
log2(40/20)
```

```
[1] 1
```

```
log2(20/40)
```

```
[1] -1
```

```
log2(80/20)
```

```
[1] 2
```

This log2 transformation has this nice property where if there is no change the log2 value will be zero and if it double the log2 value will be 1 and if halved it will be -1.

So lets add a log2 fold change column to our results so far

```
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

We need to get rid of zero count genes that we can not say anything about

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

arr.ind function tells us which columns there are zeros in. We called the unique function because it states how many zeros are in each row/gene without its id attached to it.

how many genes are remaining?

```
nrow(mycounts)
```

```
[1] 21817
```

use fold change to see up and down regulated genes.

A common threshold used for calling something differentially expressed is a $\log_2(\text{foldchange})$ 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?

There are 250 up regulated genes.

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

There are 367 down regulated genes.

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

No, Where is the statistical significance.

DESeq2 analysis

Let's do this the right way. DESeq2 is an R package specifically for analyzing count-based NGS data like RNA-seq. It is available from Bioconductor. Bioconductor is a project to provide tools for analyzing high-throughput genomic data including RNA-seq, chIP-seq and arrays.

```
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},
}
```

```
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

```
dds = DESeq(dds)
```

estimating size factors

estimating dispersions

gene-wise dispersion estimates

mean-dispersion relationship

final dispersion estimates

fitting model and testing

```

res = results(dds)
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>
ENSG000000000003  747.1942 -0.3507030  0.168246 -2.084470 0.0371175
ENSG000000000005  0.00000   NA        NA        NA        NA
ENSG000000000419  520.1342  0.2061078  0.101059  2.039475 0.0414026
ENSG000000000457  322.6648  0.0245269  0.145145  0.168982 0.8658106
ENSG000000000460  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>
ENSG000000000003  0.163035
ENSG000000000005  NA
ENSG000000000419  0.176032
ENSG000000000457  0.961694
ENSG000000000460  0.815849
...
...
ENSG00000283115  NA
ENSG00000283116  NA
ENSG00000283119  NA
ENSG00000283120  NA
ENSG00000283123  NA

```

We can get some basic summary tallies using the `summary()` function

```
summary(res, alpha=0.05)
```

```

out of 25258 with nonzero total read count
adjusted p-value < 0.05
LFC > 0 (up)      : 1242, 4.9%

```

```

LFC < 0 (down)      : 939, 3.7%
outliers [1]        : 142, 0.56%
low counts [2]       : 9971, 39%
(mean count < 10)
[1] see 'cooksCutoff' argument of ?results
[2] see 'independentFiltering' argument of ?results

```

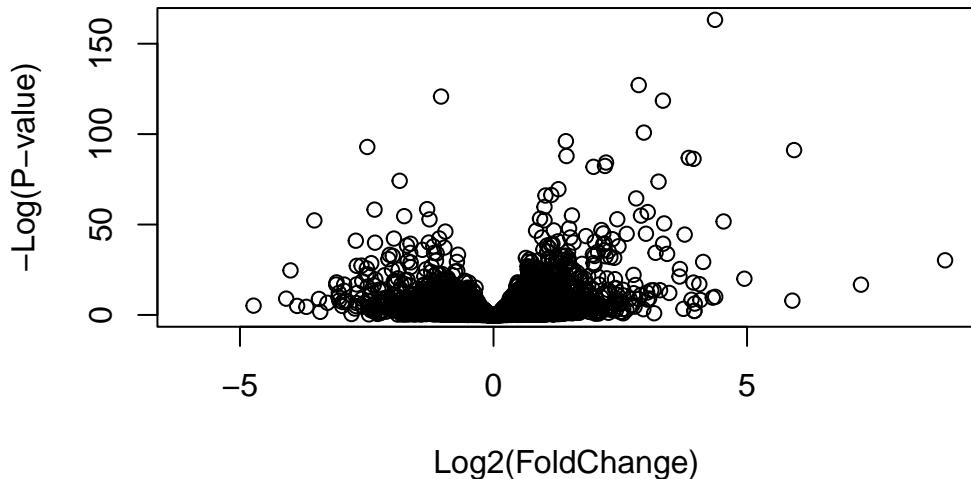
Volcano plot

Make a summary plot of our results.

```

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

```



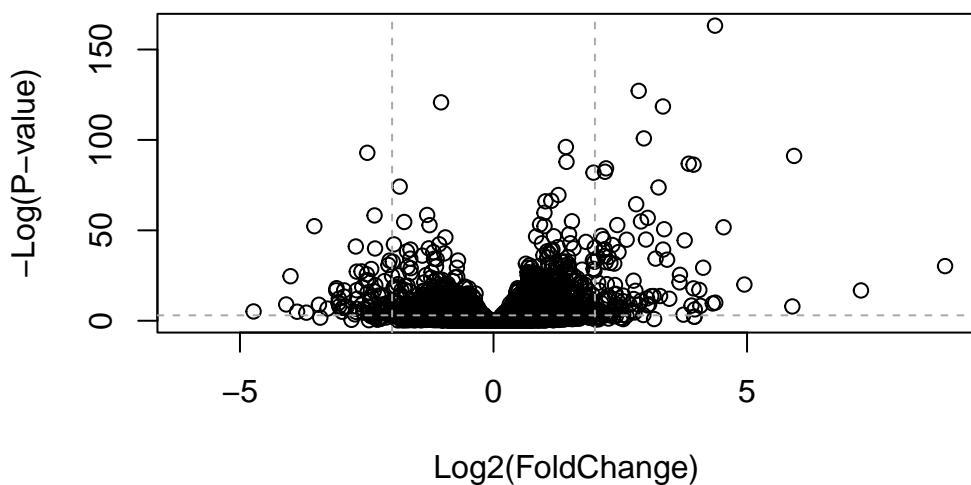
To make this more useful we can add some guidelines (with the abline() function) and color (with a custom color vector) highlighting genes that have $\text{padj} < 0.05$ and the absolute $\text{log2FoldChange} > 2$.

```

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

# Add some cut-off lines
abline(v=c(-2,2), col="darkgray", lty=2)
abline(h=-log(0.05), col="darkgray", lty=2)

```



To color the points we will setup a custom color vector indicating transcripts with large fold change and significant differences between conditions:

```

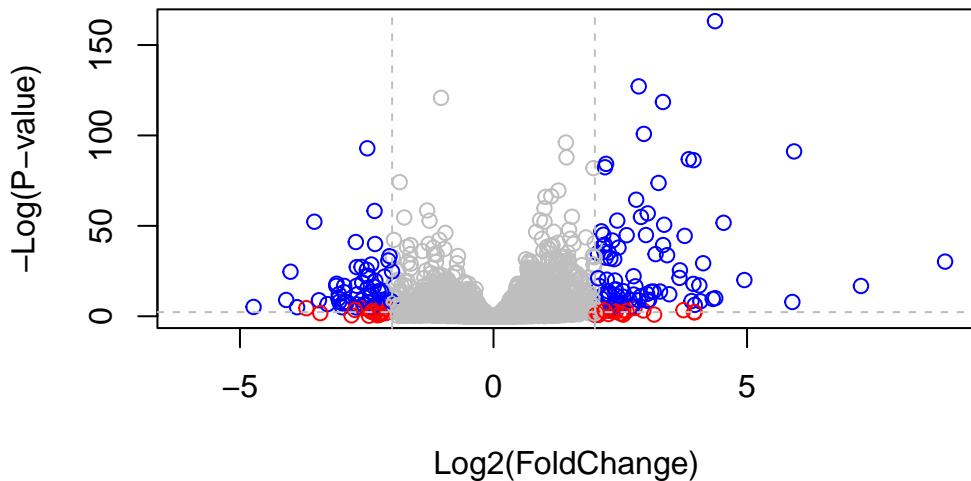
# Setup our custom point color vector
mycols <- rep("gray", nrow(res))
mycols[ abs(res$log2FoldChange) > 2 ] <- "red"

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

# Volcano plot with custom colors
plot( res$log2FoldChange, -log(res$padj),
      col=mycols, ylab="-Log(P-value)", xlab="Log2(FoldChange)" )

```

```
# Cut-off lines  
abline(v=c(-2,2), col="gray", lty=2)  
abline(h=-log(0.1), col="gray", lty=2)
```



Finish for today by saveing our results

```
write.csv(res, file="DESeq2_results.csv")
```

Section 5: Gene Annotation

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

Loading required package: AnnotationDbi

```
library(AnnotationDbi)
```

Look at what types of IDs I can translate between from the `org.Hs.eg.db` packages with the `column` function.

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

```
[1] "ACNUM"      "ALIAS"       "ENSEMBL"      "ENSEMLPROT"   "ENSEMLTRANS"  
[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"
```

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

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

and do the same for ENTREZID and GENENAME

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

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

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

```
'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>
ENSG000000000003 747.194195 -0.3507030 0.168246 -2.084470 0.0371175
ENSG000000000005 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      entrez      genename
      <numeric> <character> <character> <character>
ENSG000000000003 0.163035   TSPAN6      7105      tetraspanin 6
ENSG000000000005  NA        TNMD      64102      tenomodulin
ENSG00000000419 0.176032   DPM1      8813      dolichyl-phosphate m..
ENSG00000000457 0.961694   SCYL3      57147      SCY1 like pseudokina..
ENSG00000000460 0.815849   C1orf112    55732      chromosome 1 open re..
ENSG00000000938  NA        FGR       2268      FGR proto-oncogene, ..

```

Pathway Analysis

We will finish this lab with a quick pathway analysis. Here we play with just the **GAGE package** (which stands for generally applicable gene set enrichment), to do **KEGG pathway enrichment analysis**

```
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, 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"

```

The main `gage()` function requires a named vector of fold changes, where the names of the values are the Entrez gene IDs.

```
c(barry = 4, clair = 3, chandra = 2)
```

```
barry    clair chandra
4        3       2
```

```

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

```

7105	64102	8813	57147
-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)
```

Now lets look at the object returned from `gage()`, i.e. our results here:

```
attributes(keggres)
```

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

Lets look at the first few down (less) pathway results:

```
# 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

Let's pull up the highlighted pathways and show our differentially expressed genes on the pathway. I will use the "hsa" KEGG id to get the pathway from KEGG and my `foldchange` vector to show my genes.

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

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

```
Info: Working in directory C:/Users/jimmi/OneDrive/Desktop/bimm 143/Class_12
```

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

Put this into my document.

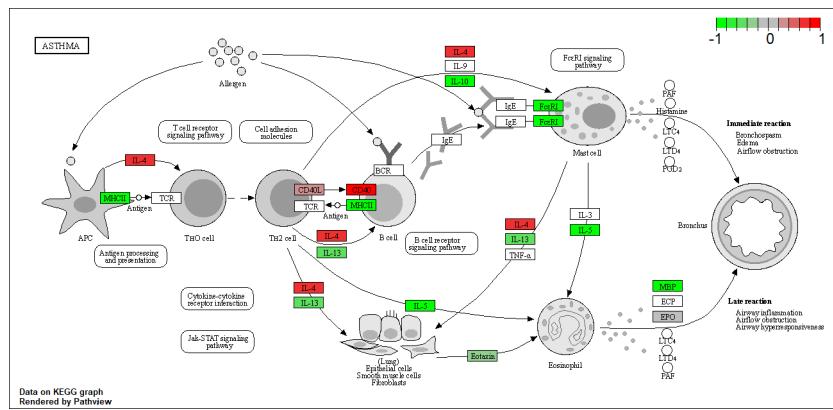


Figure 1: The Asthma pathway with my highlighted differentially expressed genes in color