

Class 12: Transcriptomics & RNA-Seq

Kira

In this session we will read and explore the gene expression data from this experiment using base R functions and then perform a detailed analysis with the DESeq2 package from Bioconductor.

2. Import countData and colData

To begin our RNA-Seq analysis, we first need to import our data:

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

	SRR1039508	SRR1039509	SRR1039512	SRR1039513	SRR1039516
ENSG000000000003	723	486	904	445	1170
ENSG000000000005	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		
ENSG000000000003	1097	806	604		
ENSG000000000005	0	0	0		
ENSG00000000419	781	417	509		
ENSG00000000457	447	330	324		
ENSG00000000460	94	102	74		
ENSG00000000938	0	0	0		

```
(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
7 SRR1039520 control    N061011 GSM1275874
8 SRR1039521 treated    N061011 GSM1275875
```

Q1. How many genes are in this dataset?

```
nrow(counts)
```

```
[1] 38694
```

There are 38694 genes in this dataset.

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

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

```
control treated
        4       4
```

There are 4 ‘control’ cell lines.

We also want to ensure that the id column of the metadata matches the order of the columns in countData.

We can use the `all()` function to check if all the inputs are TRUE:

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

```
[1] TRUE
```

The column names of the count data matches the id column of the metadata, so we are all set to start our analysis.

3. Toy differential gene expression

Let's first extract our counts for the control samples so that we can compare this to the counts for the treated samples (i.e with drug).

```
# Extracting control information into a table
control inds <- metadata$dex == "control"
control ids <- metadata$id[control inds]
control counts <- counts[, control ids]
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

I want a single summary counts value for each gene in the control experiments. To start, I will take the average:

```
# Finding an average count value for each gene
control mean <- rowMeans(control counts)
head(control mean)
```

ENSG000000000003	ENSG000000000005	ENSG000000000419	ENSG000000000457	ENSG000000000460
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?

It is more robust to find the mean using either the `rowMeans()` function or the `apply()` function so that it can work for any amount of samples (i.e if the number of samples changed).

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`)

```
# Extracting control information into a table
treated inds <- metadata$dex == "treated"
treated ids <- metadata$id[treated inds]
```

```
treated.counts <- counts[, treated.ids]
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

```
# Finding an average count value for each gene
treated.mean <- rowMeans(treated.counts)
head(treated.mean)
```

ENSG000000000003	658.00	ENSG000000000005	0.00	ENSG000000000419	546.00	ENSG000000000457	316.50	ENSG000000000460	78.75
ENSG000000000938	0.00								

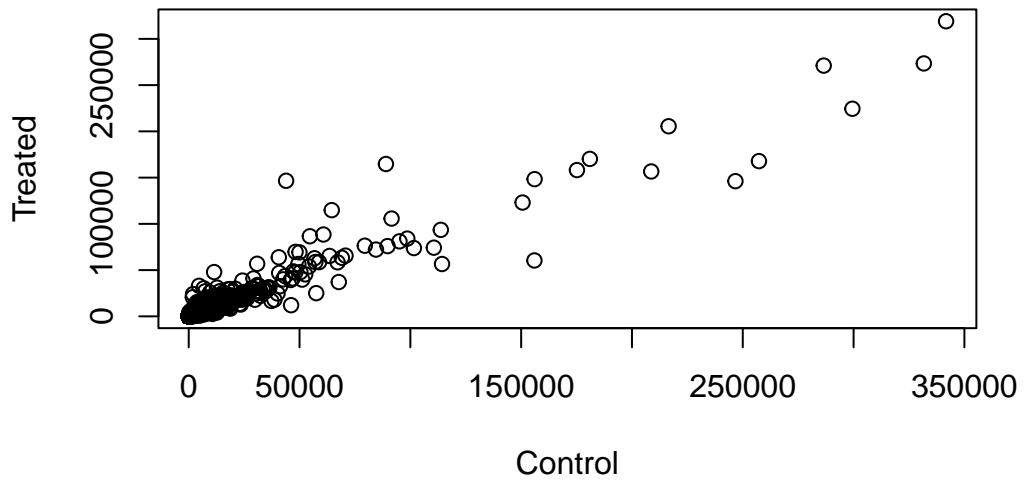
```
meancounts <- data.frame(control.mean, treated.mean)

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.

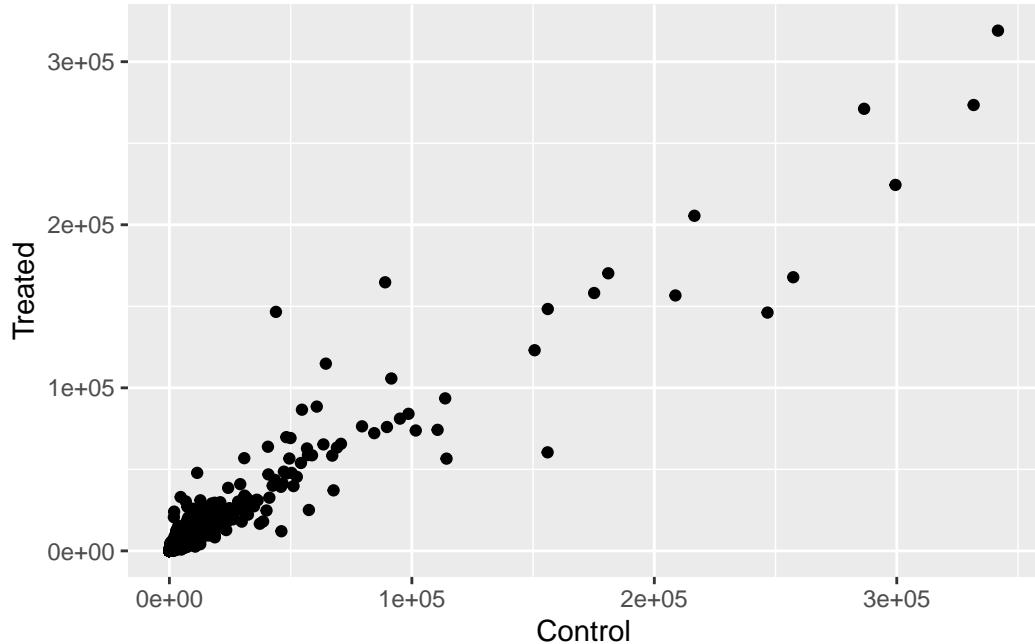
```
plot(meancounts$control.mean, meancounts$treated.mean, xlab="Control", ylab="Treated")
```



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 would be useful for this plot:

```
library(ggplot2)
ggplot(meancounts) + aes(control.mean,treated.mean) + geom_point() + xlab("Control") + yla
```



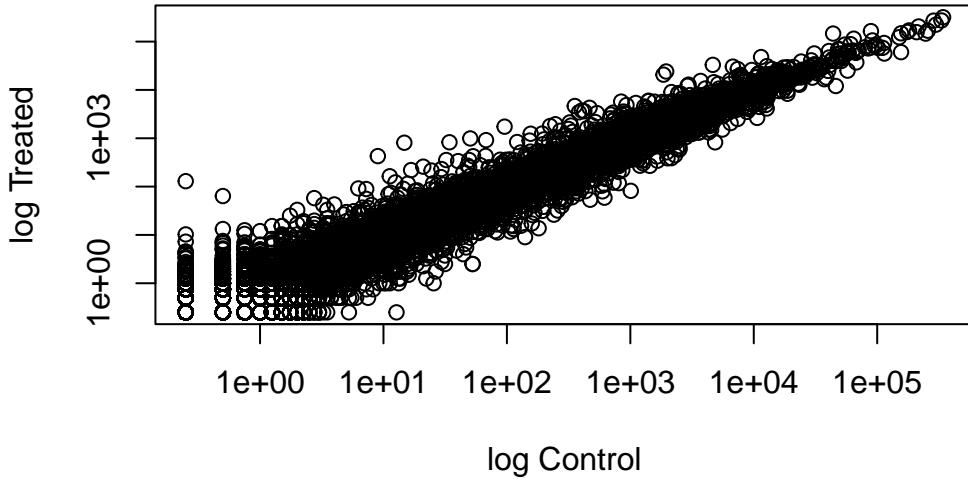
Q6. Try plotting both axes on a log scale. What is the argument to `plot()` that allows you to do this?

We can use the `log.scale` argument to plot both axes on a log scale using base R plotting:

```
plot(meancounts$control.mean, meancounts$treated.mean, log="yx", xlab="log Control", ylab="log Treated")
```

```
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
```



The most useful and most straightforward to understand is log2 transformation. We will add a “log2 fold-change” to our data frame by creating a new column:

```
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
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

Hmmm ... we need to get rid of the genes where we have no count data as taking the log2 of these 0 counts does not tell us anything.

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

```
nrow(mycounts)
```

[1] 21817

```
# Another method to remove zero values for gene counts

#zero.vals <- which(meancounts[,1:2]==0, arr.ind=TRUE)

#to.rm <- unique(zero.vals[,1])
#mycounts <- meancounts[-to.rm,]
#head(mycounts)
```

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 arr.ind argument specifies that the array indices should be returned. We would use the unique() function to gather the zero values into a vector that we then can exclude from the counts dataset.

```
up.ind <- sum(mycounts$log2fc >= +2)
down.ind <- sum(mycounts$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?

```
up.ind
```

[1] 314

There are 314 genes that are up regulated at the greater than 2 fc level.

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

```
down.ind
```

```
[1] 485
```

There are 485 genes that are down regulated at the greater than 2 fc level.

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

We can't really trust these results yet because we haven't determined if the difference in mean and fold change values are statistically significant or not.

4. DESeq2 analysis

```
library(DESeq2)
```

Like most bioconductor packages, DESeq2 wants its input and output in a very specific format:

```
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 main DESeq function is called `DESeq()`:

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

```
estimating size factors
```

```
estimating dispersions
```

```
gene-wise dispersion estimates
```

```
mean-dispersion relationship
```

```
final dispersion estimates
```

```
fitting model and testing
```

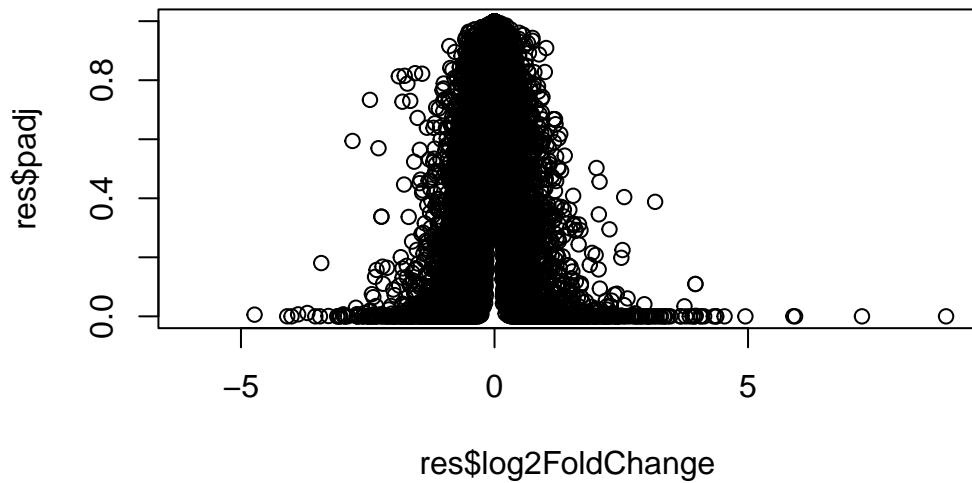
```
res <- results(dds)
head(res)
```

```
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
```

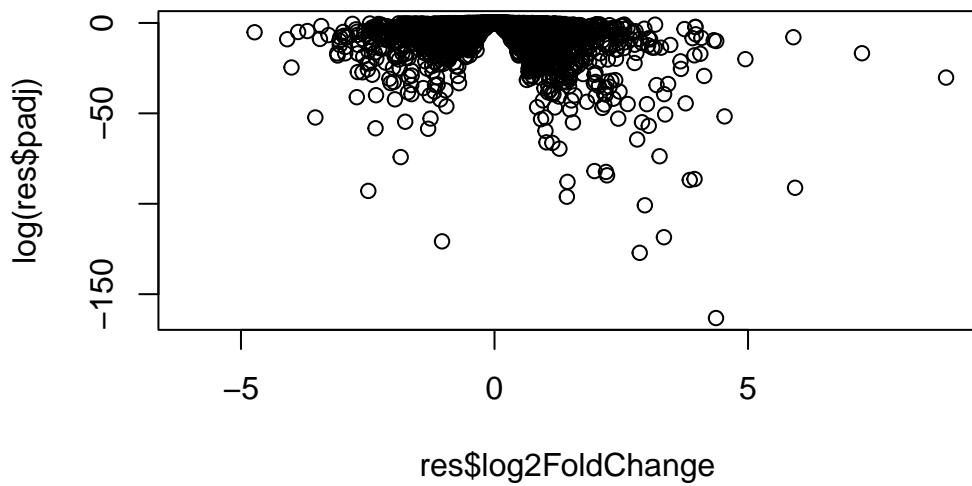
6. Data Visualization: Volcano Plot

These summary figures are frequently used to highlight the proportion of genes that are both significantly regulated and display a high fold change (this will keep both our inner biologist and statistician happy).

```
plot( res$log2FoldChange, res$padj)
```

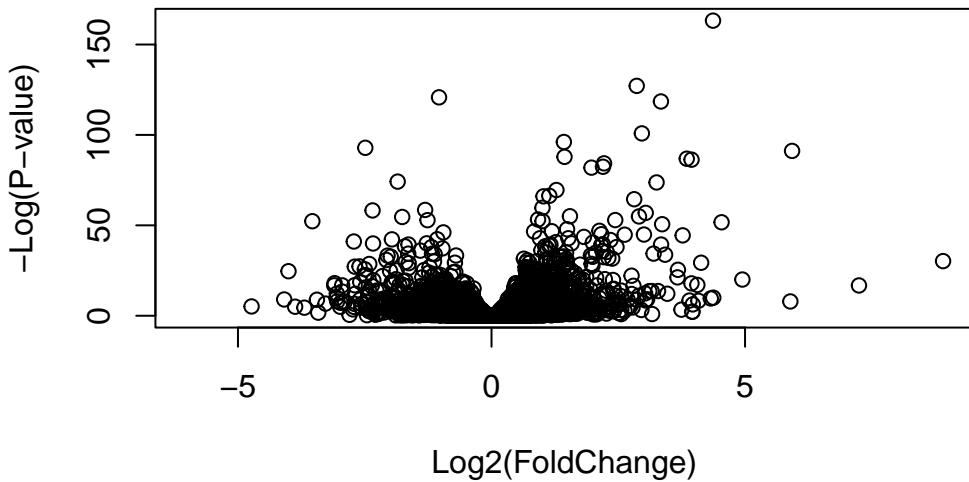


```
plot( res$log2FoldChange, log(res$padj))
```



We want to flip the y-axis so that the values we're interested in (i.e the low p-value or high log(p-value)) are at the top of the axis. We can improve this plot by taking the negative log of the y-axis (padj):

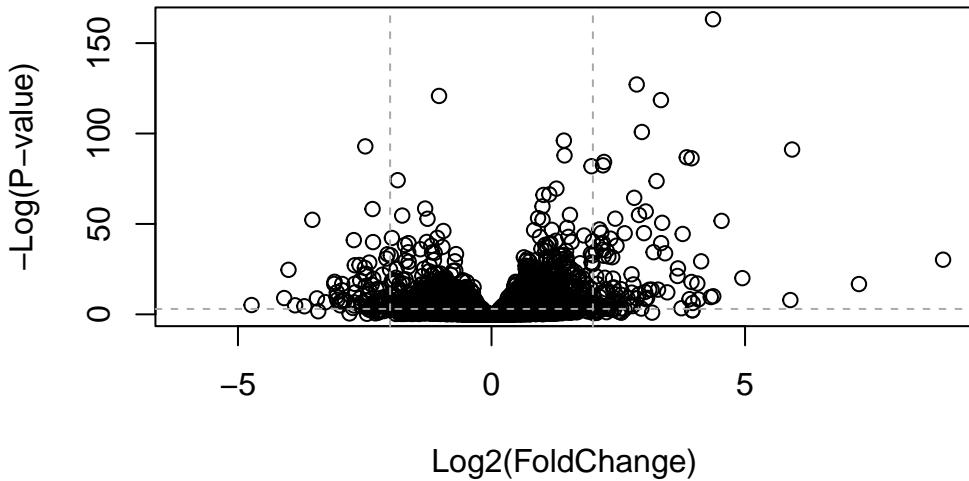
```
plot( res$log2FoldChange, -log(res$padj),
      xlab="Log2(FoldChange)",
      ylab="-Log(P-value)")
```



To make this more useful we can add some guidelines and color 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)
```



```

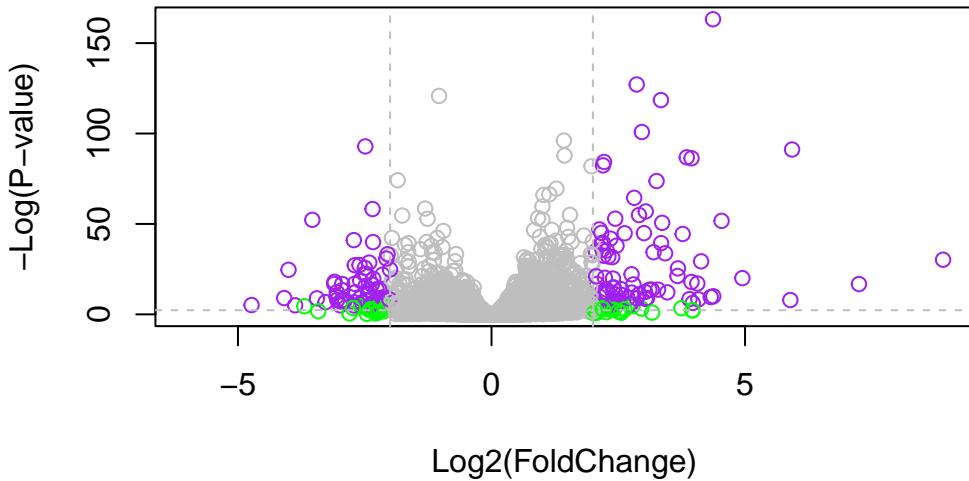
# Setup our custom point color vector
mycols <- rep("gray", nrow(res))
mycols[ abs(res$log2FoldChange) > 2 ] <- "green"

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

# 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)

```



These colored transcripts have both a large fold change and a significant difference between conditions.

5. Gene Annotation

We will use one of Bioconductor's main annotation packages to help with mapping between various ID schemes. Here we load the `AnnotationDbi` package and the annotation data package for humans `org.Hs.eg.db`.

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

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"
```

We can use the `mapIds()` function to add individual columns to our results table.

```
res$symbol <- mapIds(org.Hs.eg.db,  
                      keys=row.names(res), # Our gene names  
                      keytype="ENSEMBL",   # The format of our gene names  
                      column="SYMBOL")    # The new format we want to add  
  
'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 7 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      symbol  
  <numeric> <character>  
ENSG000000000003 0.163035    TSPAN6  
ENSG000000000005  NA          TNMD  
ENSG000000000419 0.176032    DPM1  
ENSG000000000457 0.961694    SCYL3  
ENSG000000000460 0.815849    C1orf112  
ENSG000000000938  NA          FGR
```

Q11. Run the `mapIds()` function two more times to add the Entrez ID and UniProt accession and GENENAME as new columns called `res$entrez`, `res$uniprot` and `res$genename`.

```
res$entrez <- mapIds(org.Hs.eg.db,  
                      keys=row.names(res),
```

```

keytype="ENSEMBL",
column="ENTREZID")

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

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

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

res$uniprot <- mapIds(org.Hs.eg.db,
                      keys=row.names(res),
                      keytype="ENSEMBL",
                      column="UNIPROT")

'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 10 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      symbol      entrez      genericname
  <numeric> <character> <character>      <character>
ENSG000000000003  0.163035    TSPAN6       7105      tetraspanin 6
ENSG000000000005    NA         TNMD      64102      tenomodulin
ENSG000000000419  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, ..
                           uniprot
                           <character>
ENSG00000000003  AOA024RCI0
ENSG00000000005  Q9H2S6
ENSG00000000419  D60762
ENSG00000000457  Q8IZE3
ENSG00000000460  AOA024R922
ENSG00000000938  P09769

```

7. Pathway Analysis

Here we will use the GAGE package (which stands for Generally Applicable Gene set Enrichment), to do KEGG pathway enrichment analysis on our RNA-seq based differential expression results.

```

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.

```
# Linking Entrez ID to log 2 fold change value
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
```

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

Now we can look at the object returned from `gage()`:

```
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

Let's look at the highlighted pathways and display our differentially expressed genes:

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

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

```
Info: Working in directory /Users/kirbear/Desktop/BIMM 143/class12
```

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

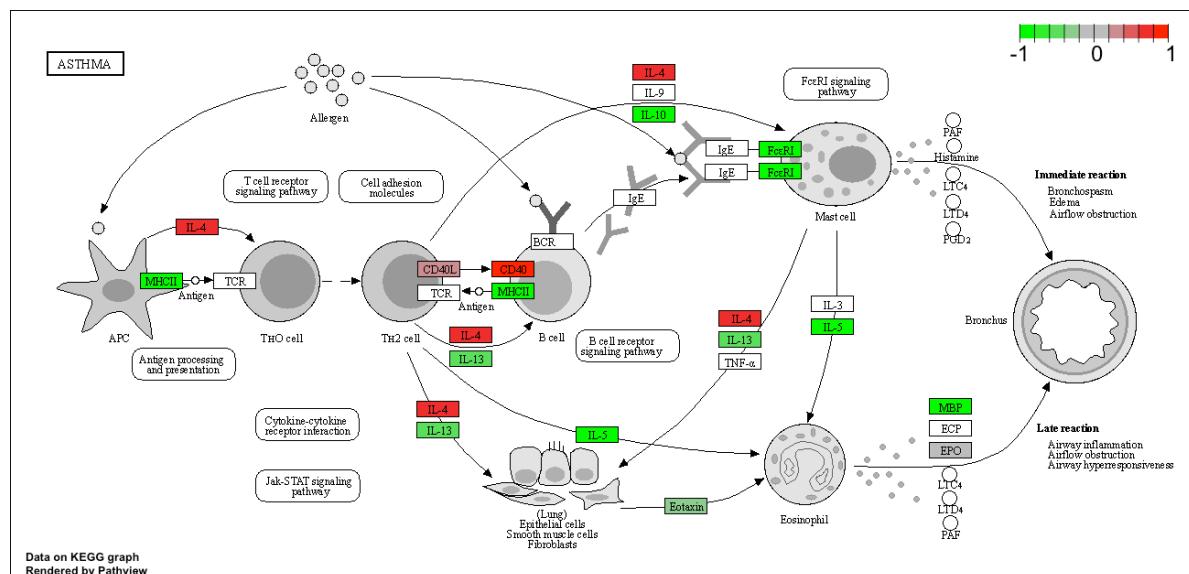


Figure 1: The Asthma pathway with my highlighted differentially expressed genes in color