

class13: Transcriptomics & the analysis of RNA-seq Data

Anna Waters (PID: A16271985)

In today's class we will explore and analyze data from a published RNA-seq experiment where airway smooth muscle cells were treated with dexamethasone, a synthetic glucocorticoid steroid with anti-inflammatory effects (Himes et al. 2014).

3. Data Import

We have two input files, so called the “countData” and the “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
```

Data Explore

Q1. How many genes are in this dataset?

```
nrow(counts)
```

```
[1] 38694
```

There are 38694 genes in this data set.

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

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

```
[1] 4
```

There are 4 control cell lines.

##4. Toy differential gene expression

Time to do some analysis. We have 4 control and 4 treated samples/experiments/columns.

Make sure the meta data id column matches the columns in our count data.

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

```
[1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

To check that all elements of a vector are TRUE, we can use the `all()` function.

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

```
[1] TRUE
```

To start I will calculate the `control.mean` and the `treated.mean` values and compare them.

- Identify and extract the `control` only columns
- Determine the mean value for each gene (i.e. row)
- Do the same for `treated`.

```
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
```

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

The `apply()` and `mean()` function could be useful here to make the approach more robust. I would use the `apply()` and `mean()` function rather than calculate by hand.

```
# Finding which samples are control
control inds <- metadata$dex=="control"
#Isolating the control only files using select
control.count <- counts[, control inds]
# using apply to find the mean across every row for the 4 controls
control.mean <- apply(control.count, 1, mean)
head(control.mean)
```

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

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)

Do the same for treated:

```
# Finding which samples are control
treated inds <- metadata$dex=="treated"
#Isolating the control only files using select
treated.count <- counts[, treated inds]
# using apply to find the mean across every row for the 4 controls
treated.mean <- apply(treated.count, 1, mean)
head(treated.mean)
```

```
ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460
       658.00          0.00        546.00        316.50        78.75
ENSG000000000938
       0.00
```

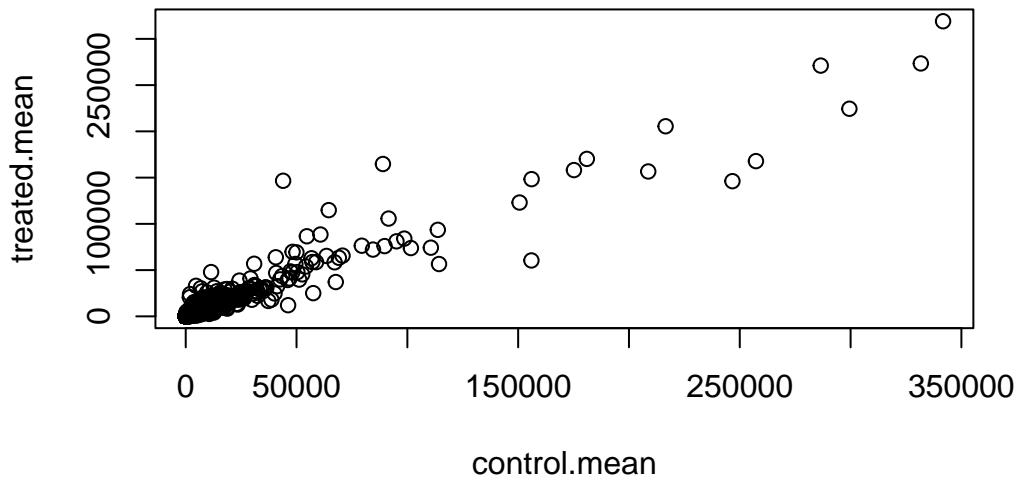
For book-keeping:

```
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.

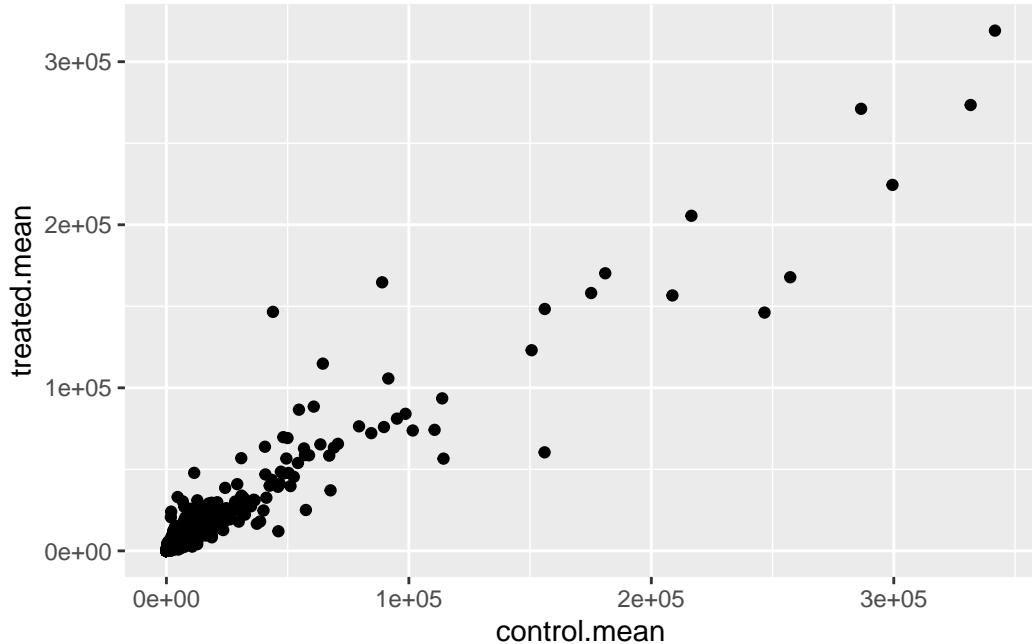
Quick view of this data:

```
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?

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



Geom_point() can be used to plot the two with ggplot.

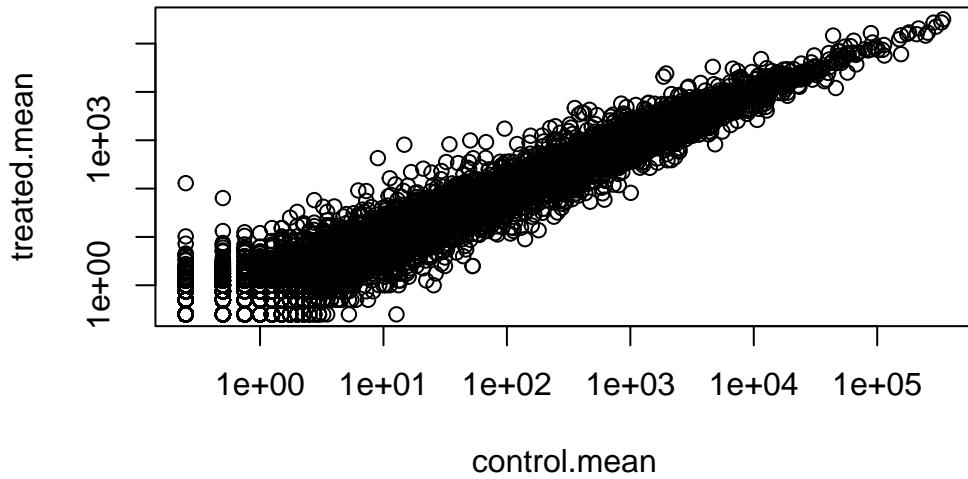
The data is heavily skewed and over plotted so we want to use a log transformation.

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 <= 0 omitted from logarithmic plot

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



The `log= "xy"` can be used to do the transformation to see the points.

I want to compare the treated and the control values here and we will use. Fold change in log2 units to do this. `log2(Treated/Control)`

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

No difference

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

```
[1] 0
```

Doubling in the treated:

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

```
[1] 1
```

Halving in the treated

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

```
[1] -1
```

A common rule of thumb cut-off for calling a gene differentially expressed is a log2 fold change value of either greater than +2 for “upregulated” or less than -2 for “down regulated”.

```
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

We first need to remove the zero count genes - as we can't say anything about these genes anyway and their division of log values are messing things up (divide by zero) or the -infinity log problems.

```
# if it is more than 0, one of the values equaled 0  
to.rm.ind <- rowSums(meancounts[,1:2]==0) > 0  
# save everything but the rows that should be removed  
mycounts <- meancounts[!to.rm.ind, ]
```

Q. How many genes do we have left that we can say something about (i.e. they don't have any zero counts)

```
nrow(mycounts)
```

```
[1] 21817
```

We have 21817 genes with transcripts.

```
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
```

There are 250 genes that are up regulated.

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
```

There are 367 down regulated genes.

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

No, I don't trust these results because we can't tell if the difference in the means is significant. We don't know the range of the controls or variability between samples. We want genes that are significantly different with big expression changes. We need statistics.

```
##DESeq
```

Let's do this properly with the help of the DESeq2 package

```
library(DESeq2)
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},  
}
```

We have to use a specific data object for working with DESeq.

```
dds <- DESeqDataSetFromMatrix(counts, 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
```

Run our main analysis with the `DESeq()`:

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

```
estimating size factors
```

```
estimating dispersions
```

```
gene-wise dispersion estimates
```

```
mean-dispersion relationship
```

```
final dispersion estimates
```

```
fitting model and testing
```

To get the results out of our `dds` object, we can use the `DESeq` function called `results()`

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

```

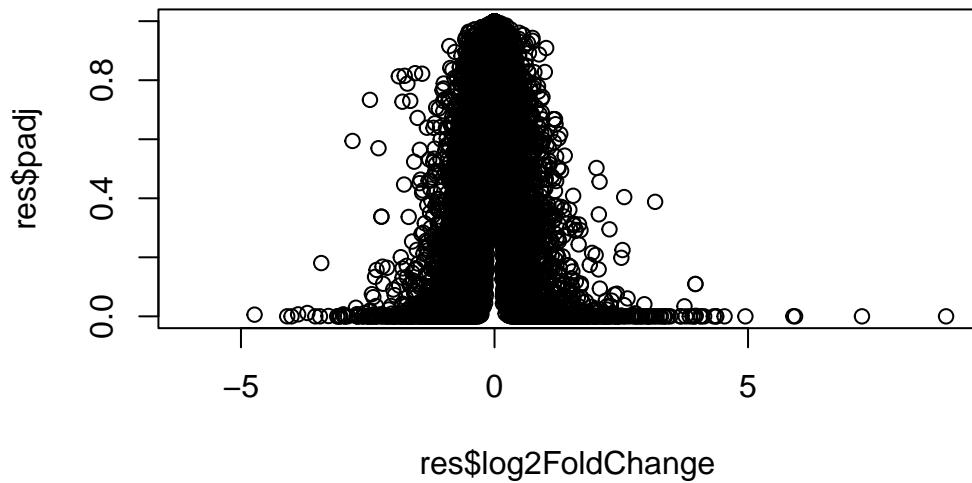
log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 6 rows and 6 columns
  baseMean log2FoldChange    lfcSE      stat   pvalue
  <numeric>     <numeric> <numeric> <numeric> <numeric>
ENSG000000000003 747.194195 -0.3507030 0.168246 -2.084470 0.0371175
ENSG000000000005 0.000000      NA       NA       NA       NA
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>
ENSG000000000003 0.163035
ENSG000000000005      NA
ENSG00000000419 0.176032
ENSG00000000457 0.961694
ENSG00000000460 0.815849
ENSG00000000938      NA

```

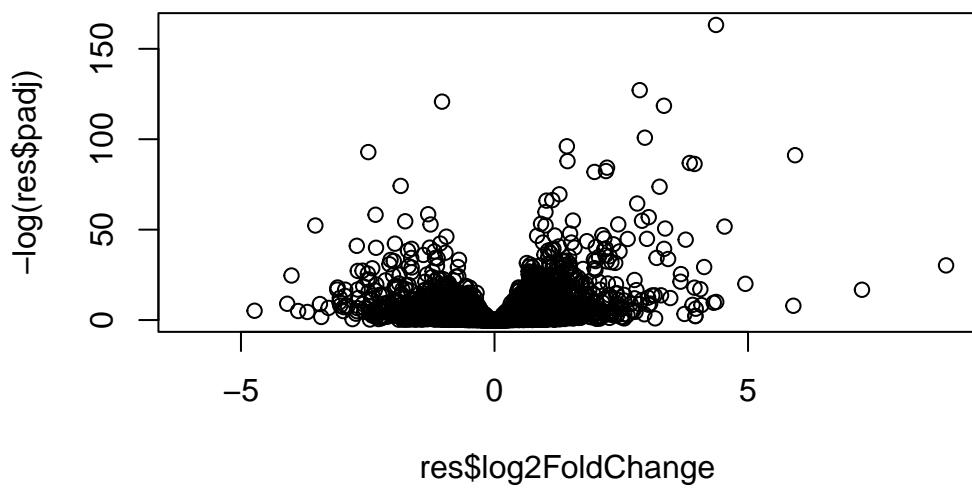
Volcano Plot

A very common and useful summary figure from this type of analysis is called a volcano plot - a plot of log2FC vs adj P-value. We use the `padj` - the adjusted p value for multiple testing correction.

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



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

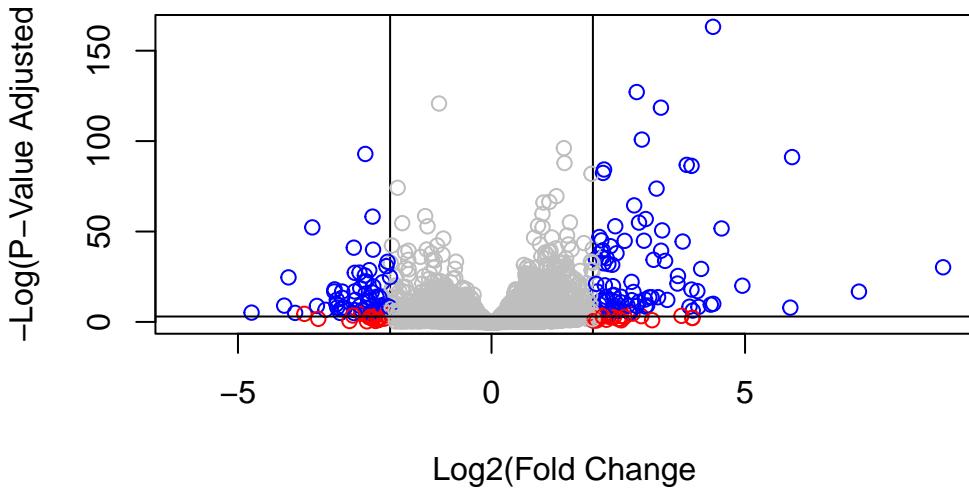


Add some colors to the plot:

```
my_cols <- rep("grey", nrow(res))
my_cols[abs(res$log2FoldChange)>2] <- "red"

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

plot(res$log2FoldChange, -log(res$padj), col = my_cols, ylab= "-Log(P-Value Adjusted", xla  
abline(h= -log(0.05))
```



Add Annotation data

```
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
ENSG00000000004	100.000000	0.000000	0.000000	0.000000	1.000000
ENSG00000000005	100.000000	0.000000	0.000000	0.000000	1.000000
ENSG00000000006	100.000000	0.000000	0.000000	0.000000	1.000000

```

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

```

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")
```

```
Attaching package: 'AnnotationDbi'
```

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

```
select
```

```
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"
```

```

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)

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    FIRRM
ENSG000000000938  NA          FGR

```

I also want entrez IDs

```

res$entrez <- mapIds(org.Hs.eg.db,
                      keys=row.names(res), # Our genenames
                      keytype="ENSEMBL",   # The format of our genenames
                      column="ENTREZID",  # The new format we want to add
                      multiVals="first")

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

```

```

res$name <- mapIds(org.Hs.eg.db,
                    keys=row.names(res), # Our genenames
                    keytype="ENSEMBL",   # The format of our genenames
                    column="GENENAME",    # The new format we want to add
                    multiVals="first")

'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
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           name
  <numeric> <character> <character>           <character>
ENSG000000000003 0.163035    TSPAN6       7105      tetraspanin 6
ENSG000000000005  NA          TNMD        64102     tenomodulin
ENSG000000000419 0.176032    DPM1        8813      dolichyl-phosphate m..
ENSG000000000457 0.961694    SCYL3       57147      SCY1 like pseudokina..
ENSG000000000460 0.815849    FIRRM       55732      FIGNL1 interacting r..
ENSG000000000938  NA          FGR         2268      FGR proto-oncogene, ..

```

Pathway Analysis

Now that I have added the necessary annotation data, I can talk to different databases that use these IDs.

We will use the `gage()` package to do geneset analysis (aka. pathway analysis, geneset enrichment, overlap 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)
```

We will use KEGG first ()

```
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"
```

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

```
foldchange <- res$log2FoldChange
names(foldchange) <- res$entrez
```

```
head(foldchange)
```

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

Run the analysis

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

Lets look at what is in our results here:

```
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

I can now use the returned pathway ID from KEGG as an input to the pathview() package to make pathway figures with our DEGs highlighted.

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

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

```
Info: Working in directory /Users/anna/Documents/BIMM 143/class13
```

Info: Writing image file hsa05310.pathview.png

