

# class13

Whitney Tran (PID:16781338)

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).

## Data Import

We have two input files, so-called

```
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		

```
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
```

There are 38694 genes in this dataset.

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

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

```
[1] 4
```

There are 4 control cell lines.

## Toy differential gene expression.

Time to do some analysis.

We have 4 control and 4 treated samples/experiments/columns.

Make sure the metadata id column matches the columns in our countdata.

```
colnames(counts)
```

```
[1] "SRR1039508" "SRR1039509" "SRR1039512" "SRR1039513" "SRR1039516"
[6] "SRR1039517" "SRR1039520" "SRR1039521"
```

```
metadata$id
```

```
[1] "SRR1039508" "SRR1039509" "SRR1039512" "SRR1039513" "SRR1039516"
[6] "SRR1039517" "SRR1039520" "SRR1039521"
```

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

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

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

```
all(c(T,T,T,F))
```

```
[1] FALSE
```

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

```
[1] TRUE
```

To start I will calculate the `control.mean` and `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

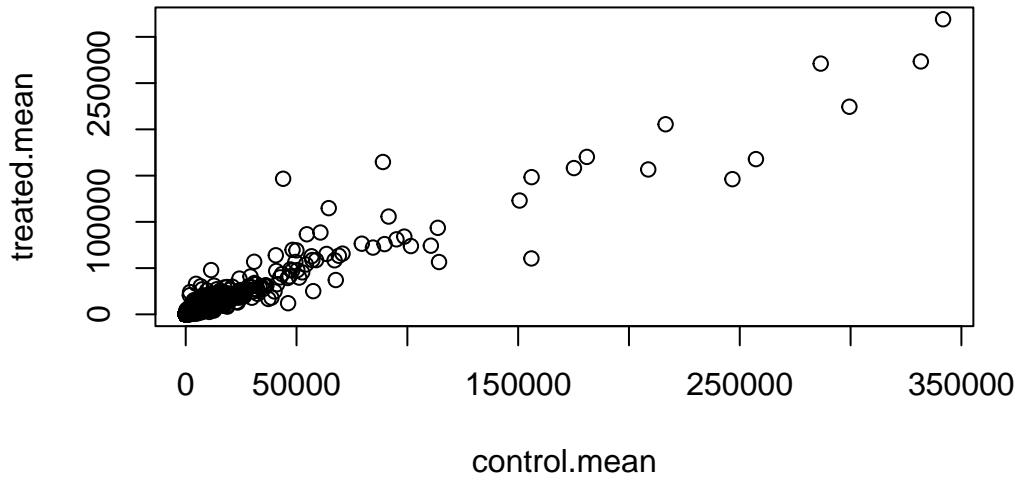
```
control inds <- metadata$dex == "control"  
control counts <- counts[ , control inds]  
control mean <- apply(control counts, 1, mean)  
  
treated inds <- metadata$dex == "treated"  
treated counts <- counts[ , treated inds]  
treated mean <- apply(treated counts, 1, mean)
```

Let's store these together for ease of book-keeping

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

Have a quick view of this data

```
plot(meancounts)
```

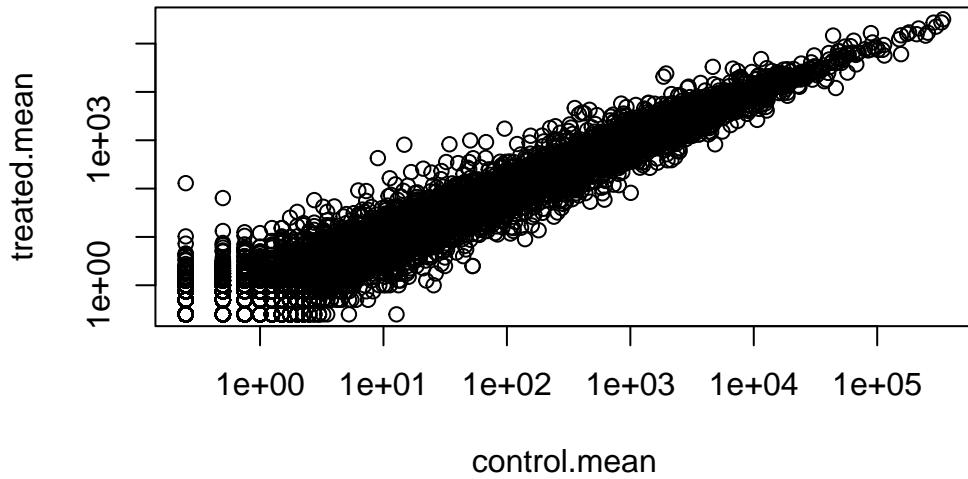


This data is screaming at us to log transform as it is so heavily skewer and over such a wide range.

```
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
```



I want to compare the treated and the control values here and we will use fold change in log2 units to do this. $\log_2(\text{Treated}/\text{Control})$

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

$\log_2(20/20)$

[1] 0

No difference

A doubling in the treated:

$\log_2(20/10)$

[1] 1

$\log_2(5/10)$

```
[1] -1
```

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

```
[1] 2
```

A common rule of thumb cut-off for calling a gene “differentially expressed” is a log2 fold-change value of either  $>+2$  or  $<-2$  for “up regulated” and “down regulated” respectively.

```
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

```
sum(meancounts$log2fc > +2, na.rm=T)
```

```
[1] 1846
```

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

```
to.rm.ind <- rowSums(meancounts[,1:2] ==0)>0
```

```
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
```

```
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
```

250 upregulated 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?

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

```
[1] 367
```

367 downregulated genes.

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

No we are missing stats!! Are these differences significant?

```
##DESeq analysis
```

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

```
#/ message:false  
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 object is masked from 'package:utils':
```

```
findMatches
```

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

```
expand.grid, I, unname
```

```
Loading required package: IRanges
```

```
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, 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
```

We have to use a specific data object for working with 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
```

Run our main analysis with the `DESeq()` function.

```
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

```

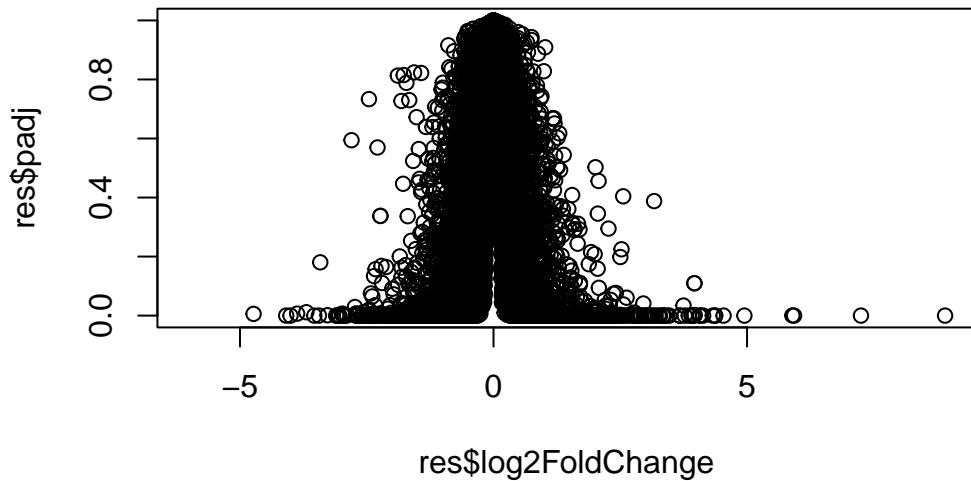
ENSG00000000419 520.134160      0.2061078  0.101059  2.039475 0.0414026
ENSG00000000457 322.664844      0.0245269  0.145145  0.168982 0.8658106
ENSG00000000460 87.682625      -0.1471420  0.257007 -0.572521 0.5669691
ENSG00000000938 0.319167      -1.7322890  3.493601 -0.495846 0.6200029
                    padj
<numeric>
ENSG00000000003 0.163035
ENSG00000000005 NA
ENSG00000000419 0.176032
ENSG00000000457 0.961694
ENSG00000000460 0.815849
ENSG00000000938 NA

```

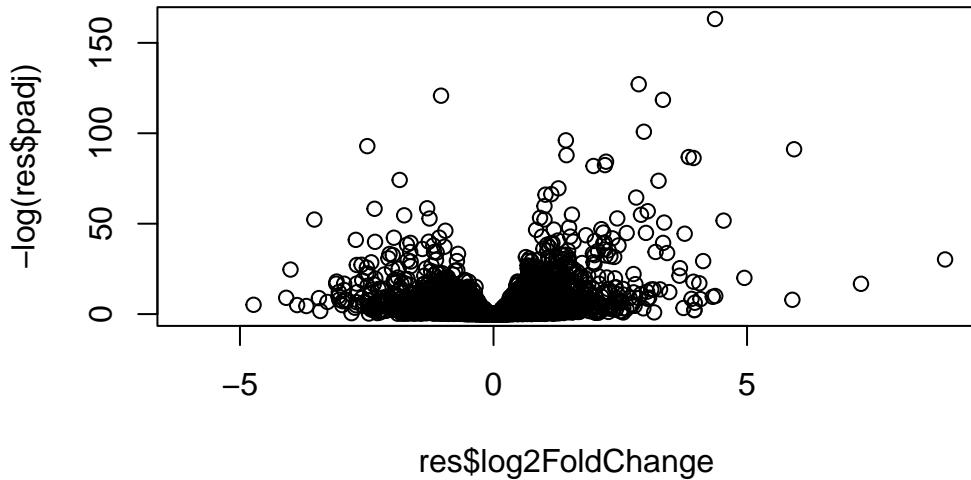
## Volcano Plot

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

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



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

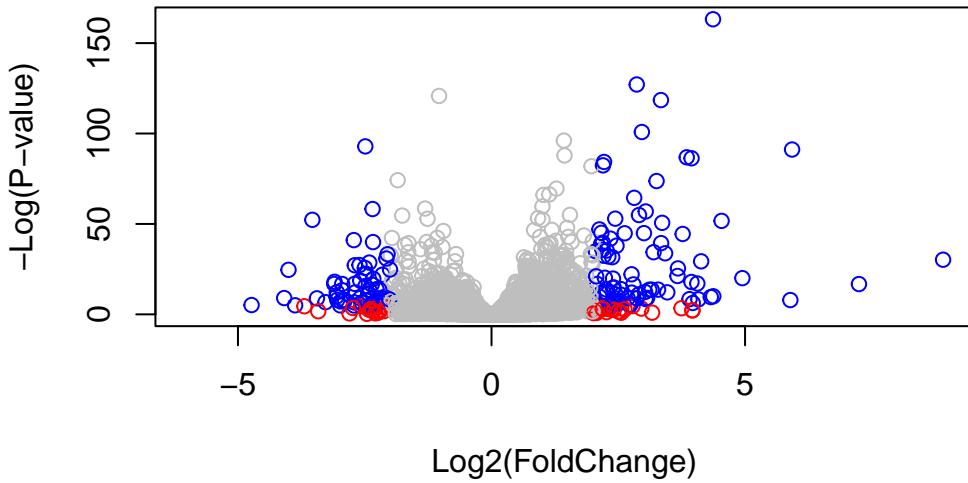


add some color and nice labels for this plot.

```
# 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)" )
```



## 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>
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
      <numeric>
ENSG000000000003 0.163035
ENSG000000000005   NA
ENSG000000000419 0.176032
ENSG000000000457 0.961694

```

```
ENSG000000000460 0.815849
ENSG000000000938 NA
```

```
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"
```

```
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

head(res)

log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 6 rows and 8 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
  <numeric> <character> <character>
ENSG000000000003  0.163035      TSPAN6      7105
ENSG000000000005      NA      TNMD      64102
ENSG000000000419  0.176032      DPM1      8813
ENSG000000000457  0.961694      SCYL3      57147
ENSG000000000460  0.815849      FIRRM      55732
ENSG000000000938      NA      FGR       2268

```

```
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
```

## Pathway Analysis

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

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

```
BiocManager::install( c("pathview", "gage", "gageData") )
```

We will use KEGG first()

```
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)
```

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

```
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)
```

```
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 pathways IDs from KEGG as input to the `pathview` package to make pathway figures with our DEGs highlighted.

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

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

Info: Working in directory /Users/whitneytran/Desktop/BIMM143/class13

Info: Writing image file hsa05310.pathview.png

